The Addison-Wesley Signature Series

A Mike Cohn Signature Book

Основы Scrum

Практическое руководство по гибкой разработке ПО

Кеннет С. Рубин

Предисловие Майкла Кона и Рона Джеффриса

Essential Scrum

A Practical Guide to the Most Popular Agile Process

KENNETH S. RUBIN

Addison-Wesley

Upper Saddle River, NJ · Boston · Indianapolis · San Francisco
New York · Toronto · Montreal · London · Munich · Paris · Madrid
Capetown · Sydney · Tokyo · Singapore · Mexico City

Основы Scrum

Практическое руководство по гибкой разработке ПО

Кеннет С. Рубин

Диалектика
Москва • Санкт-Петербург
2020

ББК 32.973.26-018.2.75
Р82
УДК 681.3.07

ООО "Диалектика"

Зав. редакцией *С.Н. Тригуб*

Перевод с английского и редакция *И.В. Берштейна*

По общим вопросам обращайтесь в издательство "Диалектика" по адресу:
info@dialektika.com, http://www.dialektika.com

Рубин, Кеннет С.

Р82 Основы Scrum: практическое руководство по гибкой разработке ПО. : Пер. с англ. — СПб. : ООО "Диалектика", 2020. — 544 с. : ил. — Парал. тит. англ.

ISBN 978-5-907144-96-5 (рус.)

ББК 32.973.26-018.2.75

Научно-популярное издание
Кеннет С. Рубин

Основы Scrum:
практическое руководство по гибкой разработке ПО

ООО "Диалектика", 195027, Санкт-Петербург, ул. Магнитогорская, д. 30, лит. А, пом. 848

ISBN 978-5-907144-96-5 (рус.)
ISBN 978-0-13-704329-3 (англ.)

Оглавление

Отзывы о книге 19
Вступительное слово Майка Кона 25
Вступительное слово Рона Джеффриза 27
Предисловие 29
Благодарности 33
Об авторе 37
Глава 1. Введение 39

Часть I. Основные понятия 51

Глава 2. Инфраструктура Scrum 53
Глава 3. Принципы гибкой разработки 71
Глава 4. Спринты 109
Глава 5. Требования и пользовательские истории 131
Глава 6. Задел продукта 155
Глава 7. Оценивание и скорость работы 177
Глава 8. Технический долг 201

Часть II. Роли 229

Глава 9. Владелец продукта 231
Глава 10. Scrum-мастер 255
Глава 11. Команда разработчиков 267
Глава 12. Структуры Scrum-команд 287
Глава 13. Руководители 301

Часть III. Планирование 323

Глава 14. Принципы планирования в Scrum 325
Глава 15. Многоуровневое планирование 335
Глава 16. Планирование портфеля заказов 345
Глава 17. Планирование продукта (выработка общего замысла) 369
Глава 18. Планирование выпуска (долгосрочное планирование) 393

Часть IV. Проведение спринтов 423

Глава 19. Планирование спринта 425
Глава 20. Выполнение спринта 439
Глава 21. Подведение итогов спринта 457
Глава 22. Ретроспектива спринта 471
Глава 23. Дальнейшие действия 495
Приложение А. Словарь терминов 501
Приложение Б. Библиография 525
Предметный указатель 529

Содержание

Отзывы о книге 19
Вступительное слово Майка Кона 25
Вступительное слово Рона Джеффриза 27
Предисловие 29
Благодарности 33
Об авторе 37

Глава 1. Введение 39
Что такое Scrum 39
Истоки Scrum 41
Причины для применения Scrum 42
Результаты, достигнутые в компании Genomica 43
Чем Scrum может помочь вам 44
Комплексная область 47
Сложная область 47
Простая область 48
Хаотическая область 48
Беспорядочная область 48
Работа с прерываниями 49
Заключение 50

Часть I. Основные понятия 51
Глава 2. Инфраструктура Scrum 53
Краткий обзор 53
Роли в Scrum 54
Владелец продукта 55
Scrum-мастер 56
Команда разработчиков 56
Виды деятельности и артефакты Scrum 57
Задел продукта 59
Спринты 61
Планирование спринта 62
Выполнение спринта 64
Ежедневные летучки 65
Результаты спринта 67

Подведение итогов спринта 68
Ретроспектива спринта 69
Заключение 70

Глава 3. Принципы гибкой разработки 71
Краткий обзор 71
Изменчивость и неопределенность 73
Учет полезной изменчивости 73
Применение итеративной и инкрементной разработки 75
Эффективное использование изменчивости путем обследования, адаптации и прозрачности 78
Одновременное сокращение всех форм неопределенности 79
Прогнозирование и адаптация 80
Наличие свободы выбора 81
Способность принять, что нельзя все сделать сразу правильно 82
Предпочтение адаптивного, исследовательского подхода 83
Учет изменений экономически обоснованным способом 84
Оптимальное сочетание предварительно прогнозирующих работ с адаптивными своевременными работами 88
Утвержденное обучение 89
Быстрое утверждение важных предположений 89
Эффективное использование нескольких параллельных петель обучения 90
Организация рабочего процесса для получения быстрой ответной реакции 91
Незавершенные работы 93
Использование экономически обоснованных объемов партий 93
Учет запасов и управление ими для оптимального хода работ 95
Акцент на простаивающей работе, а не на простаивающих работниках 96
Учет стоимости задержки 99
Прогресс 100
Адаптация к информации в реальном времени и перепланирование 100
Определение достигнутого прогресса путем проверки достоверности рабочих ресурсов 101
Акцент на доставке ценности 101
Исполнение 102
Быстрое продвижение, но без спешки 102
Упор на качество 103
Сведение церемоний к минимуму 104
Заключение 105

Глава 4. Спринты 109
Краткий обзор 109
Ограничение по времени 110
Установление предела для незавершенных работ 110
Побуждение к назначению приоритетов 111
Демонстрация достигнутого прогресса 111
Исключение излишнего перфекционизма 112
Стимулирование завершения 112
Повышение предсказуемости 112
Краткосрочность спринтов 112
Простота планирования 112
Быстрота ответной реакции 112
Повышение эффективности инвестиций 113
Ограничение ошибок 113
Оживление интереса 114
Частость контрольных точек 114
Постоянная продолжительность 116
Преимущества размеренного ритма 116
Упрощенное планирование 117
Запрет на изменение цели спринта 118
Что такое цель спринта 118
Взаимное обязательство 118
Изменение в сравнении с уточнением цели спринта 119
Последствия изменения цели спринта 120
Прагматичность 121
Ненормальное завершение спринта 122
Критерий готовности 124
Что такое критерий готовности 125
Возможное развитие критерия готовности во времени 127
Критерии готовности и приемки 128
Критерий готовности и неполная готовность 129
Заключение 130

Глава 5. Требования и пользовательские истории 131
Краткий обзор 131
Проведение обсуждений 134
Постепенное уточнение требований 135
Что такое пользовательские истории 135
Карточка 136
Обсуждение 137
Подтверждение 138

Уровень детализации 140
Критерии INVEST оценки пригодности историй 142
Независимость 142
Обсуждаемость 143
Ценность 144
Оцениваемость 146
Подходящий размер (мелкость) 146
Тестируемость 147
Нефункциональные требования 147
Истории приобретения знаний 148
Собирание историй 150
Совещания по написанию пользовательских историй 151
Построение карт историй 152
Заключение 154

Глава 6. Задел продукта 155

Краткий обзор 155
Элементы задела продукта 155
Характеристики пригодного задела продукта 157
Детализация должным образом 157
Развитие 158
Оцениваемость 159
Приоритизация 160
Упорядочение задела 161
Что означает упорядочение задела 161
Кто упорядочивает задел продукта 162
Когда происходит упорядочение задела продукта 163
Критерий подготовленности 165
Управление потоками 167
Управление потоком выпуска 167
Управление потоком спринта 168
Требующиеся заделы продуктов и их количество 169
Что такое продукт 170
Иерархические заделы для крупных продуктов 171
Один задел продукта на несколько команд 173
Несколько продуктов на одну команду 174
Заключение 176

Глава 7. Оценивание и скорость работы 177

Краткий обзор 177
Что и когда оценивается 179

Оценивание элементов из задела портфеля заказов 179
Оценивание элементов из задела продукта 180
Оценивание задач 181
Принципы оценивания элементов из задела продукта 181
Коллективное оценивание 182
Оценки не являются обязательствами 182
Акцент на правильности, а не точности оценок 184
Употребление относительных размеров 184
Единицы оценивания элементов в заделе продукта 187
Очки за историю 187
Идеальные дни 188
Покер планирования 189
Оценочная шкала 189
Порядок проведения покера планирования 190
Преимущества покера планирования 193
Что такое скорость работы 193
Расчет скоростного диапазона 194
Прогнозирование скорости работы 196
Влияние на скорость работы 196
Злоупотребление скоростью работы 199
Заключение 200

Глава 8. Технический долг 201

Краткий обзор 201
Последствия накапливания технического долга 203
Непредсказуемый переломный момент 204
Увеличение времени доставки 205
Значительное количество дефектов 205
Увеличение затрат на разработку и сопровождение 205
Отмирание продукта 206
Снижение предсказуемости 206
Низкая эффективность 207
Общее разочарование 207
Снижение удовлетворенности заказчика 207
Причины технического долга 208
Нажим с целью уложиться в срок 208
Попытка ошибочно увеличить скорость работы 209
Миф об увеличении скорости работы за счет сокращения тестирования 210
Образование одних долгов из других 211
Технический долг должен поддаваться управлению 212

Контроль накапливания технического долга 212
Применение норм надлежащей инженерной практики 213
Применение строгого критерия готовности 214
Правильное понимание экономических последствий технического долга 214
Выявление технического долга 218
Выявление технического долга на деловом уровне 218
Выявление технического долга на инженерном уровне 219
Обслуживание технического долга 221
Неполное погашение технического долга 222
Применение правила бойскаутов 224
Постепенное погашение технического долга 225
Погашение высокопроцентного технического долга в первую очередь 226
Погашение технического долга при выполнении ценной для заказчика работы 226
Заключение 228

Часть II. Роли 229

Глава 9. Владелец продукта 231

Краткий обзор 231
Основные обязанности 232
Ведение экономического анализа 232
Участие в планировании 235
Упорядочение задела продукта 235
Определение критериев приемки и проверка их соответствия 235
Сотрудничество с командой разработчиков 236
Сотрудничество с участниками проекта 238
Характеристики и навыки 238
Навыки в предметной области 238
Навыки работы с людьми 239
Принятие решений 240
Ответственность 240
Типичный график работы владельца продукта 241
Кто должен быть владельцем продукта 244
Внутренняя разработка 244
Коммерческая разработка 245
Субподрядная разработка 248
Разработка компонентов 249
Сочетание роли владельца продукта с другими ролями 250
Команда владельцев продукта 251

Заместитель владельца продукта 252
Главный владелец продукта 253
Заключение 254

Глава 10. Scrum-мастер 255
Краткий обзор 255
Основные обязанности 255
Наставничество 255
Лидерство служителя 257
Ответственность за процесс разработки 257
Защита от вмешательства 257
Устранение препятствий 257
Инициирование перемен 258
Характеристики и навыки 258
Компетентность 258
Выведывание 259
Терпеливость 259
Готовность сотрудничать 260
Покровительство 260
Открытость 260
Типичный график работы Scrum-мастера 261
Выполнение роли Scrum-мастера 262
Кто должен быть Scrum-мастером 262
Должен ли Scrum-мастер выполнять свою роль весь рабочий день 263
Сочетание роли Scrum-мастера с другими ролями 264
Заключение 265

Глава 11. Команда разработчиков 267
Краткий обзор 267
Команды с конкретными ролями 267
Основные обязанности 268
Выполнение спринта 268
Обследование и адаптация каждый день 269
Упорядочение задела спринта 269
Планирование спринта 270
Обследование и адаптация продукта и процесса его разработки 270
Характеристики и навыки 270
Самоорганизация 270
Межфункциональная неоднородность и достаточность 273
Т-образные навыки 274
Мушкетерские отношения 276
Высокая пропускная способность передачи информации 278

Открытость в общении 279
Оптимальный штат 280
Целеустремленность и обязательность 281
Способность работать в постоянном темпе 283
Долговечность 284
Заключение 286

Глава 12. Структуры Scrum-команд 287

Краткий обзор 287
Команды для функциональных средств и компонентов 288
Координирование работы многих команд 293
Схватка над схватками 294
Выпускной поезд 296
Заключение 299

Глава 13. Руководители 301

Краткий обзор 301
Формирование команд 303
Определение границ 303
Постановка ясной возвышенной цели 304
Составление команд 304
Изменение состава команд 305
Наделение команд полномочиями 306
Опека команд 307
Стимулирование людей 308
Развитие компетентности 308
Руководство функциональными областями 309
Сохранение целостности команд 310
Согласование и адаптация производственной среды 310
Продвижение ценностей гибкой разработки 311
Устранение организационных препятствий 311
Согласование работы во внутренних группах 311
Согласование работы с партнерами 312
Управление ходом работ по созданию ценности 313
Системный подход 313
Экономический анализ 314
Контроль показателей и отчетов 314
Руководители проектов 315
Обязанности руководителя проекта в Scrum-команде 315
Сохранение отдельной роли руководителя проекта 317
Заключение 321

Часть III. Планирование 323
Глава 14. Принципы планирования в Scrum 325
Краткий обзор 325
Не следует допускать, что планы можно составить заранее 326
Предварительное планирование должно быть полезным, но не чрезмерным 326
Варианты планирования остаются в силе до самого последнего момента 328
Акцент на адаптации и перепланировании, а не на соответствии плану 328
Правильное управление запасами планирования 331
Предпочтение более мелких и частых выпусков 332
Планирование с целью быстрого обучения и резкая смена стратегии по мере надобности 334
Заключение 334
Глава 15. Многоуровневое планирование 335
Краткий обзор 335
Планирование портфеля заказов 337
Планирование продукта (выработка общего замысла) 337
Концепция продукта 337
Задел продукта на верхнем уровне 338
График выпуска продукта 338
Планирование выпуска 340
Планирование спринта 341
Ежедневное планирование 343
Заключение 344
Глава 16. Планирование портфеля заказов 345
Краткий обзор 345
Временны*е* рамки 346
Участники 346
Процесс 347
Стратегии календарного планирования 349
Оптимизация прибылей в течение срока службы продуктов 349
Расчет стоимости задержки 350
Оценивание ради правильности, а не точности 354
Стратегии управления притоком продуктов 355
Применение экономического фильтра 356
Уравновешивание скорости поступления и отправления 357
Быстрый охват возникающих возможностей 359

Планирование более мелких, но частых выпусков 360
Стратегии управления оттоком продуктов 361
Акцент на простоях в работе, а не на простаивающих работниках 362
Установление предела для незавершенных работ 362
Ожидание полного задействования команд 364
Стратегии управления внутрипроцессными продуктами 365
Применение граничных экономических показателей 365
Заключение 367

Глава 17. Планирование продукта (выработка общего замысла) 369
Краткий обзор 369
Временны*е* рамки 370
Участники 371
Процесс 371
Пример выработки общего замысла 373
Составление концепции продукта 374
Создание задела продукта на высоком уровне 378
Составление графика выпуска продукта 379
Другие виды деятельности 382
Экономически обоснованная выработка общего замысла 384
Нацеливаться на реалистичный порог доверия 385
Делать акцент на короткий горизонт 387
Действовать быстро 388
Платить за утвержденное обучение 389
Пользоваться методом постепенного/временного финансирования 390
Учиться быстро и резко сменять концепцию 391
Заключение 392

Глава 18. Планирование выпуска (долгосрочное планирование) 393
Краткий обзор 393
Временны*е* рамки 394
Участники 395
Процесс 395
Ограничения на выпуск 397
Все задано жестко 398
Жестко заданные сроки и объем работ 398
Жестко заданный объем работ 400
Жестко заданные сроки 401
Переменное качество 401
Коррекция ограничений 402
Упорядочение задела продукта 402

Уточнение состава минимально выпускаемых функциональных средств 403
Составление карты спринтов 405
Планирование выпуска с жестко заданными сроками 408
Планирование выпуска с жестко заданным объемом работ 414
Расчет затрат на выпуск 416
Сообщение о достигнутом прогрессе 418
Сообщение о достигнутом прогрессе в выпуске с жестко заданным объемом работ 418
Сообщение о достигнутом прогрессе в выпуске с жестко заданными сроками 420
Заключение 422

Часть IV. Проведение спринтов 423

Глава 19. Планирование спринта 425

Краткий обзор 425
Временны*е* рамки 425
Участники 426
Процесс 426
Методы планирования спринта 429
Двухэтапное планирование спринта 429
Одноэтапное планирование спринта 430
Определение возможностей команды 431
Что означают возможности команды 431
Оценивание возможностей команды в очках за историю 433
Оценивание возможностей команды в человеко-часах 433
Отбор элементов из задела продукта 435
Приобретение уверенности 435
Уточнение цели спринта 438
Окончательное взятие обязательств 438
Заключение 438

Глава 20. Выполнение спринта 439

Краткий обзор 439
Временны*е* рамки 439
Участники 440
Процесс 440
Планирование выполнения спринта 441
Управление ходом работ 442
Параллельная работа и роение 442
С какой работы следует начинать выполнение спринта 445

Как организовать работу над отдельными задачами 446
Какая работа должна быть сделана 446
Кто должен выполнять работу 447
Ежедневная летучка 448
Исполнение задач — нормы инженерной практики 448
Сообщение 450
Доска задач 450
Диаграмма сгорания спринта 451
Диаграмма выгорания спринта 454
Заключение 455

Глава 21. Подведение итогов спринта 457
Краткий обзор 457
Участники 458
Подготовительная работа 460
Определение состава приглашенных 461
Календарное планирование мероприятия 461
Подтверждение готовности работы, выполняемой в течение спринта 462
Подготовка к демонстрации 463
Распределение обязанностей 464
Метод 464
Краткий обзор 465
Демонстрация 466
Обсуждение 467
Адаптация 467
Вопросы подведения итогов спринта 468
Одобрения 468
Нерегулярная посещаемость 469
Крупные проектные работы 470
Заключение 470

Глава 22. Ретроспектива спринта 471
Краткий обзор 471
Участники 473
Подготовительная работа 475
Выбор средоточия ретроспективы спринта 475
Выбор упражнений 476
Сбор объективных данных 477
Структура ретроспективы спринта 477
Метод 478
Создание атмосферы 480

Создание общего контекста 480
Сейсмограмма эмоций 483
Выявление наблюдений 483
Определение действий 486
Выбор действий 488
Завершение мероприятия 489
Доведение дела до логического конца 490
Вопросы проведения ретроспективы спринта 491
Заключение 494

Глава 23. Дальнейшие действия 495

Конечного состояния в Scrum не существует 495
Обнаружение собственного пути 496
Соблюдение общих норм передовой практики 496
Обнаружение пути дальнейшего продвижения по методике Scrum 498
Итак, за дело! 499

Приложение А. Словарь терминов 501

Краткий обзор 501
Определения 501

Приложение Б. Библиография 525
Предметный указатель 529

Отзывы о книге

"Эта книга порадует инструкторов по гибкой разработке. Кенни Рубин создал для нас незаменимый ресурс. Есть ли у вас начальник, который "не в курсе дела"? Вручите ему эту книгу и попросите его хотя бы бегло просмотреть первые две ее главы, где подробно поясняется, насколько методика Scrum менее рискованна, чем плановое управление. Она написана специально для тех, кто руководит проектами, на их языке. А что поможет команде разработчиков прийти к общему пониманию Scrum? Им и вам поможет особый язык визуальных образов, употребляемый в этой книге. Оба эти способа изложения материала книги помогут вам в обучении команд методике Scrum. Употребите ее с пользой".

Лисса Эдкинс (Lyssa Adkins),
наставник инструкторов по гибкой разработке
из Института обучения гибкой разработке
(Agile Coaching Institute), автор книги *Coaching Agile Teams*

"Это одно из самых лучших и исчерпывающих описаний основ инфраструктуры Scrum! Эта книга для всех (начинающих и имеющих опыт применения Scrum), кого интересуют наиболее важные аспекты данного процесса. Кенни отлично справился с задачей выделить главные принципы инфраструктуры Scrum в простой формат с неотразимыми иллюстрациями. Обучая многие команды разработчиков методике Scrum, я постоянно обращаюсь к материалу на данную тему, находя в нем новые способы помочь командам в изучении и практическом применении инфраструктуры Scrum. В течение более десяти лет мне не раз приходилось сталкиваться с неправильной трактовкой и неудачной реализацией Scrum в крупных компаниях и среди поставщиков инструментальных средств. Чтение этой книги поможет вам вернуться назад к основам и сосредоточиться на самом главном".

Джо Балистриери (Joe Balistrieri),
руководитель отдела технологических
разработок в компании Rockwell Automation

"Руководство отделов информационных технологий, которое не спешит внедрять методики гибкой разработки, извлечет немалую выгоду, предоставив экземпляр этой книги каждому руководителю проектов и ответственному за их доставку клиентам. Кенни Рубин изложил в своей книге весь материал

по прагматичному анализу деловой ситуации и производственного процесса, необходимый любому отделу информационных технологий для успешной реализации Scrum".

Джон Ф. Бауэр III (John F. Bauer III),
опытный специалист по доставке технических решений
для крупных интернет-магазинов электроники и вычислительной техники.

"Обширный опыт Кенни в качестве консультанта, инструктора, а в прошлом руководителя организации Scrum Alliance очевидно проявляется в этой книге. Помимо основ и введения в Scrum, в этой книге обсуждаются многие вопросы, волнующие руководителей проектов. Она помогает лучше понять общую картину и наставляет руководство организаций на поддержку и участие в их Scrum-командах для успешного перехода к гибкой разработке".

Самир С. Брендр (Sameer S. Bendre),
управляющий по работе с клиентами,
специалист по управлению проектами,
старший консультант в компании 3i Infotech Inc.

"Если вы только начинаете осваивать гибкую разработку или методику Scrum, эта книга поможет вам положить отличное начало. Примеры и описания в ней ясны и выразительны. Читая эту книгу, невольно ловишь себя на том, что вопрос возникает еще до того, как он разъясняется в тексте книги".

Йоханнес Бродвалл (Johannes Brodwall),
главный архитектор решений в компании Steria Norway.

"Ясность хорошо структурированных пояснений Кенни невольно вызывает ощущение Smalltalk — среды разработки, где он работал многие годы и откуда зародились методики Scrum и экстремального программирования. В этой книге сводятся воедино принципы управления гибкой разработкой, которые попадают прямо в цель и, без сомнения, направляют к более эффективному подходу к гибкой разработке".

Роуэн Баннинг (Rowan Bunning),
основатель компании Scrum WithStyle.

"Ныне существует немало книг по Scrum, но в этой книге принят новый подход: проверка данной методики в реальных условиях, хорошо знакомых тем, кто практически занимается разработкой программного обеспечения. Кенни пользуется реальными примерами и ясными иллюстрациями, чтобы показать, что именно служит прочным основанием для успешной гибкой разработки. Читатели поймут значение, которое придается достижению качества, а также ту сермяжную правду, что нельзя получить все сразу. Мы должны работать

постепенно, обучаясь по ходу дела. Несмотря на слово “Scrum” в названии книги, в ней рассматриваются эффективные практические методики из всей обширной области гибкой разработки, чтобы помочь руководителям и их командам добиться успеха”.

Лиза Криспин (Lisa Crispin),
соавтор книги *Agile Testing*.

“Кенни Рубину удалось написать книгу, которую я бы советовал прочитать всем, кто имеет какое-то отношение к разработке по методике Scrum! В этой книге он излагает все, что требуется знать о Scrum и даже больше!”

Мартин Девос (Martine Devos),
инициатор Scrum в Европе
и аттестованный инструктор по Scrum.

“За последние несколько лет мне пришлось рецензировать целый ряд книг по гибкой разработке, и в связи с этим невольно возникает вопрос: ”А нужна ли еще одна такая книга?” Что касается книги Кенни, то я твердо уверен, что такая книга нужна. Очень ценно извлечь пользу из разных, опытных точек зрения на часто встречающийся и нужный материал. И одну из таких ценных точек зрения предлагает в своей книге Кенни. К особенностям этой книги относится любопытная ее художественная иллюстрация — новый язык визуальных образов, изобретенный Кенни для пояснения Scrum и гибкой разработки. Считаю, что этот ценный дополнительный материал поможет читателям расширить свои представления о возможном применении Scrum”.

Скотт Дункан (Scott Duncan),
наставник и инструктор по Scrum и гибкой разработке.

“Все, кому приходилось обучать методике Scrum или принимать участие в работе Scrum-команды, найдут в этой книге отличное руководство к действию. В этой книге подробно поясняется, как повысить гибкость разработки путем реализации Scrum-процессов и как разбивать сложные проекты на поддающиеся управлению инициативы (так называемые “спринты”). Кенни Рубин предоставляет для анализа немало уместных практических примеров того, что оказалось работоспособным (или не работоспособным) в различных организациях. Простая подача материала и деловая графика упрощает и ускоряет поиск ответов на конкретные вопросы гибкой разработки. В любой организации, где ищут пути перехода от традиционной водопадной методики к гибкой методике разработки, обнаружат в этой книге обстоятельное руководство, направляющее по нужному пути”.

Джулия Фрэйзер (Julia Frazier),
ответственная за выпуск продукции.

"Разрабатывать программное обеспечение непросто. Еще труднее принять новый подход к работе над проектом. Эта книга предлагает многие пути преодоления скрытых препятствий, ускорения способности команды достигать коммерческой ценности разрабатываемой продукции и успешной работы по методике Scrum. Как бы мне хотелось иметь такую книгу, когда я только начинал пользоваться Scrum".

Гейр Хедемарк (Geir Hedemark),
руководитель разработки в компании Basefarm AS.

"Убежден, что эта книга станет основательным справочным пособием для следующего поколения тех, кто применяет методику Scrum на практике. Это не только самое исчерпывающее введение в Scrum из всех ныне существующих, но и очень грамотно и просто написанная книга с фантастическими иллюстрациями Scrum на новом языке визуальных образов. Мало того, Кенни делится своими наблюдениями и личным опытом, из которого мы можем, конечно, извлечь немалую пользу".

Илан Гольдшейн (Ilan Goldstein),
руководитель по гибким решениям в компании Reed Elsevier.

"Методика Scrum изящно проста, хотя и обманчиво сложна. В своей книге Кенни Рубин шаг за шагом раскрывает сложности, сохраняя существенную простоту. Реальный опыт в сочетании с наглядными иллюстрациями оживляет Scrum. Эта книга обязательна для чтения как старшим руководством, так и членами команд разработчиков, если они собираются реализовывать Scrum в своей организации. И конечно, я рекомендую эту книгу своим учащимся".

Джон Хебли (John Hebley),
компания Hebley & Associates.

"Кенни раскрывает в своей книге немало мудрости и знаний, делясь ценными и обширными представлениями о практическом применении методики Scrum гибкой разработки. Это настольная книга для всех, кто только начинает осваивать гибкую разработку или стремится достичь большей зрелости, непрерывно совершенствуясь в своей организации".

Дэвид Лузквинос (David Luzquiños),
руководитель отдела внедрения гибкой разработки,
инструктор по гибкой разработке на бирже ставок Betfair.

"Кенни Рубин продолжает вносить ясность и понимание в прагматичный способ принятия гибкой разработки. В одной руке он держит формальное или идеальное определение методики Scrum, а в другой — ее прагматичное

применение. Многолетний опыт своих семинаров и учебных курсов он выкладывает на стол перед читателями данной книги. Если вы только собираетесь вступить на путь внедрения гибкой разработки или ищете совета, находясь посредине этого пути, непременно приобретите экземпляр данной книги".

Куан Маллиган (Cuan Mulligan),
инструктор по совместному обучению гибкой разработке.

"Через десять лет после выхода в свет первой книги по Scrum настало время соединить основные свойства инфраструктуры Scrum с практическим опытом и подходами, накопившимися за последнее десятилетие. Кенни Рубин делает это удовлетворительным и недогматическим способом. Вниманию читателя предлагается прагматичный взгляд на Scrum, из чего он может узнать, как и когда лучше всего применять Scrum для достижения коммерческих выгод".

Ив Стальжиз (Yves Stalgies),
доктор философских наук, руководитель отдела
информационных технологий (`www.etracker.com`).

"Принятие методики Scrum оказывается наиболее успешным, если все (даже косвенно) заинтересованные в разработке продукции лица ясно понимают основы. В этой книге делается идеальный обзор как общей картины, так и подробностей в доступном стиле. Она, безусловно, станет образцовым справочным пособием".

Кевин Турески (Kevin Tureski),
директор формы Kevin Tureski Consulting.

Посвящения

Моей жене Дженин — за всю ее нежную поддержку; моим сыновьям Ионе и Ашеру — за то, что они меня вдохновляли; моему отцу Манни — за то, что он научил меня ценить тяжелый труд; моей матери Джойс — за то, что она показала мне, как выглядит настоящее мужество (да будет ее память благословенна).

Вступительное слово Майка Кона

Сегодня я обедал в ресторане быстрого питания “Burger King”. Вывеска на стене этого ресторана гласила “Родина воппера”, а ниже перечислялось множество различных способов заказать воппер — фирменный гамбургер сети ресторанов быстрого питания “Burger King”. Если существует множество способов приготовить этот гамбургер в различных сочетаниях с соленым огурцами, помидорами, латуком, сыром и прочими ингредиентами, то еще больше способов имеется реализовать Scrum. И в то же время не существует единственно верного способа реализовать Scrum, а только более или менее пригодные способы.

Эта книга поможет читателям найти наилучшие способы реализовать Scrum. Это не нормативный справочник, предписывающий, что нужно делать, а скорее учебное пособие по основным принципам успешной организации Scrum, позволяющее выбрать путь следования этим принципам на практике. Например, не существует единственно верного для всех команд разработчиков способа планировать спринт. То, что годится для одной компании или проекта, не подходит для другой компании или проекта. И поэтому автор книги дает читателю возможность выбора среди разных вариантов. Он описывает общую структуру, по которой команды разработчиков должны планировать спринты в Scrum, поясняет, каких результатов следует ожидать от такого планирования, а также предлагает пару альтернативных и в равной степени работоспособных подходов к планированию спринтов. Но окончательное решение остается за каждой командой разработчиков. И в этом им может оказать посильную помощь данная книга.

Неожиданное преимущество данной книге сообщает язык, который автор вводит для визуального сообщения о Scrum. Иллюстрации к книге служат очень удобным дополнением ее текста, и я подозреваю, что в последующих дискуссиях на тему Scrum они станут общепринятыми.

Потребность в подобной книге возникла давно. В начале понятие Scrum употреблялось в узком смысле. Первой на эту тему была книга *Wicked Problems, Righteous Solutions*, написанная Дегрейсом (DeGrace) и Сталем (Stahl) в 1990 году, где Scrum было посвящено всего лишь шесть страниц. Но за последующие 20 лет понятие Scrum значительно расширилось внедрением и уточнением новых ролей, совещаний и артефактов. С каждым нововведением повышался риск утратить саму суть Scrum: планирование работ в команде разработчиков,

выполнение этих работ и анализ того, что и как команде удалось добиться совместными усилиями.

Автор книги возвращает нас к самой сути Scrum, исходя из которой команды разработчиков могут приступить к принятию решений, необходимых для реализации Scrum по-своему. Эта книга служит руководством, помогающим командам разработчиков выбрать среди множества возможных вариантов реализации Scrum свой единственный путь, ведущий к успеху.

Майк Кон,
автор книг *SScrum: гибкая разработка ПО*,
Agile Estimating and Planning и *User Stories Applied*
(`www.mountaingoatsoftware.com`).

Вступительное слово Рона Джеффриза

Когда Кенни попросил меня написать вступительное слово к данной книге, я было подумал, что сделаю это быстро и легко. Мне казалось, что эта книга должна быть кратким и простым описанием того, что собой представляет Scrum. Я знал, чем занимается Кенни, и поэтому считал, что чтение его книги будет приятным и недолгим занятием. Казалось бы, что может быть лучше!

Каково же было мое удивление и восхищение, когда я обнаружил в этой книге практически все, что нужно знать о методике Scrum, начиная с первого дня ее применения. И Кенни на этом не останавливается. Он начинает с главных идей, включая принципы, положенные в основу всех методик гибкой разработки, а затем делает краткий обзор инфраструктуры Scrum. Далее он все больше и больше углубляется в данный процесс. Но от этого чтение книги не теряет свою занимательность, а охват темы — всесторонний характер.

Кенни довольно подробно описывает планирование, анализ требований, истории, задел, оценивание и скорость работы. Затем он раскрывает основные принципы и помогает нам, читателям, усвоить все уровни планирования и промежутки времени. Он описывает порядок планирования, выполнения, анализа и улучшения спринтов. А между тем он не просто преподает основы, но выделяет главные вопросы, которые приходится решать по ходу дела.

На мой взгляд, для Scrum и гибкой разработки требуется достаточная квалификация разработчиков, чтобы их команда могла доставлять с каждым спринтом реальное, работоспособное, ориентированное на конкретную предметную область программное обеспечение. В своей книге Кенни помогает нам лучше понять, как надежно и грамотно пользоваться такими понятиями, как скорость работы и технический долг. Оба этих понятия имеют решающее значение, и поэтому я обращаю на них особое внимание читателей.

Скорость работы сообщает, насколько быстро команда доставляет результат во времени. Этот показатель позволяет оценить объем выполненных работ и выяснить, продвигается ли команда по пути повышения производительности труда. Но Кенни предостерегает, что использование скорости работы как количественного показателя производительности труда может пагубно сказаться на коммерческих результатах, и помогает нам лучше понять причины.

В последнее время широкое распространение нашел термин *технический долг*, которым принято обозначать практически все, что неверно в коде. Кенни

раскрывает истинное значение этого термина и помогает нам лучше понять, почему так важны все эти, на первый взгляд, технические подробности. В частности, мне понравилось, как он умело поясняет, что если поставить команду разработчиков в цейтнот, то это неизбежно нарушит планы получить качественную продукцию в срок.

Как и все остальные методики гибкой разработки, Scrum опирается на диагностический подход с быстрой реакцией. Кенни вкратце описывает свой опыт употребления перфокарт, что напомнило мне первый опыт компьютерной обработки данных, который я приобрел задолго до того, как Кенни столкнулся со своей первой перфокартой.

Когда я учился в колледже, мне посчастливилось работать стажером в штаб-квартире Стратегического авиационного командования в Омахе. В те дни все компьютерная обработка данных выполнялась на перфокартах. Мои перфокарты направлялись на несколько этажей вниз в подвал данного учреждения, где они прогонялись на компьютере, применявшемся на случай войны, если бы таковая настала. И мне удавалось совершать один или даже два таких прогона в день.

Пройдя необходимую проверку на секретность, я спускался вниз в машинный зал ближе к полуночи. Ублажив сержанта Уиттокера позволить мне прогнать свои программы, я садился за консоль вычислительной машины, главная задача которой состояла в том, чтобы начать ядерное нападение. Остальное было просто, поскольку красная кнопка находилась в другом помещении.

Набирая свои программы на вычислительной машине вручную, я справлялся со своим заданием в десять раз быстрее, чем если бы мне пришлось ждать ввода с переданных мною вниз перфокарт и получения листингов обратно. Реакция наступала намного быстрее, я учился программировать скорее, а мои программы раньше становились работоспособными.

Именно в этом и состоит вся суть Scrum. Вместо того чтобы ждать месяцами или даже годами, чтобы выяснить, чем же занимаются программисты, Scrum позволяет сделать это в течение двух недель. Если у владельца продукта Scrum имеется действительно хорошая команда разработчиков, то уже через несколько дней он увидит, как начнут воплощаться в нечто реальное конкретные функциональные средства!

Именно этому и посвящена книга Кенни. Если вы только начинаете осваивать Scrum, прочитайте эту книгу от корки до корки и держите ее под рукой. А если у вас уже имеется опыт работы по методике Scrum, просмотрите эту книгу и все равно держите ее под рукой.

Если окажется, что вы размышляете над происходящим в вашей команде или же над разными приемами, которые вам хотелось бы опробовать, возьмите эту книгу и просмотрите ее. Вероятнее всего, вы найдете в ней что-нибудь ценное для себя.

Рон Джеффриз

Предисловие

В этой книге рассматриваются основы Scrum, т.е. то, что нужно знать для успешной разработки по методике Scrum передовых программных продуктов и доставки современных услуг.

Что составляет основы Scrum

Методика Scrum опирается на небольшой ряд основных *ценностей*, *принципов* и *норм надлежащей практики*, совместно называемых *инфраструктурой Scrum*. Организации, в которых применяется Scrum, должны полностью охватывать инфраструктуру Scrum, возможно, не во всей организации сразу, но хотя бы в первоначальных командах, где применяется Scrum. Но полный охват Scrum совсем не означает, что организации должны реализовывать Scrum по некоторому единому для всех шаблону. Напротив, это означает, что организации должны всегда придерживаться инфраструктуры Scrum, выбирая определенное сочетание подходов к реализации в них методики Scrum.

Эта книга сочетает в себе ценности, принципы и нормы надлежащей практики Scrum с рядом испытанных подходов, которые согласуются с инфраструктурой Scrum, но не предписываются ею. Одни из этих подходов могут подойти, а другие — не подойти для конкретной ситуации. Но любой подход требует исследования и приспосабливания к особым обстоятельствам.

Предпосылки к написанию этой книги

Меня как наставника и инструктора по методике Scrum гибкой разработки часто спрашивали о справочном пособии по Scrum, где можно было бы найти полное описание инфраструктуры Scrum и изложение наиболее распространенных подходов к применению Scrum. Мне не удалось найти ни одной книги, где все эти вопросы рассматривались бы достаточно глубоко, чтобы быть полезными для тех, кто ныне практикует методику Scrum в своей профессиональной деятельности, и поэтому я обычно рекомендовал собрание книг, в которых инфраструктура Scrum обсуждается, но неполно и не на современном уровне. Одни из этих книг посвящены гибкой разработке, и хотя они высоко ценятся, тем не

менее, они не сосредоточены только на методике Scrum. А в других книгах обсуждаются лишь отдельные аспекты Scrum или конкретные подходы, но сама инфраструктура Scrum — не полностью и не глубоко. И вся эта литература рассчитана на тех, кому требуется лишь единственный, отдельный первоисточник, в котором излагаются основы Scrum!

Изобретатели методики Scrum, Джефф Сазерленд (Jeff Sutherland) и Кен Швабер (Ken Schwaber), опубликовали в 2011 году документ "The Scrum Guide" (Руководство по Scrum). Авторы описывают этот краткий документ объемом около 15 страниц как "обстоятельный свод правил Scrum и документацию собственно на Scrum". Они приравнивают свой документ к правилам игры в шахматы, где "описывается, как перемещать фигуры, менять их позиции, что означает выигрыш и т.д.". Несмотря на всю пользу этого документа как обзора и свода правил Scrum, он не предназначен в качестве исчерпывающего источника для получения знаний об основах Scrum. Если продолжить аналогию авторов с игрой в шахматы, то документ "The Scrum Guide" на 15 страницах для новой Scrum-команды можно сравнить с краткой инструкцией по шахматам, прочитав которую вряд ли можно сразу же добиться успехов в этой игре. Этот документ просто нельзя считать самостоятельным ресурсом.

Данная книга представляет собой попытку восполнить пробел в отсутствии единого источника для получения знаний об основах Scrum. В ней подробно обсуждаются принципы, ценности и нормы надлежащей практики Scrum, которые в большинстве случаев согласуются с мнением авторов документа "The Scrum Guide" и других авторитетных специалистов по гибкой разработке. А там, где методика Scrum рассматривается под другим углом зрения, в книге указывается специально и поясняются соответствующие причины. В этой книге описываются подходы, согласующиеся с инфраструктурой Scrum и успешно применявшиеся на практике мной и теми командами разработчиков, которых я обучал. Эта книга не предназначена заменить собой другие книги, где дается углубленное по вертикали изложение отдельного практического приема или подхода к Scrum. Такие книги служат лишь дополнением и расширением данной книги, которую следует рассматривать скорее как отправную точку на пути к применению методики Scrum с целью доставить удовольствие заказчикам.

Кому адресована книга

Многим тысячам людей, которые прошли мои курсы по работе в Scrum-команде, аттестации Scrum-мастера и владельца продукта Scrum, а также многим командам, которые я обучал, эта книга напомнит, а возможно, и прояснит уже обсуждавшиеся вопросы. Для еще большего числа людей, с которыми мне не посчастливилось работать, эта книга станет первым введением в методику Scrum

гибкой разработки или возможностью посмотреть на Scrum под другим углом зрения, а может быть, и усовершенствоваться в работе по данной методике.

Эта книга не написана для одной конкретной роли, т.е. для владельцев продукта, Scrum-мастеров ил членов команды разработчиков. Напротив, она предназначена для того, чтобы дать всем, кто вовлечен в Scrum-процесс, будь то члены Scrum-команды или те, с кем они взаимодействуют в организации, общее представление о Scrum на основании базового ряда понятий с ясно определенным словарем терминов для их обсуждения. На этом общем основании я надеюсь ваша, читатель, организация окажется в лучшем положении, чтобы применять методику Scrum для достижения коммерческой ценности разрабатываемой продукции.

В моем представлении каждый член Scrum-команды должен обзавестись экземпляром данной книги, держа ее под рукой, чтобы открыть на нужной главе во время работы. Я также считаю, что руководители всех уровней в организации должны прочитать эту книгу, чтобы понять причины, по которым Scrum может стать эффективным подходом к управлению проектными работами, а также те виды организационных перемен, которые могут потребоваться для успешной реализации Scrum. Немало полезных сведений в этой книге имеется и для тех сотрудников организаций, где применяется или предполагается применять другой подход к гибкой разработке, кроме Scrum. Эти сведения полезны тем, что они имеют отношение к принятому в организации подходу к гибкой разработке.

Организация книги

Эта книга начинается с краткого введения в Scrum (глава 1), а завершается обсуждением путей, по которым предстоит далее пойти читателю (глава 23). Остальные главы книги разделены на четыре части.

- **Часть I. Основные понятия (главы 2-8).** Инфраструктура Scrum, принципы гибкой разработки, спринты, требования и пользовательские истории, задел продукта, оценивание и скорость работы, а также технический долг.
- **Часть II. Роли (главы 9-13).** Владелец продукта, Scrum-мастер, команда разработчиков, структуры Scrum-команды и руководители.
- **Часть III. Планирование (главы 14–18).** Принципы Scrum-планирования, многоуровневое планирование, планирование портфеля заказов, выработка общего замысла или планирование продукта, а также планирование выпуска.
- **Часть IV. Проведение спринтов (главы 19–22).** Планирование, выполнение, подведение итогов и ретроспектива спринта.

Как пользоваться этой книгой

Как и следовало ожидать, эта книга написана с тем, чтобы большинство купивших ее прочитали ее от корки до корки. Если вы только начинаете осваивать Scrum, вам придется поступить именно так, поскольку материал последующих глав данной книги основывается на материале предыдущих глав. А если вам требуется полный обзор инфраструктуры Scrum, т.е. богато иллюстрированный вводный курс в Scrum, прочитайте главу 2.

Те, кто уже знаком со Scrum, могут пользоваться этой книгой как справочным пособием по данной методике. Так, если вас интересуют ретроспективы спринтов, перейдите непосредственно к главе 22. А если вы хотите разобраться в особенностях задела продукта, перейдите к главе 6. Тем не менее всем читателям, даже знакомым со Scrum, настоятельно рекомендуется полностью прочитать главу 3. Изложенные в ней принципы положены в основу инфраструктуры Scrum и материала остальной части книги. Это не просто сформулированные еще раз ценности и принципы из "Манифеста гибкой разработки" [Agile Manifesto, Beck et al. 2001], которые являются общими для многих других рукописных описаний Scrum.

Язык визуальных образов

Я горд тем, что эта книга включает в себя новый язык визуальных образов для описания Scrum. Этот язык состоит из словаря образов, специально предназначенных для представления основных ролей, артефактов и видов деятельности в Scrum. Он служит эффективным средством для сообщения понятий, улучшая общее представление о Scrum. Если вас интересует получение и применение цветных иллюстраций, составленных на этом новом языке визуальных образов, поскольку книга напечатана в черно-белом варианте, обращайтесь за справкой на веб-сайт по адресу `www.innolution.com`. На этом веб-сайте можно найти много других полезных ресурсов и дискуссий на темы, связанные с данной книгой.

Итак, приступаем

Какую бы роль вы ни выполняли и в какой бы ситуации ни оказались, вы выбрали эту книгу по определенной причине. Уделите теперь немного времени ознакомлению с методикой Scrum. На последующих страницах книги вы обнаружите описание эффективной инфраструктуры, которую вы можете приспособить под свои нужды, чтобы значительно усовершенствовать процесс выпуска продукции и доставки услуг к общему удовольствию заказчиков.

Благодарности

Эта книга не увидела бы свет без посильного вклада многих людей, в том числе тысяч посетителей курсов и учебных занятий по гибкой разработке. Упоминая здесь лишь некоторых из этих людей, я рискую забыть остальных. Поэтому, если я не упомяну кого-нибудь из них, они все равно должны знать, что все наши дискуссии и обмен сообщениями по электронной почте оказали мне неоценимую помощь и определенно повлияли на качество данной книги!

Особую благодарность мне хотелось бы выразить следующим людям: Майку Кону (Mike Cohn), Ребекке Трегер (Rebecca Traeger) и Джеффу Шайху (Jeff Schaich). Без особого участия каждого из них эта книга была бы бледной тенью того, чем она стала теперь.

Майк Кон стал моим приятелем и коллегой еще в 2000 году, когда мы вместе работали в компании Genomica. Он любезно включил мою книгу в свою серию *Mike Cohn Signature Series* (Серия книг с подписью Майка Кона). Присоединившись к обществу Майка и других престижных авторов книг из этой серии, "Я выгляжу хорошо в компании, которую я поддерживаю", — как сказали бы мои родители. Майк был тем собеседником, с которым я мог всегда обсудить свои замыслы и стратегии работы над книгой. Он всегда находил время в своем напряженном рабочем графике для просмотра каждой главы книги и давал свои содержательные отзывы. Работа с Майком все эти годы была для меня бесценным опытом, и я надеюсь, что она продолжится и в будущем.

Ребекка Трегер была моим личным редактором этой книги. Мы работали вместе еще в 2007 году, когда я руководил организацией Scrum Alliance. В то время Ребекка выполняла функции редактора веб-сайта Scrum Alliance, и я посчитал, что эта и последующая работа сделала ее ведущим в отрасли редактором материалов по гибкой разработке. Едва приступив к работе над книгой, я связался с Ребеккой и попросил ее поработать со мной снова. К моему счастью, она согласилась. Первым читателем каждой главы книги была Ребекка. Иногда ее отзывы вызывали у меня стыд, поскольку она часто улучшала то, что я пытался выразить в письменном виде, и благодаря этому текст получался более понятным и удобочитаемым. Если вам понравится стиль изложения материала книги, будьте уверены, что это заслуга, прежде всего, Ребекки. А если не понравится, значит, я проявил глупость не последовав ее рекомендациям.

Джефф Шайх — исключительный художник-оформитель. Он работал над столькими художественными проектами, что я даже не упомню всех. Как только у меня сложился замысел данной книги, мне захотелось создать словарь образов методики Scrum гибкой разработки, чтобы воспользоваться им для своих учебных презентаций и более чем 200 иллюстраций к книге. Очевидно, что для этой цели мне был нужен опытный художник-оформитель. И Джефф согласился взять на себя решение этой нелегкой задачи. Иногда мне казалось, что эта книга похожа на два разных проекта: написание текста и создание художественных замыслов. Честно говоря, я даже не знаю, на что именно у меня ушло больше времени. Но я уверен, что без художественного оформления, выполненного Джеффом, качество этой книги пострадало бы безмерно.

Для меня большая честь, что вступительные слова к этой книге были написаны Майком Коном и Роном Джеффризом (Ron Jeffries) — двумя выдающимися личностями в области гибкой разработки! Каждый из них по-своему поставил эту книгу в нужный контекст, пригласив к обсуждению основ Scrum. Кроме того, Майк перестал питаться в ресторане быстрого питания "Burger King", а Рон, слава Богу, не нажал красную кнопку ядерного нападения!

Мне бы хотелось также поблагодарить многих людей, выкроивших время в своем напряженном рабочем графике для просмотра глав книги и отправки своих отзывов. Прежде всего, мне хотелось бы упомянуть следующих людей, просмотревших рукопись и приславших свои отзывы: Джо Балистриери (Joe Balistrieri), Йоханнеса Бродвалла (Johannes Brodwall), Лейну Котран (Leyna Cotran), Мартина Девоса (Martine Devos), Скотта Дункана (Scott Duncan), Илана Гольдштейна (Ilan Goldstein), Джона Хебли (John Hebley), Гейра Хедемарка (Geir Hedemark), Джеймса Ковача (James Kovacs), Лаури Маккиннона (Lauri Mackinnon), Роберта Максимчука (Robert Maksimchuk), а также Кевина Турески (Kevin Tureski).

Мне бы хотелось выразить свою признательность и другим рецензентам, приславшим свои отзывы на отдельные главы книги, в том числе: Лиссе Эдкинс (Lyssa Adkins), Джону Бауэру (John Bauer), Самиру Брендру (Sameer Bendre), Сьюзен Брискоу (Susan Briscoe), Павелу Бродзински (Pawel Brodzinski), Роуэну Баннингу (Rowan Bunning), Джошу Чаппеллу (Josh Chappell), Лизе Криспин (Lisa Crispin), Уэрду Канингхэму (Ward Cunningham), Корнелиусу Энгельбрехту (Cornelius Engelbrecht), Джулии Фрэйзер (Julia Frazier), Бриндузе Габуру (Brindusa Gabur), Каролин Гордон (Caroline Gordon), Дрю Джемило (Drew Jemilo), Майку Климкоски (Mike Klimkosky), Тому Лангерхорсту (Tom Langerhorst), Брайне Ларсену (Bjarne Larsen), Дину Леффингуэллу (Dean Leffingwell), Морису ле Рютте (Maurice le Rutte), Дэвиду Лузквиносу (David Luzquiños), Льве Яи (Lv Yi), Шею Маколэю (Shay McAulay), Армонду Мерабиану (Armond Mehrabian), Шериффу Мохамеду (Sheriff Mohamed), Куану Маллигану (Cuan Mulligan), Грегу Пизу (Greg Pease), Роману Пихлеру (Roman Pichler), Джакопо Ромеи (Jacopo Romei), Йенсу Шаудеру

(Jens Schauder), Биллу Шрёдеру (Bill Schroeder), Иву Стальжизу (Yves Stalgies), Бранко Стояковичу (Branko Stojaković), Говарду Саблетту (Howard Sublett), Джули Сильван (Julie Sylvain), Кевину Тамбаскио (Kevin Tambascio), Стивену Вольфраму (Stephen Wolfram), а также Майклу Уоллину (Michael Wollin).

Выражаю также благодарность сотрудникам издательства Pearson, оказавшимся надежными партнерами в данном проекте. Они снисходительно терпели мои задержки в работе над книгой и всегда оказывали мне всяческую поддержку. Особая признательность выражается Крису Гузиковски (Chris Guzikowski), контролировавшему весь процесс. Он опекал меня от первой встречи с представителями издательства Pearson в пабе г. Лексингтон, шт. Массачусетс до окончательного выпуска книги в свет. Благодарю также Оливию Базеджио (Olivia Basegio) — за грамотно организованное материальное снабжение, Джули Наил (Julie Nahil) — за фантастическую работу по общему надзору за проектом, Барбару Вуд (Barbara Wood) — за скрупулезную вычитку рукописи книг и Гэйла Кокера (Gail Cocker) — за отличную подготовку книги к печати.

Я признателен также своему помощнику Линдсею Калицки (Lindsey Kalicki), на плечи которого я переложил многие важные задачи, чтобы сосредоточиться на работе над книгой. Я счастлив работать с таким квалифицированным профессионалом.

Больше всего мне хотелось поблагодарить свою семью, жену Дженин и сынов Иону и Ашера, — за их решающую роль в данном проекте. Я был в огромном долгу перед ними на протяжении длительного периода работы над книгой, когда я не мог уделить должного внимания семье и мало проводил времени с ними.

Моя любимая вторая половина Дженин постоянно поддерживала меня в ходе работы над книгой. Жертвы, на которые она пошла ради того, чтобы я написал эту книгу, вдвойне превысили бы ее объем, если бы я попытался их перечислить. И если бы не она, я бы не смог написать книгу!

Любопытно, что спустя год, как мы поженились в 1993 году, я написал свою первую книгу *Succeeding with Objects*. И тогда я обещал Дженин, что больше не напишу ни одной книги. К счастью для меня, за 15 лет мои обещания изгладились из памяти, и в воспоминаниях напряженная работа над той книгой, нарушившая привычный семейный образ жизни, уже не казалась такой страшной. И каково же было мое удивление, когда моя жена убедила меня взяться за новую книгу! Она, конечно, не сказала, что на третью книгу я могу не рассчитывать, но я подозреваю, еще через 15 лет память снова ослабнет, и я смогу написать еще одну книгу!

Выражаю также глубокую признательность моим сынам Ионе и Ашеру за нежную поддержку. Они пожертвовали временем, которое могли бы провести со своим отцом, ради того, чтобы он написал свою книгу. Они постоянно носились с интересными идеями и внесли свой посильный вклад в эту книгу. В частности,

они внести ряд дельных предложений по содержанию и оформлению книги, от чего она стала только лучше! Надеюсь, они научились терпению и усвоили, что даже такую трудную работу можно завершить, если упорно продвигаться к намеченной цели, не отступая.

И наконец, мне хотелось бы воздать должное своей матери Джойс Рубин за всю оказанную мне любовь и моральную поддержку. Без ее влияния эта книга никогда бы не увидела свет. К сожалению, она не дожила до этого момента. Ее уход в январе 2012 года оставил в моей жизни и жизнях моих родных пустоту, которую вряд ли удастся восполнить. Она была особой личностью для тех, кто с ней соприкасался. Мне недостает матери больше, чем я мог бы выразить словами.

Об авторе

Кенни Рубин тренирует и обучает методике Scrum и гибкой разработке, помогая сотрудникам компаний эффективно и экономично разрабатывать программные продукты. Являясь аттестованным инструктором, Кенни обучил более 18 тыс. человек гибкой разработке по методике Scrum, в среде Smalltalk, а также управлению объектно-ориентированными проектами и переходами между разными стадиями процесса разработки. Он тренировал специалистов из более 200 компаний, от начинающих до входящих в десятку самых крупных.

Кенни был первым руководителем всемирно известной некоммерческой организации Scrum Alliance, специализирующейся на успешном внедрении Scrum. Помимо данной книги, Кенни является соавтором книги *Succeeding with Objects: Decision Frameworks for Project Management*, вышедшей в 1995 году. Он получил степень бакалавра наук в области информатики и вычислительной техники, окончив Технологический институт в шт. Джорджия, а также степень магистра наук в области вычислительной техники, окончив Стенфордский университет.

Истоки квалификации Кенни коренятся в объектно-ориентированной технологии. Он начинал как разработчик в среде Smalltalk, приняв участие в проекте, финансировавшемся НАСА в 1985 году. В рамках этого проекта он разработал на LISP первую экспертную систему по принципу доски объявлений. В 1988 году ему посчастливилось поступить на работу в компанию ParcPlace Systems, основанную как дочернюю компанией Xerox PARC для выпуска объектно-ориентированной технологии из исследовательских лабораторий в свет. Консультируя многие организации по разработке в среде Smalltalk с конца 1980-х годов и в течение 1990-х годов, Кенни рано взял на вооружение практические приемы гибкой разработки. Впервые он применил методику Scrum в 2000 году для разработки биоинформационного программного обеспечения.

В течение своей карьеры Кенни выполнял самые разные роли, в том числе успешно справлялся с обязанностями владельца продукта, Scrum-мастера и члена команды разработчиков. Кроме того, он выполнял многочисленные роли руководителей: исполнительного директора, главного управляющего, технического директора, вице-президента по управлению продукцией, вице-президента по профессиональным услугам. Он также заведовал отделом по разработке и выпуску пяти семейств коммерческих программных продуктов с совокупным

годовым доходом более 200 млн долларов США. Кроме того, он принимал участие в становлении венчурного инвестиционного фонда с капиталом более 150 млн долларов США и помогал двум компаниям выйти на американскую фондовую биржу NASDAQ.

Многогранная квалификация дает Кенни возможность ясно понимать и объяснять особенности методики Scrum, а также последствия ее внедрения с разных точек зрения: от команды разработчиков до высшего руководства.

От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо, либо просто посетить наш веб-сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг.

Наши электронные адреса:

E-mail: `info@dialektika.com`

WWW: `http://www.dialektika.com`

Глава 1

Введение

21 июня 2000 года я поступил на должность исполнительного вице-президента в биоинформационную компанию Genomica (Боулдер, шт. Колорадо). Я запомнил эту дату потому, что в час ночи того дня родился мой сын Ашер.

Его рождение стало хорошим началом того дня. Ашер родился в прогнозированный день (в Соединенных Штатах такое происходит в 5% всех случаев). Таким образом, мы (я и моя жена Дженин) завершили свой девятимесячный "проект" в срок. Более того, Ашер получил очень высокую оценку по шкале Апгар, которая свидетельствовала о том, что мы произвели на свет здорового ребенка, добившись качественного результата! Самый крупный участник нашего проекта, наш старший сын Иона, испытывал приятный трепет от того, что у него появился младший брат. Своевременное завершение проекта с высоким качеством и всеобщее удовольствие участников свидетельствовало о том, что это был действительно удачный день!

Вздремнув немного, я проверил свою электронную почту и обнаружил, что президент компании Genomica прислал мне экстренное сообщение, попросив меня прибыть на совещание совета директоров в 8:00 того же дня. Неохотно покинув больницу, я отправился на совещание.

Когда я прибыл на совещание, мне сказали, что технический директор был уволен накануне, и теперь я должен нанять проектно-техническую команду из 90 человек. Меня это не удивило, поскольку в течение нескольких месяцев руководящий состав и совет директоров активно обсуждали неспособность компании Genomica доставлять ценную продукцию в срок с приемлемым качеством, и в центре этого обсуждения оказался технический директор.

Теперь в мои обязанности входило осуществлять надзор над мероприятиями по улучшению результатов в организации разработки продукции в нашей компании. Я помню, как меня поразила ирония судьбы, когда в один и тот же день успешно завершились роды и сразу же на меня были возложены новые обязанности.

Я был очень занят надзором за продажами и сбытом продукции, и поэтому мне сказали, что я могу сам нанять технического директора, подотчетного мне. На эту должность я выбрал Майка Кона, автора известных книг по гибкой разработке, и мы решили воспользоваться методикой Scrum.

Что такое Scrum

Scrum — это методика *гибкой* разработки передовых программных продуктов и доставки современных услуг. На рис. 1.1 в простом, обобщенном виде показан гибкий подход к разработке.

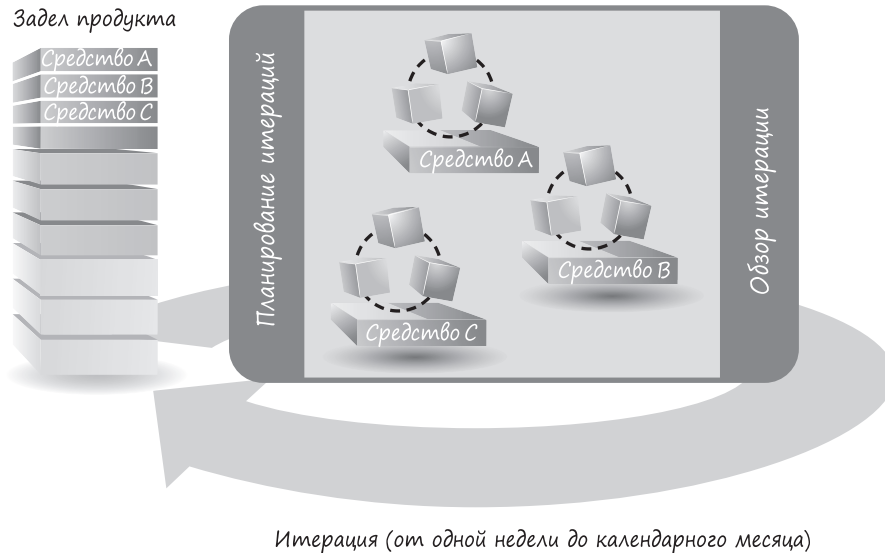


Рис. 1.1. Общее представление о гибкой разработке

При гибком подходе разработка начинается с создания *задела продукта* — приоритетного списка средств и других возможностей, которые требуются для разработки удачного программного продукта. Руководствуясь заделом продукта, можно всегда сосредоточиться прежде всего на работе над наиболее важными или высокоприоритетными элементами данного списка. Когда ресурсы (например, время) исчерпаются, любая незавершенная работа будет иметь более низкий приоритет, чем завершенная.

Сама работа выполняется в течение коротких, ограниченных по времени рабочих циклов, которые называются *итерациями* и обычно длятся от одной недели до календарного месяца. В течение каждой итерации самоорганизующаяся, *межфункциональная команда разработчиков* выполняет всю работу (разработку, сборку и тестирование), требующуюся для получения завершенных, работоспособных функциональных средств, которые могут быть переданы в эксплуатацию.

Как правило, объем работ в заделе продукта оказывается намного больше, чем команда может выполнить в течение одной краткосрочной итерации. Поэтому в начале каждой итерации команда планирует, какую именно часть задела продукта следует реализовать в первую очередь в течение предстоящей итерации. В качестве примера на рис. 1.1 команда согласилась с тем, что она может создать функциональные средства A, B и C.

В конце итерации команда оценивает завершенные функциональные средства вместе с другими участниками проекта, чтобы получить их отзывы в виде ответной реакции. Исходя из отзывов, владелец продукта и команда могут

изменить то, над чем они планируют работать дальше, и каким образом команда планирует выполнить работу. Так, если участники проекта находят функциональное средство завершенным и приходят к выводу, что другое функциональное средство, которое раньше не предполагалось, должно быть включено в разрабатываемый программный продукт, то владелец продукта может просто создать новый элемент, представляющий данное средство, введя его в задел продукта в таком порядке, чтобы работать над ним в будущей итерации.

В конце каждой итерации команда должна получить потенциально готовый к поставке продукт (или прирост продукта), который может быть выпущен, если это уместно. Если же выпуск после каждой итерации неуместен, то ряд функциональных средств из нескольких итераций может быть выпущен вместе. По завершении каждой итерации весь процесс начинается с самого начала, т.е. с планирования следующей итерации.

Истоки Scrum

Богатая история методики Scrum прослеживается в статье "The New New Product Development Game" (Новейшая игра в разработку продукции), опубликованной в *Harvard Business Review* в 1986 году [Takeuchi, Hirotaka, and Ikujiro Nonaka. 1986]. В этой статье описывается, как в компаниях Honda, Canon и Fuji-Xerox удалось добиться первоклассных результатов с помощью масштабируемого, ориентированного на команды подхода к *единовременной разработке продукции*. В этой главе подчеркивалось также особое значение уполномоченных, самоорганизующихся команд и выделялась особая роль руководства в процессе разработки.

Эта статья оказала большое влияние, связав воедино понятия, которые положили начало тому, что ныне называется Scrum. Термин *Scrum* не сокращение, он заимствован из спортивной игры регби и обозначает схватку вокруг мяча, т.е. способ возобновить игру после непреднамеренного нарушения правил или выхода мяча из игры. Даже если вы и не являетесь большим поклонником регби, то, вероятнее всего, видели схватку вокруг мяча, когда форварды обеих команд борются за мяч, сцепившись руками и наклонив головы.

Авторы упомянутой выше статьи, Такеучи (Takeuchi) и Нонака (Nonaka), употребили схватку вокруг мяча в регби и другие спортивные термины как метафоры для описания разработки продукции следующим образом:

> *"Подход в виде эстафетной гонки к разработке продукции может вступить в противоречие с целями максимальной скорости и гибкости разработки. Вместо этого целостный или "регбийный" подход, когда команда пытается преодолеть расстояние вместе, передавая мяч вперед и назад, может послужить для удовлетворения требований современной конкуренции".*

В 1993 году Джефф Сазерленд (Jeff Sutherland) и его команда разработчиков из компании Easel Corporation изобрела Scrum-процесс для разработки программного обеспечения, соединив понятия из упомянутой выше статьи с понятиями объектно-ориентированной разработки, управления эмпирическими процессами, итеративной и инкрементной разработки, исследованиями в области производительности труда в процессе создания программного обеспечения и сложных адаптивных систем. А в 1995 году Кен Швабер (Ken Schwaber) опубликовал первую статью по Scrum на конференции OOPSLA [Schwaber, Ken. 1995]. С тех пор Швабер и Сазерленд совместно и раздельно издали целый ряд публикаций, посвященных Scrum, включая книги *Agile Software Development with Scrum* [Schwaber, Ken, and Mike Beedle. 2001], *Agile Project Management with Scrum* [Schwaber, Ken. 2004], а также документ "The Scrum Guide" (Руководство по Scrum) [Schwaber, Ken, and Jeff Sutherland. 2011].

Несмотря на то что методика Scrum чаще всего применяется для разработки программных продуктов, базовые ценности и принципы Scrum могут быть использованы и применяются в разработке других видов продукции или в организации *хода* различных видов работ. Мне, например, пришлось сотрудничать с организациями, где методика Scrum успешно применялась для организации и управления работами, связанными с проектированием оборудования, разработкой программ рекламных кампаний, сбыта и продажи продукции.

Причины для применения Scrum

Так почему же такой гибкий подход к разработке, как Scrum, оказался удачным выбором для компании Genomica? Прежде всего, было ясно, что предыдущий подход, применявшийся в компании Genomica для разработки, оказался просто неработоспособным. Это было плохо; но хорошо, что почти все с этим согласились.

Компания Genomica действовала в комплексной области, где больше было неизвестного, чем известного. Мы выпускали продукцию, которая не производилась прежде. Наша деятельность была сосредоточена на переднем крае постоянно развивавшейся области исследовательских информационных платформ, которыми ученые-исследователи могли бы пользоваться как вспомогательными средствами для открытия новых, неизвестных до сих пор молекул. Нам нужен был способ разработки, который позволил бы оперативно внедрять новые идеи и подходы и быстро учиться принимать жизнеспособные решения, отвергая нежизнеспособные. У нас был стратегический партнер в лице компании, которой нам нужно было показывать результаты своих трудов через каждые несколько недель и получать ее ответную реакцию, поскольку нашу продукцию требовалось интегрировать с базовой серией секвенсоров ДНК, выпускавшихся этой

компанией. Такое требование быстрого исследования и ответной реакции не вполне согласовывалось с подробным, предварительным планированием, которое мы практиковали.

Нам нужно было также избежать предварительного проектирования архитектуры. При предыдущей попытке создать следующее поколение базовой продукции компании Genomica был потрачен почти год только на проектирование архитектуры, чтобы получить единую крупную биоинформационную платформу. Когда первое реальное исследовательское приложение было внедрено в эту архитектуру и мы наконец-то проверили верность проектных решений, принятых многие месяцы назад, то на перемещение путем табуляции от одного поля ввода данных к другому потребовалось 42 секунды. Если учесть нетерпение типичного пользователя, то нетрудно себе представить реакцию молекулярного биолога со степенью доктора наук, которому придется ждать целых 42 секунды! Это была катастрофа. Нам нужен был другой, более уравновешенный подход к разработке, который сочетал бы в себе элементы предварительного проектирования со здоровой порцией постепенного проектирования точно в срок.

Нам нужно было также, чтобы наши команды разработчиков были межфункциональными. Исторически сложилось так, что компания работала так, как и большинство других организаций. Разработанная продукция передавалась командам тестирования только после полного ее завершения. Но теперь нам требовалось, чтобы все члены команды синхронизировали свою работу чаще — желательно ежедневно. В прошлом ошибки накапливались, поскольку важные вопросы обсуждались слишком поздно в процессе разработки. Специалисты из разных областей общались редко. По этим и другим причинам мы решили, что методика Scrum вполне подойдет для внедрения в компании Genomica.

Результаты, достигнутые в компании Genomica

Когда мы остановили свой выбор на методике Scrum, она была малоизвестна. Первая книга по Scrum появилась только в следующем году [Schwaber, Ken, and Mike Beedle. 2001]. Тем не менее мы собрали всю необходимую информацию и постарались сделать все, что было в наших силах, а результаты оказались намного лучшими, чем прежде (табл. 1.1).

Таблица 1.1. Результаты, достигнутые в компании Genomica благодаря методике Scrum

Мера	Водопадная методика	Методика Scrum
Затраты труда	**10x**	**1x**
Скорость работы	**1x**	**7x**
Удовлетворенность заказчика	Плохо	Отлично

Разработка по методике Scrum потребовала в десять раз меньше затрат труда, рассчитанных в человеко-месяцах, чем по предыдущей плановой водопадной методике разработки сравнимого количества функциональных средств программного продукта. Не менее важно, что разработка по методике Scrum продвигалась в семь раз быстрее, чем по водопадной методике. Это означает, что за единицу времени по методике Scrum было разработано в семь раз больше ценных средств, чем по водопадной методике. Еще более впечатляющим оказалось то, что мы предоставили программное обеспечение нашему партнеру как раз в те сроки, когда он собирался запустить свою новую аппаратную платформу. Это дало нам возможность упрочить долгосрочное сотрудничество, что значительно повысило стоимость акций компании Genomica.

Чем Scrum может помочь вам

Предыдущий опыт компании Genomica до внедрения Scrum, когда никому ненужные функциональные средства разрабатывались с запаздыванием и плохим качеством, совсем не уникален. Как и многие другие организации, компания Genomica существовала благодаря тому, чтобы была ничем не хуже своих конкурентов. Аналогичные проблемы я наблюдал, когда только начинал заниматься разработкой коммерческого программного обеспечения в средине 1980-х годов. И спустя почти тридцать лет ситуация во многих компаниях совсем не улучшилась.

Если ныне собрать вместе деловых людей и разработчиков и спросить их: “Довольны ли вы результатами ваших трудов по разработке программного обеспечения?” или “Считаете ли вы, что доставляете хорошую потребительскую ценность в срок, экономично и качественно?”, то что они ответят?

Чаще всего люди, с которыми мне приходилось встречаться на учебных курсах по всему миру, дают на оба эти вопроса отрицательный ответ, который далее сопровождается целым рядом жалоб вроде “Вероятность провала проекта неприемлемо высока”, “Доставка запаздывает”, “Окупаемость часто не оправдывает ожиданий”, “Качество программного обеспечения низкое”, “Производительность труда далека от идеала”, “Никто не учитывает последствия”, “Трудовая дисциплина низкая”, “Текучесть кадров слишком высокая”. Затем следует сдавленный от смущения смешок и лукавое замечание “Должен быть лучший способ”.

Несмотря на всю эту досаду, большинство людей, по-видимому, смиряются с тем, что неудовлетворенность составляет часть той реальности, которая присуща разработке программного обеспечения. Но ведь так не должно быть. Тем организациям, где прилежно применяется методика Scrum, ведома совершенно другая реальность (рис. 1.2).

Рис. 1.2. Преимущества Scrum

Такие организации постоянно удовлетворяют своих заказчиков, предоставляя им именно то, что они хотят получить, а не только функциональные средства, которые требовались поначалу, когда они еще не знали, что же им действительно нужно. Кроме того, такие организации сразу обнаруживают улучшение окупаемости инвестиций благодаря доставке программных продуктов более мелкими и частыми *выпусками*. А благодаря безжалостному выявлению нарушений в нормальной деятельности и *расточительства* таким организациям удается сократить расходы.

Акцент Scrum на выпуске работоспособных, интегрированных, проверенных, коммерчески ценных функциональных средств в течение каждой итерации приводит к тому, что результаты достигаются быстро. Методика Scrum вполне подходит и для того, чтобы помочь организациям добиться успеха в сложном мире, где они должны быстро адаптироваться, исходя из взаимосвязанных действий конкурентов, потребителей, пользователей, регуляторных органов и прочих заинтересованных сторон. Кроме того, что Scrum доставляет б*о*льшую удовлетворенность участникам проекта, удовлетворенными оказываются не только заказчики, но и исполнители! Они испытывают удовлетворенность от частого и содержательного сотрудничества, что способствует улучшению межличностных отношений и повышению взаимного доверия среди членов команды разработчиков.

Поймите меня правильно. Несмотря на то что методика Scrum оказывается отличным решением во многих ситуациях, она годится не на все случаи жизни. Так, смыслообразующая инфраструктура *Cynefin* [Snowden, David J., and Mary

E. Boone. 2007] помогает лучше понять ситуацию, в которой приходится действовать, и выбрать подход, наиболее пригодный для данной ситуации. В этой инфраструктуре определяются и сравниваются разные области: *простая*, *сложная*, *хаотичная*, *комплексная* и *беспорядочная* (в последней области вы оказываетесь в том случае, если не знаете, в какой из четырех других находитесь; рис. 1.3). Воспользуемся инфраструктурой Cynefin, чтобы обсудить ситуации, для которых Scrum подходит или не годится.

Рис. 1.3. Инфраструктура Cynefin

Прежде всего, очень важно понять, что многие особенности разработки и сопровождения программного обеспечения не вписываются только в одну инфраструктуру Cynefin. Разработка программного обеспечения — непростое предприятие, изобилующее многими перекрывающимися аспектами и видами деятельности, которые могут быть отнесены к разным областям [Pelrine, Joseph. 2011]. Несмотря на то что б*о*льшую часть разработки программного обеспечения можно отнести к сложной или комплексной области, было бы наивно прямо

заявить, что вся разработка программного обеспечения принадлежит только комплексной области, особенно если определить ее таким образом, чтобы она включала в себя целый спектр работ: от разработки передовых, совершенно новых продуктов до непрерывающегося сопровождения, эксплуатации и поддержки ранее выпущенных продуктов.

Комплексная область

Когда приходится решать комплексные проблемы, ситуация может оказаться в большей степени непредсказуемой, чем предсказуемой. Если и имеется правильный ответ, то мы узнаем его только задним числом. Это область *непредвиденного*. Мы должны исследовать, чтобы изучить проблему, а затем *обследовать и приспособить*, опираясь на изученное. Для работы в комплексных областях требуются творческие и передовые подходы. Рутинные, шаблонные решения здесь просто непригодны. Нам нужно создать надежную среду для экспериментирования, чтобы выявить важную информацию. В этой среде очень важно взаимодействие и общение на высоком уровне. Именно к этой категории относится разработка передовых, совершенно новых продуктов, как, впрочем, и усовершенствование уже существующих продуктов передовыми, совершенно новыми функциональными средствами.

Методика Scrum особенно пригодна для работы в комплексной области. В подобных случаях решающее значение имеет наша способность *опробовать* (исследовать), *осмысливать* (обследовать) и *реагировать* (адаптироваться).

Сложная область

Сложные проблемы относятся к области норм надлежащей практики, где господствуют опытные специалисты. Здесь может быть несколько правильных ответов, но для их выявления требуется грамотная диагностика. И хотя методика Scrum позволяет справиться с таким проблемами, она может оказаться не самым лучшим решением. Например, работы по оптимизации производительности системы, требующие настройки ее параметров, лучше поручить специалистам по сборке, предоставив им возможность оценить ситуацию, рассмотреть разные варианты и основывать свою реакцию на нормах надлежащей практики. К этой категории относится б*о*льшая часть сопровождения современного программного обеспечения, т.е. технической поддержки или устранения дефектов в программных продуктах. Именно для этого оказываются особенно пригодными многие тактические, количественные подходы вроде Six Sigma, хотя они применимы и в простых областях.

Простая область

Когда приходится решать простые проблемы, каждому ясна причина и следствие. Нередко правильный ответ очевиден и бесспорен. Это область узаконенных норм передовой практики, где решения хорошо известны. Оценив факты по конкретной ситуации, мы можем найти подходящее готовое решение. Методику Scrum можно, конечно, применять для решения простых проблем, но она может оказаться далеко не самым эффективным для этого средством. Поэтому в данном случае лучше воспользоваться процессом с вполне определенными, повторяющимися стадиями, на которых проблема будет точно решена. Так, если требуется неоднократно воспроизвести один и тот же программный продукт, то для этой цели вполне определенный поточный производственный процесс подойдет лучше, чем Scrum. А развертывание одного и того же готового коммерческого продукта в 100-й среде заказчика лучше всего выполнить, повторяя вполне определенные и опробованные стадии установки и конфигурирования продукта.

Хаотическая область

Хаотические проблемы требуют быстрой реакции. Мы находимся в критической ситуации и должны действовать немедленно, чтобы предотвратить дальнейший ущерб и хотя бы восстановить порядок. Допустим, что университет опубликовал статью, в которой заявляется, что наш программный продукт содержит дефектный алгоритм, дающий ошибочные результаты. Наши клиенты вложили немало средств в производство, опираясь на результаты, получаемые с помощью нашего программного продукта, и поэтому они готовятся подать иск против нас за нанесенный крупный ущерб. Наш ведущий разработчик алгоритмов находится в отпуске на острове Борнео и будет отсутствовать больше двух недель. В данном случае методика Scrum оказывается не самым лучшим решением. Ведь нас не интересует приоритетность работ из задела и определение тех работ, которые следует выполнить в течение следующей итерации. Нам нужно действовать немедленно и решительно, чтобы устранить неполадку. Для решения хаотических проблем кому-то нужно взять ситуацию под контроль и действовать решительно.

Беспорядочная область

Если вы не знаете, в какой из других четырех областей оказались, значит, вы находитесь в беспорядочной области. Это опасное место, поскольку вы не знаете, как осмыслить данную ситуацию. В подобных случаях люди стремятся интерпретировать сложившееся положение и действовать в соответствии с их личными

предпочтениями относительно предпринимаемого действия. Многим специалистам по разработке программного обеспечения знакомы поэтапные подходы, вполне пригодные в простых областях, и поэтому они отдают им личное предпочтение. Но, к сожалению, они малопригодны для большинства видов разработки программного обеспечения, как поясняется в главе 3. Оказавшись в беспорядочной области, попробуйте разделить ее на составляющие части и перенести каждую из них в контекст одной из четырех других областей. Это не попытка применить методику Scrum в беспорядочной области, а выйти из данной области.

Работа с прерываниями

Методика Scrum не вполне пригодна для работы с частыми прерываниями. Допустим, что вы руководите организацией по поддержке клиентов и вам требуется применить Scrum для организации и управления деятельностью по такой поддержке. Ваш задел продукта заполняется на постоянной основе по мере получения запросов на поддержку по телефону или электронной почте. Вы не знаете заранее, в какой момент задел продукта расширится чрезмерно, а его содержимое и порядок могут меняться часто (возможно, через каждый час или несколько минут).

В подобной ситуации вы не сможете надежно спланировать итерации на неделю или больший период времени, поскольку не знаете, какую работу предстоит выполнить спустя это время. И даже если вам кажется, что вы знаете, какая именно предстоит работа, вполне вероятно, что поступит высокоприоритетный запрос на поддержку, который отодвинет в сторону любые далеко идущие планы.

В средах, где работа выполняется с прерываниями, лучше принять другой гибкий подход, называемый *Kanban*. Это не решение в виде автономного процесса, а подход, который накладывается на существующий процесс. В частности, методика Kanban рекомендует делать следующее.

- Наглядно представить прохождение работ через систему (например, шаги, которые поддерживающая организация должна предпринять, чтобы разрешить запрос на поддержку).
- Ограничить незавершенные работы на каждом этапе, чтобы не выполнять больше работы, чем можно сделать.
- Измерить и оптимизировать прохождение работ через систему, чтобы вносить непрерывные усовершенствования.

К преимуществам Kanban относятся области сопровождения и поддержки программного обеспечения. Некоторые из тех, кто применяет методику Kanban на практике, подчеркивают, что благодаря акценту на устранении

перегруженности (путем согласования незавершенных работ с производственными возможностями) и сокращения непостоянства в прохождении работ при одновременном стимулировании эволюционного подхода к изменениям методика Kanban оказывается пригодной для применения и в комплексных областях.

Обе методики, Scrum и Kanban, обеспечивают гибкий подход к разработке, и у каждой из них имеются свои сильные и слабые стороны, которые следует принимать во внимание, осмысливая ту область, где приходится работать. В некоторых организациях можно применять обе методики, Scrum и Kanban, чтобы удовлетворить разные, но сосуществующие системные потребности. Например, Scrum можно применять для разработки новых продуктов, а Kanban — для сопровождения и поддержки с прерываниями.

Заключение

Методика Scrum не является панацеей или палочкой-выручалочкой. Но она поможет охватить изменения, которые сопровождают все комплексные *проектные работы*. Эта методика может работать и действительно работала в Genomica и во многих других компаниях, где ее было решено внедрить в разработку программного обеспечения, поскольку она лучше соответствовала их обстоятельствам.

Несмотря на всю простоту инфраструктуры Scrum, было бы ошибкой считать, что применение Scrum обойдется легко и безболезненно. Методика Scrum не дает готовых ответов на вопросы, возникающие в производственном процессе. Вместо этого она побуждает команды разработчиков самим ставить насущные вопросы и находить на них ответы. Методика Scrum не дает готовых рецептов для лечения организационных недугов, но в то же время позволяет выявить расточительство и нарушения в нормальной деятельности, препятствующие организациям раскрыть свой истинный потенциал.

Для многих организаций реализация нововведений по методике Scrum может оказаться болезненной. Но если организации преодолеют первоначальные неудобства и будут работать над решением проблем, которые вскрывает Scrum, то они смогут добиться немалого прогресса как в процессе разработки и выпуска программного обеспечения, так и в удовлетворенности исполнителей и заказчиков достигнутыми результатами.

Остальная часть данной книги посвящена обсуждению основных аспектов Scrum. Сначала будет описана вся инфраструктура Scrum, включая роли, виды деятельности, артефакты и правила. Если вы будете правильно применять методику Scrum в подходящих условиях, то, возможно, добьетесь такого же успешного результата в доставке ценностей, как и моя жена в тот счастливый для нас день в 2000 году.

Часть I

Основные понятия

Глава 2

Инфраструктура Scrum

В этой главе дается краткий обзор инфраструктуры Scrum, где основное внимание уделяется нормам ее практики, в том числе ролям, видам деятельности и артефактам. А в последующих главах каждая из этих норм практики рассматривается более подробно, включая углубленный анализ принципов, положенных в основу обсуждаемых норм практики.

Краткий обзор

Scrum не является стандартизированным процессом, где нужно методично придерживаться последовательных этапов, гарантирующих производство в срок и по смете высококачественной продукции, удовлетворяющей требованиям заказчиков. Напротив, Scrum — это *инфраструктура* для организации работ и управления ими. Инфраструктура Scrum опирается на ряд ценностей, принципов и норм практики, образующих то основание, которое организация дополняет своей собственной реализацией соответствующих норм инженерной практики и особыми подходами к осуществлению норм практики Scrum. В итоге должна получиться версия, характерная для особенностей данной организации.

Чтобы стало яснее понятие инфраструктуры Scrum, ее можно представить в виде фундамента, на котором возводятся стены здания. Ценности, принципы нормы практики Scrum в этом случае служат основными конструктивными элементами здания. Пренебречь или основательно изменить ценность, принцип или норму практики Scrum нельзя, чтобы не разрушить все здание в целом. Но в то же время внутри здания можно вносить конструктивные изменения до тех пор, пока не будет получен процесс, удовлетворяющий конкретным производственным потребностям.

Инфраструктура Scrum удивительно проста и человечна, поскольку она опирается на такие ценности, как честность, открытость, смелость, уважение, собранность, доверие, расширение прав и полномочий, сотрудничество. Более подробно принципы Scrum рассматриваются в главе 3, а в последующих главах поясняется, каким образом конкретные нормы практики и подходы коренятся в этих принципах и ценностях.

Сами нормы практики Scrum воплощаются в конкретных ролях, видах деятельности, артефактах и связанных с ними правилах (рис. 2.1). В остальной части этой главы основное внимание уделяется нормам практики Scrum.

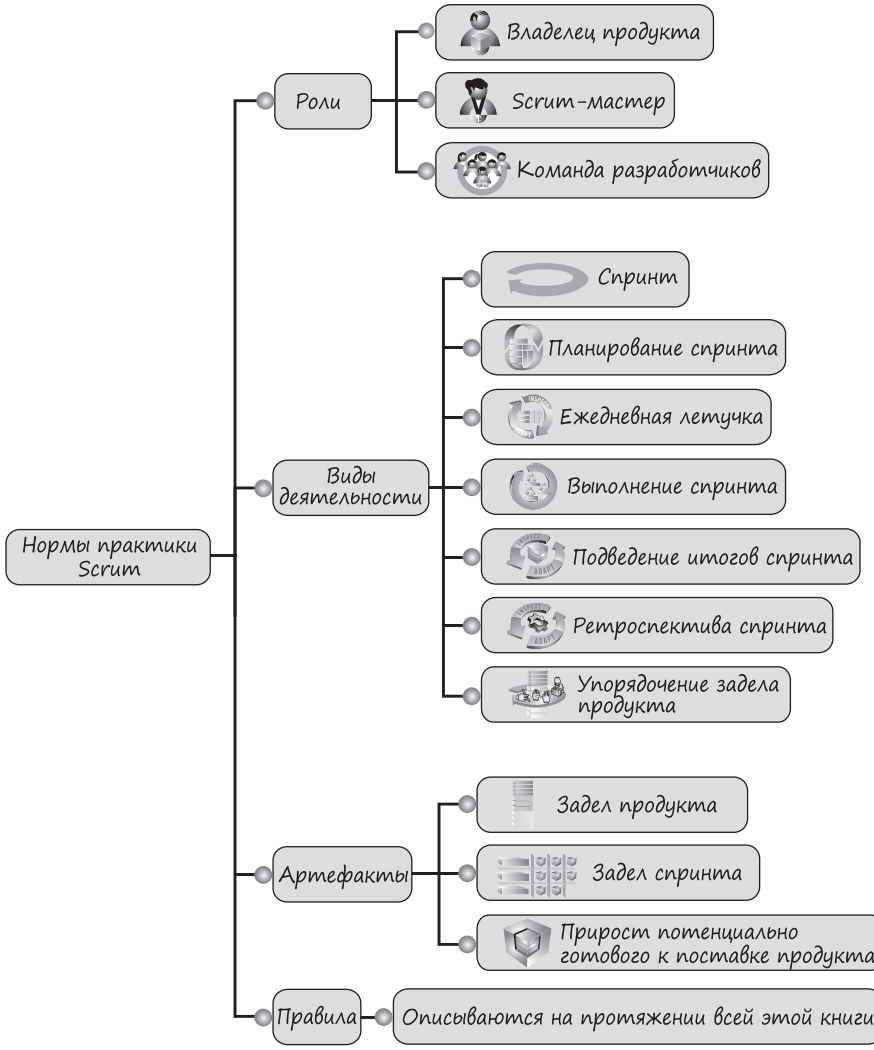


Рис. 2.1. Нормы практики Scrum

Роли в Scrum

Проектные работы по методике Scrum выполняются одной или несколькими *Scrum-командами*, в каждой из которых распределяют следующие роли: *владельца продукта*, *Scrum-мастера* и *команды разработчиков* (рис. 2.2). Для

применения Scrum могут быть назначены и другие роли, но инфраструктура Scrum требует только три упомянутых выше роли.

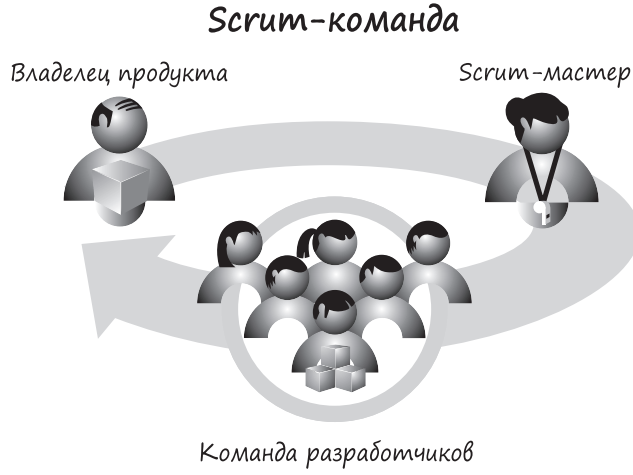


Рис. 2.2. Роли в Scrum

Владелец продукта отвечает за то, что должно быть разработано и в каком именно порядке. Scrum-мастер отвечает за то, чтобы направить команду на создание собственного процесса и следование ему, опираясь на более широкую инфраструктуру Scrum. А команда разработчиков отвечает за выпуск того, что от нее требует владелец продукта.

Если вы руководитель, то не особенно беспокойтесь о том, что роль руководителя отсутствует на рис. 2.2. Руководителям по-прежнему принадлежат важные роли в тех организациях, где применяется Scrum (подробнее об этом речь пойдет в главе 13). В инфраструктуре Scrum определяются лишь те роли, которые характерны для Scrum, а не все роли, которые могут и должны существовать в той организации, где применяется Scrum.

Владелец продукта

Владелец продукта наделен полномочиями центрального руководства разрабатываемым продуктом. Это единственное полномочное лицо, отвечающее за принятие решений относительно тех функциональных средств и возможностей, которые требуется разработать, а также за порядок их создания. Владелец продукта поддерживает и доносит до всех участников проекта ясное представление о том, чего Scrum-команда пытается достичь. Следовательно, владелец продукта несет полную ответственность за общий успех разрабатываемого или сопровождаемого проектного решения.

Независимо от того, предназначается ли разрабатываемый продукт для внешнего или внутреннего потребления, владелец продукта обязан обеспечить выполнение как можно более ценных работ, которые могут носить и сугубо технический характер. Чтобы команда быстро создала продукт, который требуется владельцу продукта, последний должен быть доступен для своевременного ответа на возникающие вопросы. Более подробно роль владельца продукта рассматривается в главе 9.

Scrum-мастер

Scrum-мастер помогает всем участникам проекта лучше понять и принять ценности, принципы и нормы практики Scrum. Он действует как наставник, обеспечивая руководство процессом и помогая Scrum-команде и остальной организации выработать собственный высокопроизводительный подход к Scrum. В то же время Scrum-мастер помогает организации пройти трудный процесс болезненных изменений в управлении, которые возможны во время принятия методики Scrum.

В качестве координатора Scrum-мастер помогает команде решать насущные вопросы и вносить коррективы в то, как она применяет Scrum. Он также отвечает за защиту команды от внешнего вмешательства и берет на себя ведущую роль в устранении *препятствий*, отрицательно сказывающихся на производительности труда в команде, когда отдельные ее члены не в состоянии устранить эти препятствия самостоятельно. Scrum-мастер наделяется полномочиями осуществлять контроль над командой, и поэтому его роль отличается от традиционной роли руководителя проекта или разработки. Scrum-мастер должен действовать как лидер, а не руководитель. Более подробно роли функционального руководителя и руководителя проекта рассматриваются в главе 13, а роль Scrum-мастера — в главе 10.

Команда разработчиков

В традиционных подходах к разработке программного обеспечения описываются различные виды должностных обязанностей вроде архитектора, программиста, тестировщика, администратора базы данных, разработчика пользовательского интерфейса и т.д. Роль команды разработчиков в Scrum определяется как разнотипное, межфункциональное собрание перечисленных выше специалистов, отвечающих за разработку, построение и тестирование требующегося программного продукта.

Команда разработчиков самоорганизуется для определения наилучшего способа достичь цели, поставленной владельцем продукта. Как правило, команда

разработчиков состоит из пяти–девяти человек. Ее члены должны обладать всей квалификацией, которая требуется для производства высококачественного, работоспособного программного обеспечения. Разумеется, методику Scrum можно применять и для проектных работ, требующих намного более крупных команд. Тем не менее вместо Scrum-команды, состоящей, скажем, из 35 человек, было бы лучше создать четыре и больше команды из девяти или менее человек. Подробнее о роли команды разработки речь пойдет в главе 11, а о координации работы нескольких команд — в главе 12.

Виды деятельности и артефакты Scrum

На рис. 2.3 демонстрируется большинство видов деятельности и артефактов Scrum, а также их взаимное соответствие.

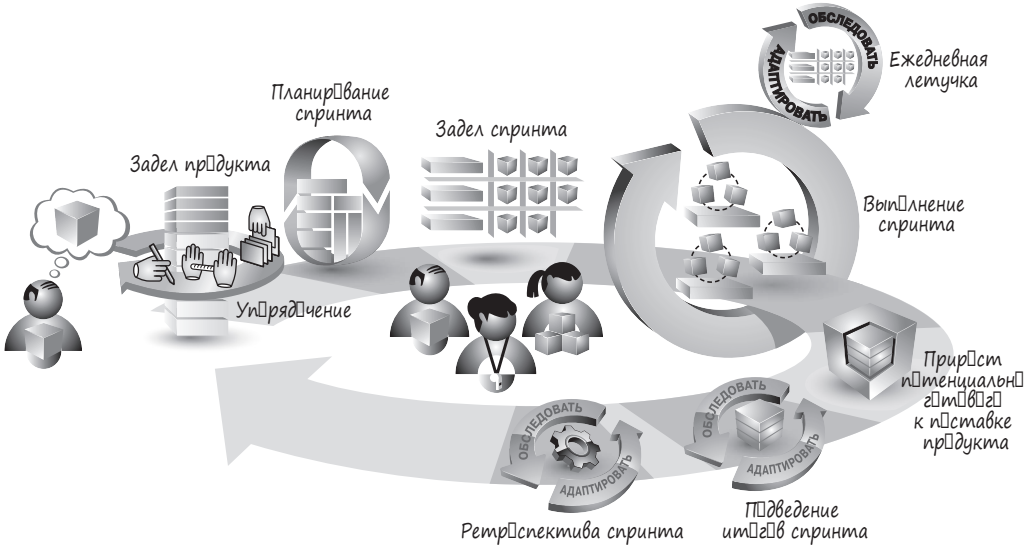


Рис. 2.3. Инфраструктура Scrum

Рассмотрим блок-схему, приведенную на рис. 2.3, начиная с левого края рисунка и далее по часовой, образующей петлю стрелке, обозначающей спринт. У владельца продукта должно быть ясное представление о том, что он хотел бы создать, как обозначает большой куб. Этот куб может быть большим, и поэтому через вид деятельности, называемый *упорядочением*, он разделяется на ряд функциональных средств, составленных в приоритетный список, называемый *заделом продукта*.

Спринт начинается с планирования и охватывает проектные работы, выполняемые в течение спринта и называемые *выполнением спринта*, а завершается он *подведением итогов* и *ретроспективой*. На рис. 2.3 спринт обозначен крупной, образующей петлю стрелкой, господствующей в центральной части рисунка. Элементов в заделе спринта, скорее всего, будет больше, чем команда разработчиков способна завершить в течение короткого спринта. Именно поэтому в начале каждого спринта команда разработчиков должна определить подмножество тех элементов из задела спринта, которые она способна завершить в течение данного спринта. Этот вид деятельности называется *планированием спринта* и соответственно обозначен справа от крупного куба, представляющего задел продукта на рис. 2.3.

В качестве краткого отступления следует заметить, что изменения, внесенные в документ "The Scrum Guide" [Schwaber, Ken, and Jeff Sutherland. 2011], вызвали полемику по поводу пригодности термина *прогноз* или *обязательство*, обозначающего результат планирования спринта. Сторонники термина *прогноз* считают, что он точнее отражает положение дел, когда, несмотря на стремление команды разработчиков наилучшим образом оценить объем работ, которые она способна выполнить, эта оценка может измениться по мере поступления дополнительной информации в течение спринта. Некоторые также считают, что в жертву обязательства, которое берет на себя команда, может быть принесено качество. А с другой стороны, "недообязательство" гарантирует выполнение взятого обязательства.

Я согласен с тем, что все команды разработчиков должны формировать прогноз (т.е. оценку) того, что они должны произвести в течение каждого спринта. Но многие команды разработчиков только выиграют от использования прогноза для взятия обязательства. Ведь обязательства поддерживают взаимное доверие владельца продукта и команды разработчиков, а также членов самой команды. Кроме того, обязательства обеспечивают обоснованное краткосрочное планирование и принятие решений в организации. А если разработка ведется несколькими командами, то обязательства обеспечивают синхронизированное планирование, когда одна команда может принимать решения, исходя из того, что обязуется выполнить другая команда. Но в этой книге предпочтение отдается термину *обязательство*, а термин *прогноз* употребляется лишь в тех случаях, когда он точнее соответствует контексту.

Чтобы приобрести уверенность в том, что команда разработчиков взяла на себя обоснованное обязательство, ее члены создают во время планирования спринта второй задел, называемый *заделом спринта*. С помощью ряда тщательно сформулированных *задач* в заделе спринта описывается порядок планирования командой разработки, построения, интеграции и тестирования

подмножества функциональных средств, выбранного из задела продукции, в течение данного конкретного спринта.

Далее следует выполнение спринта, когда команда разработки решает задачи, которые требуется реализовать для реализации выбранных функциональных средств. Каждый день в ходе выполнения спринта члены команды помогают управлять ходом работ, проводя синхронизацию, обследование и адаптивное планирование, — все эти действия вместе называют *ежедневной летучкой*. В конце выполнения спринта команда должна произвести прирост потенциально готового к поставке продукта, дающий некоторое, хотя и не полное представление о нем владельца продукта.

Scrum-команда завершает спринт, выполняя два вида деятельности по обследованию и адаптации. Первый вид деятельности называется *подведением итогов* спринта и состоит в том, что участники проекта и Scrum-команда обследуют создаваемый продукт. Второй вид деятельности называется *ретроспективой спринта* и состоит в том, что Scrum-команда обследует Scrum-процесс, применяемый для создания продукта. Следствием обоих этих видов деятельности могут стать меры адаптации, которые должны быть направлены на задел продукта или включены как составная часть в процесс, выполняемый командой разработки.

На данном этапе цикл спринтов в Scrum повторяется, снова начинаясь с определения в команде разработки следующего наиболее важного ряда элементов из задела продукта, которые она способна завершить. По завершении соответствующего числа спринтов представление владельца о продукте должно быть реализовано и полученное решение может быть выпущено. Каждый из упомянутых выше видов деятельности и артефактов более подробно обсуждается в оставшейся части этой главы.

Задел продукта

Применяя Scrum, мы всегда выполняем сначала самую ценную работу. Владелец продукта, опираясь на вклад остальных членов Scrum-команды и участников проекта, в конечном итоге отвечает за определение и управление последовательностью работ, сообщая о ней в форме приоритетного (или упорядоченного) списка, называемого *заделом продукта* (рис. 2.4). При разработке нового продукта элементами задела продукта первоначально оказываются функциональные средства, удовлетворяющие представлению владельца о продукте. А при продолжающейся разработке уже имеющегося продукта его задел может также содержать новые средства, изменения в существующих средствах, дефекты, которые нужно устранить, технические усовершенствования и т.д.

Рис. 2.4. Задел продукта

Владелец продукта сотрудничает с внутренними и внешними участниками проекта, чтобы собрать и определить элементы задела продукта. Затем он обеспечивает расположение этих элементов в нужном порядке, используя такие факторы, как ценность, стоимость, знание и риск, чтобы более ценные элементы оказались в верхней части задела продукта, а менее ценные — в нижней. Задел продукта является постоянно развивающимся артефактом. Владелец продукта может вводить, удалять и пересматривать элементы задела продукта по мере изменения коммерческих условий или представлений Scrum-команды о разрастании масштабов продукта (вследствие ответной реакции на программное обеспечение, производимое в течение каждого спринта).

Общая деятельность по созданию и уточнению элементов задела продукта, их оценка и расстановка по приоритета называется *упорядочением* (рис. 2.5).

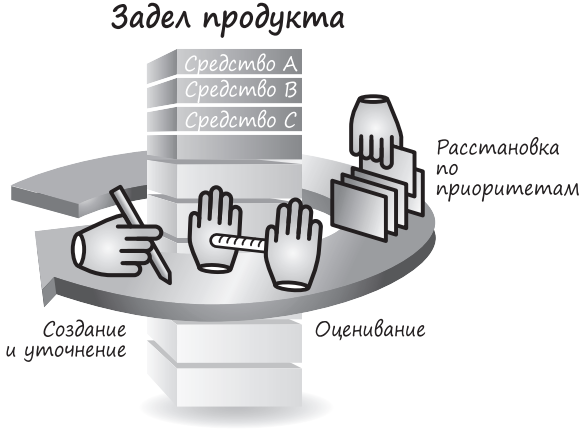


Рис. 2.5. Упорядочение задела продукта

В качестве еще одного краткого отступления следует заметить, что изменения, внесенные в документ “The Scrum Guide” [Schwaber, Ken, and Jeff Sutherland. 2011], вызвали полемику по поводу применения первоначального термина *приоритетный* и пришедшего ему на смену термина *упорядоченный*. В качестве аргумента выдвигалось пояснение, что расстановка по приоритетам — это всего лишь одна из форм упорядочения, а некоторые даже считали, что это не самая подходящая форма упорядочения. Но дело в том, что оптимальное расположение элементов в заделе продукта зависит от многих факторов, и полностью охватить это понятие одним словом невозможно. Несмотря на теоретические доводы в пользу термина *приоритетный* или *упорядоченный*, на практике большинство, в том числе и я, пользуется обоими терминами попеременно, обсуждая элементы в заделе продукта.

Прежде чем покончить с расстановкой по приоритетам, упорядочением или расположением элементов в заделе продукта, нам нужно выяснить размер каждого элемента в заделе продукта (рис. 2.6).

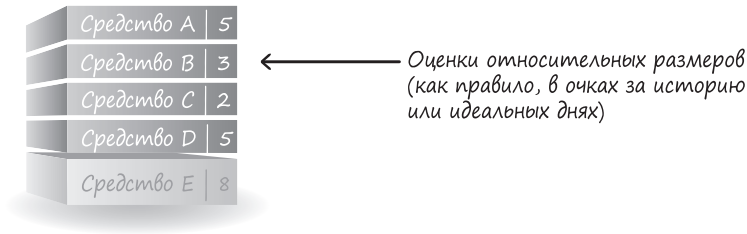


Рис. 2.6. Размеры элементов в заделе продукта

Размер элемента приравнивается к его стоимости, и поэтому владельцам продуктов нужно знать стоимость элемента, чтобы правильно определить его приоритетность. Методика Scrum не предписывает, какой именно мерой следует пользоваться для определения размеров элементов в заделе продукта. На практике многие команды пользуются такими мерами относительного размера, как *очки за историю* или *идеальные дни*. Мера относительного размера выражает общий размер элемента таким образом, чтобы принимать во внимание не абсолютное значение, а относительный размер элемента в сравнении с другими элементами. В качестве примера функциональное средство C на рис. 2.6 имеет относительный размер **2**, а функциональное средство E — относительный размер **8**. Из этого можно сделать вывод, что функциональное средство E в четыре раза крупнее, чем функциональное средство C. Более подробно эти меры обсуждаются в главе 7.

Спринты

Работа в Scrum выполняется в течение итераций или рабочих циклов, длящихся до одного календарного месяца и называемых *спринтами* (рис. 2.7).

Работа, завершаемая в каждом спринте, должна приводить к созданию какой-нибудь материальной ценности для заказчика или пользователя.

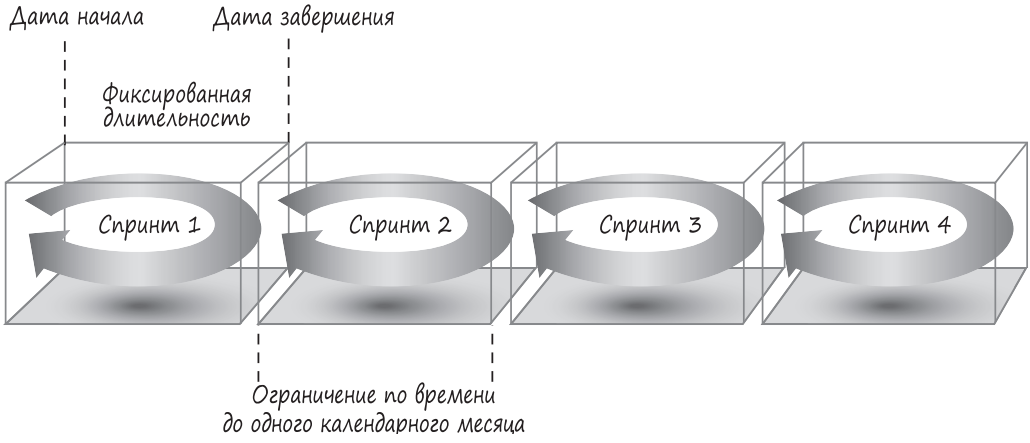


Рис. 2.7. Характеристики спринта

Спринты *ограничиваются по времени*, и поэтому они всегда имеют фиксированную дату начала и завершения и обычно одну и ту же длительность. Новый спринт начинается сразу же по завершении предыдущего спринта. Как правило, любые изменения в объеме работ или кадрах, влияющие на достижение поставленных целей, в течение спринта не допускаются. Но иногда коммерческие потребности не позволяют придерживаться этого правила. Более подробно спринты рассматриваются в главе 4.

Планирование спринта

Задел продукта может представлять работы в течение многих недель или месяцев, что намного больше, чем можно завершить в течение одного короткого спринта. Чтобы определить наиболее важные подмножества среди элементов задела продукта и создать их в следующем спринте, владелец продукта, команда разработчиков и Scrum-мастер выполняют *планирование спринта* (рис. 2.8).

Во время планирования спринта владелец продукта и команда разработчиков согласуют *цель спринта*, которая определяет, что, собственно, предполагается достичь в предстоящем спринте. Руководствуясь этой целью, команда разработчиков просматривает задел продукта и определяет высокоприоритетные элементы, которые она может теоретически завершить в предстоящем спринте, работая в *постоянном темпе*, т.е. таком темпе, который позволяет команде удобно работать в течение продолжительного периода времени.

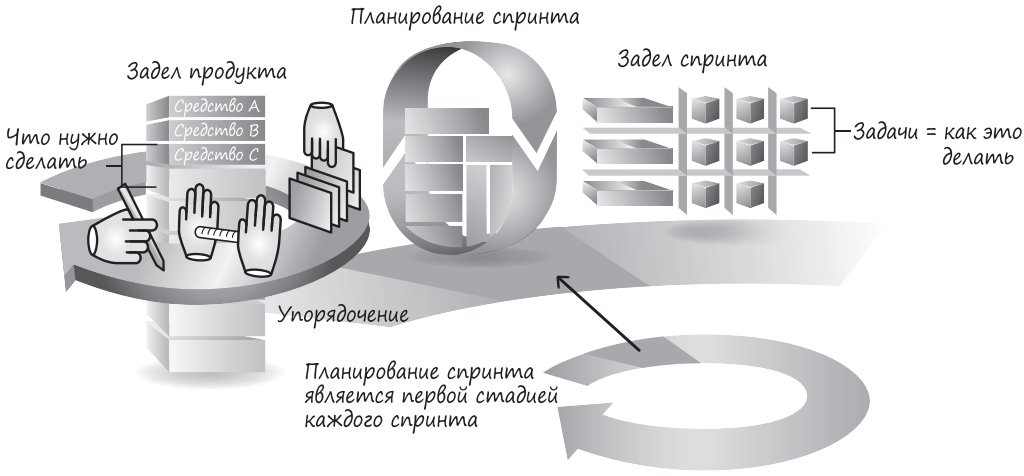


Рис. 2.8. Планирование спринта

Чтобы приобрести уверенность в своих силах, многие команды разработчиков разбивают каждое намеченное для реализации функциональное средство на ряд задач. Совокупность этих задач вместе со связанными с ними элементами из задела продукта образует второй задел, называемый *заделом спринта* (рис. 2.9).

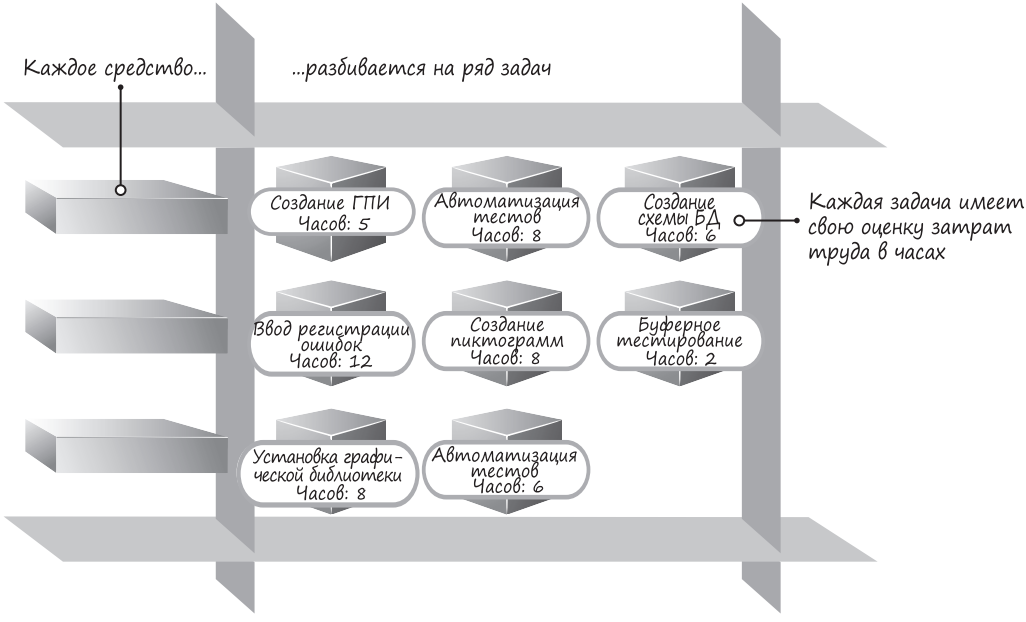


Рис. 2.9. Задел спринта

Затем команда разработчиков оценивает (обычно в часах) затраты труда, требующиеся на выполнение каждой задачи. Разбиение элементов из задела продукта на отдельные задачи является формой *своевременной* разработки и планирования точно в срок для получения готовых функциональных средств.

Большинство Scrum-команд, собирающихся выполнять спринты от двух недель до одного месяца, пытаются завершить планирование спринта в течение 4–8 часов. Для планирования однонедельного спринта требуется не больше двух часов, а возможно, и меньше. В течение этого периода времени могут быть опробованы разные подходы. Я лично чаще всего придерживаюсь следующего простого цикла: выбрать элемент из задела спринта (по возможности следующий наиболее важный элемент, определяемый владельцем продукта), разбить выбранный элемент на отдельные задачи и выяснить, насколько удачно он вписывается в рамки спринта (вместе с другими элементами, намеченными для завершения в том же самом спринте). Если он вписывает в спринт, а для выполнения работ еще остаются производственные возможности, то цикл повторяется до тех пор, пока у команды больше не останется возможностей для выполнения любой дополнительной работы.

В качестве альтернативного подхода владелец продукта и команда выбирают по очереди все намеченные элементы из задела продукта. Команда разработчиков сама разбивает элементы на отдельные задачи, чтобы убедиться, что она действительно может выпустить все элементы, выбранные из задела продукта. Каждый из этих подходов более подробно рассматривается в главе 19.

Выполнение спринта

Как только Scrum-команда завершит планирование спринта и согласует содержание следующего спринта, команда разработчиков под руководством Scrum-мастера приступает к выполнению работ над отдельными задачами, чтобы получить готовые функциональные средства (рис. 2.10). В данном контексте слово “готовые” означает, что существует высокая степень уверенности в том, что вся работа, необходимая для производства качественных функциональных средств, завершена.

Выполнение командой конкретных задач, безусловно, зависит от характера работ. Например, разрабатывается ли программное обеспечение и какое именно, проектируется ли оборудование или же это маркетинговая работа?

Никто не может указывать команде разработчиков, в каком именно порядке или как следует выполнять работы на уровне отдельных задач в заделе спринта. Вместо этого члены команды сами определяют объем работ на уровне отдельных задач, а затем самоорганизуются любым образом, каким они считают нужным, для достижения цели спринта. Более подробно выполнение спринта рассматривается в главе 20.

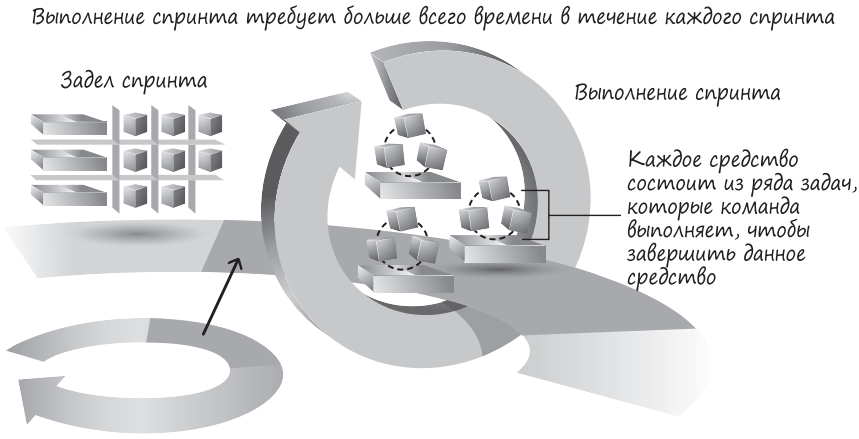


Рис. 2.10. Выполнение спринта

Ежедневные летучки

Каждый день спринта (в идеальном случае — в одно и то же время) члены команды разработчиков собираются на краткое совещание (не больше 15 минут), называемое *ежедневной летучкой* (рис. 2.11). Этот вид деятельности по обследованию и адаптации иногда еще называется *ежедневным стоянием*, поскольку все участники обычно стоят во время совещания, что способствует его краткости.

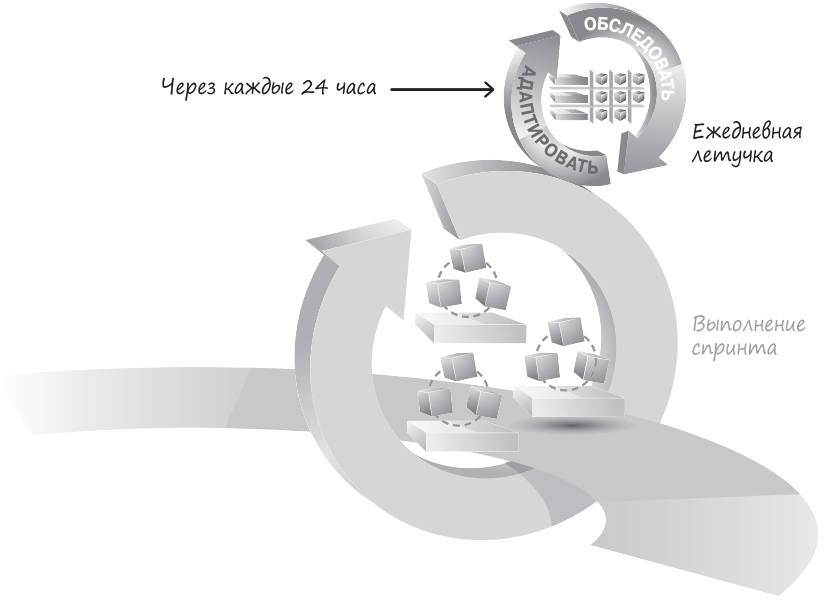


Рис. 2.11. Ежедневная летучка

Как правило, ежедневную летучку проводит Scrum-мастер, а каждый член команды по очереди отвечает на следующие три вопроса в пользу остальных ее членов.

- Что я сделал с момента последней ежедневной летучки?
- Что я собираюсь сделать к следующей ежедневной летучке?
- Какие препятствия и преграды мешают мне добиваться прогресса в моей работе?

Отвечая на эти вопросы, каждый ясно представляет себе общую картину того, что происходит, как они продвигаются к цели спринта, какие коррективы требуется внести в их планы на предстоящий рабочий день и какие вопросы нужно решить. Ежедневная летучка очень важна в том отношении, что она помогает команде разработчиков оперативно управлять ходом работ в течение спринта.

Ежедневная летучка не служит для разрешения затруднений в работе. Напротив, разработчики зачастую обсуждают возникшие проблемы после ежедневной летучки в небольших командах заинтересованных лиц. Кроме того, ежедневная летучка не является традиционной планеркой, и в особенности традиционным совещанием, которое обычно собирали руководители проектов, чтобы обновить состояние проекта. Тем не менее ежедневная летучка может оказаться полезной для ознакомления членов команды разработчиков с состоянием элементов из задела продукта. Ежедневная летучка является видом деятельности, который состоит, главным образом, в обследовании, синхронизации и адаптации планов на день, способствуя самоорганизации команды с целью делать свое дело лучше.

В методике Scrum раньше применялись термины *поросята* и *цыплята* для обозначения участников и просто наблюдателей ежедневной летучки, хотя ныне эти термины не в чести. Они позаимствованы из следующей старой шутки, имеющей разные варианты: “В завтраке с яичницей и ветчиной цыпленок только участвует, а поросенок втянут в него полностью”. Очевидно, что намерение употреблять эти термины в Scrum состояло в том, чтобы отличать тех, кто только участвует (цыплят), от тех, кто полностью вовлечен в достижение цели спринта (поросят). На ежедневной летучке поросята должны общаться, тогда как цыплята — только присутствовать как наблюдатели, если они вообще ее посещают.

На мой взгляд, поросятами лучше считать членов Scrum-команды, а всех остальных — цыплятами, хотя с этим не все согласны. Например, владелец продукта не обязан посещать ежедневные летучки, и поэтому некоторые относят его к категории цыплят. (Их логика такова: каким образом человек может быть “втянут”, если он не обязан посещать ежедневные летучки?) Мне это кажется неверным, поскольку трудно представить, каким образом владелец продукта как член Scrum-команды оказывается менее вовлеченным в достижение результатов

спринта, чем команда разработчиков. Метафора цыплят и поросят теряет свой смысл, если применять ее ко всей Scrum-команде в целом, а не только к команде разработчиков.

Результаты спринта

В методике Scrum результаты спринта обозначаются как *прирост потенциально готового к поставке продукта* (рис. 2.12). Это означает, что все, что бы ни было согласовано Scrum-командой для выполнения, действительно сделано в соответствии с заранее согласованным в ней критерием готовности. Этот критерий обозначает степень уверенности в том, что работа завершена с качественным результатом, потенциально готовым к поставке. Например, при разработке программного обеспечения минимальным критерием готовности должен быть выпуск части функциональных возможностей программного продукта, которая разработана, построена, интегрирована, протестирована и документирована. Столь решительный критерий готовности позволяет исполнителям решить в каждом спринте, требуется ли то, что было произведено, поставить (развернуть или выпустить) внутренним или внешним заказчикам.

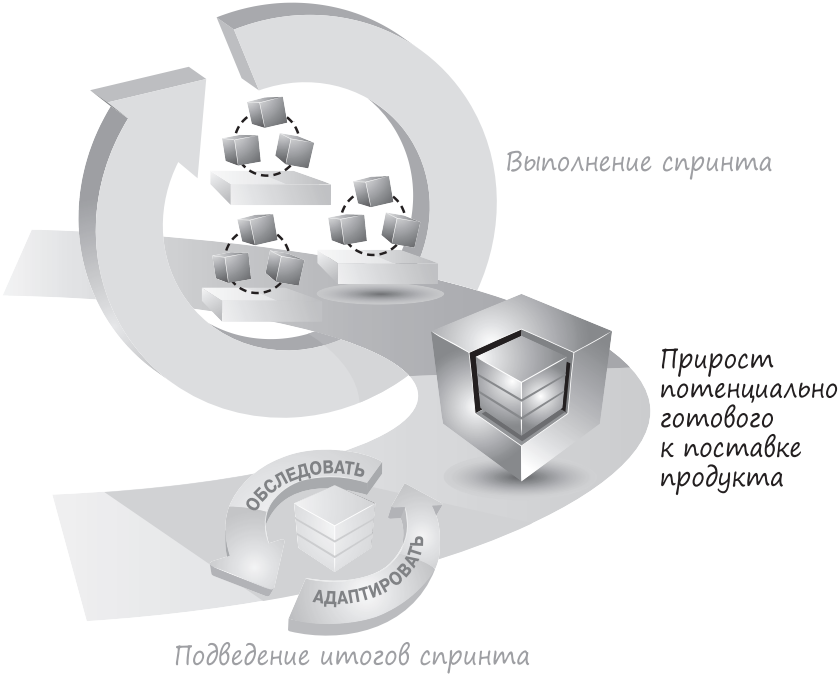


Рис. 2.12. Результаты спринта (прирост потенциально готового к поставке продукта)

Откровенно говоря, если продукт "потенциально готов к поставке", то это совсем не означает, что он фактически должен быть поставлен. Вопрос поставки является коммерческим решением, на которое зачастую оказывают влияние такие вопросы, как, например: "Достаточно ли у нас функциональных средств, чтобы охватить рабочий процесс заказчика или оправдать развертывание продукта на стороне заказчика?" или "Способны ли наши заказчики принять очередное изменение, если учесть, что мы предоставили им выпуск две недели назад?"

Потенциальную готовность к поставке лучше рассматривать как состояние уверенности в следующем: то, что было создано в течение спринта, действительно готово. Это означает, что фактически не осталось никакой незавершенной работы (например, важного тестирования, интеграции и т.д.), которую нужно завершить, прежде чем поставлять результаты спринта, если поставка преследует коммерческую цель.

На практике некоторые команды со временем могут изменять критерий готовности. Например, наличие на ранних стадиях разработки компьютерных игр функциональных средств, потенциально готовых к поставке, может быть экономически неоправданным или нежелательным, принимая во внимание исследовательский характер ранних стадий разработки компьютерных игр. В подобных случаях подходящий критерий готовности может быть частью функциональных средств программного продукта, которые достаточно функциональны и пригодны для формирования ответной реакции заказчика, позволяющей команде разработчиков решить, какие именно работы должны быть выполнены дальше и как их следует выполнить. Подробнее о критерии готовности речь пойдет в главе 4.

Подведение итогов спринта

В конце спринта остается выполнить еще два вида деятельности по обследованию и адаптации. Один из них называется *подведением итогов спринта* (рис. 2.13).

Цель подведения итогов спринта — обследование и адаптация создаваемого продукта. Решающим для этого вида деятельности является обсуждение, которое происходит среди участников, к которым относится Scrum-команда, участники проекта, попечители, заказчики и заинтересованные члены других команд. Это обсуждение сосредоточено на оценке только что завершенных функциональных средств в контексте общего объема проектных работ. Каждый участник обсуждения составляет ясное представление о происходящем и имеет возможность помочь направить последующую разработку в нужное русло, чтобы обеспечить принятие решения, наиболее подходящего с коммерческой точки зрения.

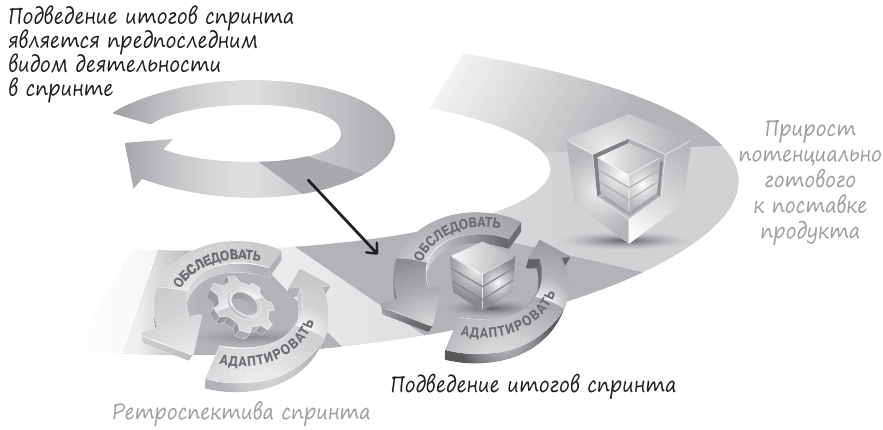


Рис. 2.13. Подведение итогов спринта

Успешное подведение итогов спринта приводит к появлению информационного потока в обоих направлениях. Те лица, которые не являются членами Scrum-команды, должны прийти к взаимопониманию относительно проектных работ, чтобы помочь направить их в нужное русло. В то же время члены Scrum-команды могут лучше понять коммерческую и маркетинговую стороны разрабатываемого продукта, получая частую ответную реакцию на его способность удовлетворить требования заказчиков или пользователей. Следовательно, подведение итогов спринта представляет собой запланированную заранее возможность обследовать и адаптировать разрабатываемый продукт. На практике те лица, которые не являются членами Scrum-команды, могут оценивать функциональные средства в пределах спринта и давать отзывы, чтобы помочь Scrum-команде эффективнее достичь цели спринта. Подробнее о подведении итогов спринта речь пойдет в главе 21.

Ретроспектива спринта

Второй вид деятельности по обследованию и адаптации в конце спринта называется *ретроспективой спринта* (рис. 2.14). Этот вид деятельности нередко происходит после подведения итогов спринта и до планирования следующего спринта.

Если подведение итогов спринта является удобным моментом для обследования и адаптации разрабатываемого продукта, то ретроспектива спринта дает возможность обследовать и адаптировать производственный процесс. В течение ретроспективы спринта команда разработчиков, Scrum-мастер и владелец продукта совместно обсуждают, что действует и что не действует по методике Scrum, а также связанные с этим нормы инженерной практики. Основное

внимание уделяется непрерывному усовершенствованию производственного процесса, чтобы помочь хорошей Scrum-команде стать отличной. В конце ретроспективы спринта Scrum-команда должна выявить и взять на себя обязательство предпринять ряд практических действий по усовершенствованию производственного процесса в следующем спринте. Подробнее о ретроспективе спринта речь пойдет в главе 22.

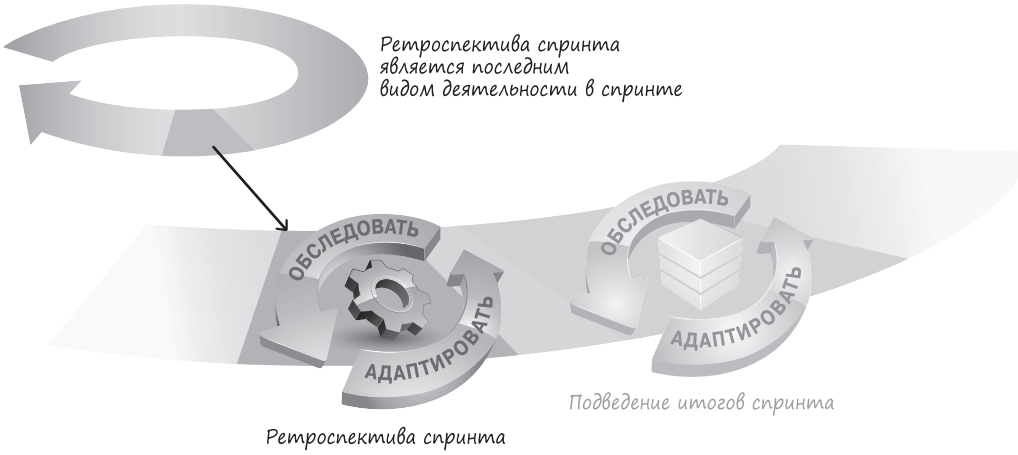


Рис. 2.14. Ретроспектива спринта

По завершении ретроспективы спринта весь рабочий цикл повторяется снова, начиная с совещания по планированию следующего спринта. На этом совещании определяется текущий наиболее ценный объем работ, на котором должна сосредоточиться команда.

Заключение

В этой главе представлены основные нормы практики Scrum с акцентом на последовательное описание ролей, видов деятельности и артефактов инфраструктуры Scrum. Имеются и другие нормы практики, в том числе планирование на более высоком уровне или отслеживание достигнутого прогресса, которые применяются во многих Scrum-командах. Эти нормы практики будут описаны в последующих главах. А в следующей главе будут представлены основные принципы, положенные в основу Scrum. Это поможет более углубленно исследовать инфраструктуру Scrum в последующих главах.

Глава 3
Принципы гибкой разработки

Прежде чем углубляться в механизм работы Scrum, имеет смысл разъяснить основополагающие принципы, приводящие в действие и наполняющие этот механизм. В этой главе описываются принципы гибкой разработки, положенные в основу методики Scrum. Они сравниваются с принципами традиционной, плановой, последовательной разработки программных продуктов. Благодаря этому подготавливается почва для понимания отличий методики Scrum от более традиционных форм разработки программных продуктов и подробного анализа норм практики Scrum, рассматриваемых в последующих главах.

Краткий обзор

На мой взгляд, основополагающие принципы Scrum было бы поучительно рассматривать, сравнивая их с представлениями о более традиционной, плановой, последовательной разработке программных продуктов. Это позволит лучше понять сходство и отличие методики Scrum от того, что уже известно и понятно читателям.

Цель сравнения принципов гибкой и традиционной разработки состоит не в том, чтобы показать, насколько плановая, последовательная разработка хуже гибкой разработки по методике Scrum. Ведь обе методики применяются в арсенале средств профессиональных разработчиков. Дело не в том, что методики или инструментальные средства разработки чем-то плохи, а в несвоевременном их применении. Как упоминалось вкратце в главе 1 при рассмотрении инфраструктуры Cynefin, методики Scrum и традиционной, плановой, последовательной разработки пригодны для решения разных видов задач.

Сравнивая обе методики, я пользуюсь чистым или "хрестоматийным" описанием методики плановой, последовательной разработки. Описывая традиционный подход к разработке с этой точки зрения, я считаю целесообразным провести различие между сравниваемыми методиками, чтобы нагляднее продемонстрировать принципы, положенные в основу гибкой разработки по методике Scrum.

Одна из чистых форм традиционной, плановой разработки нередко называется *водопадной* (рис. 3.1). Но это лишь один пример более обширного класса *плановых* процессов, называемых также *традиционными*, *последовательными*, *упреждающими*, *прогнозирующими* или *перспективными процессами разработки*.

Эти процессы называются плановыми потому, что в них предпринимается попытка спланировать и с упреждением определить все функциональные средства, которые требуются пользователю в конечном продукте, а также наметить наилучший способ их разработки. Основной замысел в данном случае состоит в следующем: чем качественнее планирование, тем лучше понимание, а следовательно, и исполнение. Плановые процессы нередко называются последовательными, поскольку те, кто применяет их на практике, последовательно и полностью выполняют анализ требований, а по его завершении — проектирование, программирование, сборку и, наконец, тестирование программного продукта.

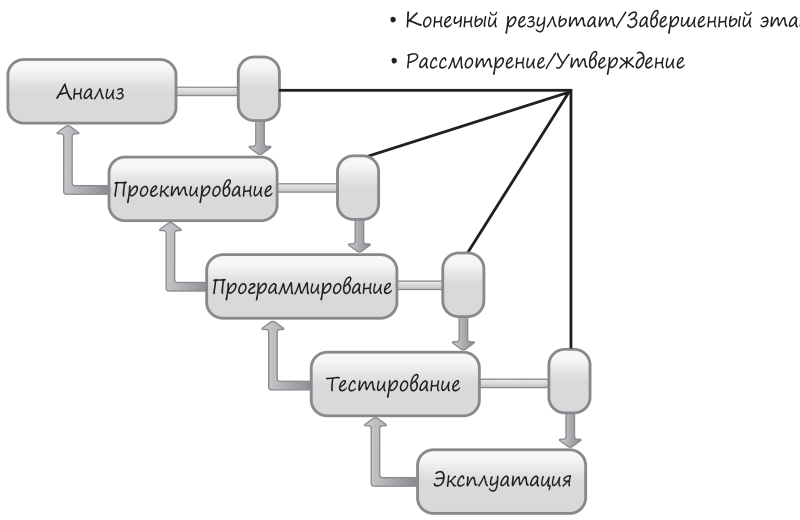


Рис. 3.1. Водопадный процесс

Плановая методика разработки пригодна в том случае, если она применяется для решения вполне определенных и предсказуемых задач, которые вряд ли будут подвержены каким-нибудь изменениям. Но дело в том, что большинство проектных работ непредсказуемы, особенно в начале. И хотя плановый процесс создает впечатление упорядоченности, отчетности и определенности, это впечатление может привести к ложному ощущению надежности. Ведь в конечном итоге разработка программного продукта редко идет по плану.

Для многих плановый, последовательный процесс просто состоит в том, чтобы осмыслить поставленную задачу, спроектировать, запрограммировать, протестировать и развернуть готовый программный продукт — и все это сделать по вполне определенному, заранее составленному плану. Для этого должна быть уверенность в том, что такой подход *должен* быть работоспособным. Если же плановый подход не работает, то преобладает следующее мнение: что-то было сделано не так. И даже если плановый процесс неизменно дает разочаровывающие результаты, то многие организации продолжают применять тот же самый

подход в полной уверенности, что если они будут работать лучше, то и результаты их трудов улучшатся. Но ведь все дело не в исполнении, а в плановых подходах, основанных на ряде представлений, которые не согласуются с неопределенностью, присущей большинству проектно-конструкторских работ.

С другой стороны, методика Scrum основывается на совсем других представлениях, которые вполне согласуются с задачами, которые настолько неопределенны, что с трудом поддаются прогнозированию. Принципы, описываемые в этой главе, взяты из целого ряда первоисточников, включая "Манифест гибкой разработки" (Agile Manifesto) [Beck et al. 2001], а также опираются на основы разработки программных продуктов [Reinertsen, Donald G. 2009b; Poppendieck, Mary, and Tom Poppendieck. 2003] и упоминавшийся ранее документ "The Scrum Guide" [Schwaber, Ken, and Jeff Sutherland. 2011]. Эти принципы разделяются на несколько категорий, как показано на рис. 3.2.

Начнем обсуждение с принципов, эффективно использующих изменчивость и неопределенность. Затем будут рассмотрены принципы, связанные с сочетанием заблаговременного прогнозирования и своевременной адаптации. Далее обсуждаются принципы, опирающиеся на обучение, а также принципы управления незавершенными работами. И в завершение будут рассмотрены принципы достижения прогресса и исполнения.

Изменчивость и неопределенность

Изменчивость и *неопределенность* эффективно используются в гибкой разработке по методике Scrum для создания инновационных решений. С этим связаны следующие принципы.

- Учет полезной изменчивости.
- Применение итеративной и инкрементной разработки.
- Эффективное использование изменчивости путем обследования, адаптации и прозрачности.
- Одновременное сокращение всех форм неопределенности.

Учет полезной изменчивости

В плановых процессах разработка продукции трактуется как ее изготовление. В них всячески избегается изменчивость и поощряется соответствие вполне *определенному процессу*. Но дело в том, что разработка продукции не похожа на ее изготовление. Цель изготовления — выполнить последовательный ряд вполне понятных этапов, исходя из фиксированного ряда требований, чтобы произвести готовую продукцию, которая всегда одинакова (в определенных пределах отклонений; рис. 3.3).

Рис. 3.2. Разделение принципов гибкой разработки на категории

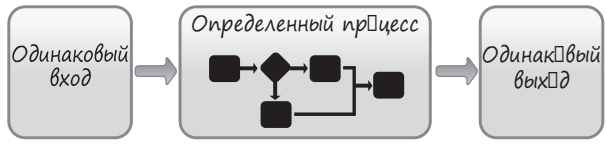


Рис. 3.3. Определенный процесс

Но цель разработки продукции — создать *единственный* в своем роде *экземпляр* продукции, а не *изготовить* ее. Этот единственный экземпляр можно сравнить с индивидуальным рецептом, который вряд ли стоит составлять дважды, чтобы не тратить зря средства. Вместо этого требуется составить индивидуальный рецепт для новой продукции. А для того чтобы производить всякий раз разную продукцию, необходима некоторая изменчивость. В действительности каждое функциональное средство, создаваемое в разрабатываемом программном продукте, отличается от любого другого функционального средства в этом же продукте, чем и объясняется потребность в изменчивости на данном уровне. И лишь после того, как будет составлен рецепт, можно изготовить продукцию. Если речь идет о программных продуктах, то для этого достаточно скопировать двоичный код.

Таким образом, некоторые принципы изготовления распространяются и на разработку продукции, и поэтому их можно применять эффективно. Например, рассматриваемый далее учет *производственных ресурсов* (или незавершенных работ) и управление ими имеет решающее значение не только для изготовления, но и для разработки продукции. Но разработка и изготовление продукции отличаются характером выполняемых работ, а следовательно, им требуются совершенно разные процессы.

Применение итеративной и инкрементной разработки

Плановая, последовательная разработка предполагает, что результаты достигаются безотлагательно и большая часть или все части продукции собираются вместе на последующем этапе работ. С другой стороны, методика Scrum опирается на *итеративную и инкрементную разработку*. Несмотря на то что оба эти обозначения видов разработки нередко применяются раздельно, итеративная разработка на самом деле отличается от инкрементной разработки.

Итеративная разработка состоит в признании того, что дело может не заладиться, прежде чем оно наладится, и что работы выполняются плохо, прежде чем они будут направлены в правильное русло [Goldberg, Adele, and Kenneth S. Rubin. 1995]. Следовательно, итеративная разработка является стратегией планируемой доработки. Улучшение процесса разработки происходит в течение нескольких проходов, чтобы найти правильное решение. Например, сначала можно создать прототип, чтобы приобрести важные сведения о малоизвестной

части продукции. Затем можно создать исправленный вариант, который в чем-то лучше, хотя он и может быть, в свою очередь, улучшен в последующем, более совершенном варианте. Например, в процессе написания этой книги я несколько раз переписывал отдельные главы по мере получения ответной реакции и уяснения способов улучшить изложение отдельной темы.

Итеративная разработка отлично подходит для улучшения продукции по мере ее разработки. Главный недостаток итеративной разработки заключается в том, что при наличии неопределенности очень трудно заранее определить (и спланировать) количество проходов, требующихся для улучшения процесса разработки.

Инкрементная разработка опирается на старый принцип "построить хотя бы что-нибудь, прежде чем построить все". Это попытка избежать одного крупного события, совершаемого "одним махом" в конце разработки, когда все части продукции собираются вместе, и она выпускается полностью. Вместо этого продукция разделяется на мелкие части, чтобы создать хотя бы часть ее, выяснить, насколько каждая часть способна выжить в той среде, где она должна существовать, приспособиться к тому, что выяснено, а затем создать нечто большее. Так, в процессе работы над этой книгой я поочередно переписывал каждую главу и по завершении посылал ее на рецензию, вместо того чтобы пытаться получить отзыв сразу обо всей книге. Это дало мне возможность учитывать полученный отзыв в работе над последующими главами, корректируя стиль и манеру изложения или способ подачи материала. Кроме того, это дало мне возможность постепенно учиться и применять навыки, приобретенные при написании предыдущих глав, в работе над последующими главами.

Инкрементная разработка позволяет получить важные сведения, чтобы адаптировать проектные работы и внести коррективы в дальнейшее их выполнение. Главный недостаток инкрементной разработки заключается в том, что, создавая продукцию по частям, мы рискуем потерять общую картину, не видя леса за деревьями.

В методике Scrum эффективно используются преимущества как итеративной, так и инкрементной разработки, но в то же время исключаются недостатки их применения по отдельности. С этой целью гибкая разработка по методике Scrum выполняется в виде адаптивных, ограниченных по времени итераций, называемых спринтами (рис. 3.4).

В течение каждого спринта выполняются все виды деятельности, необходимые для создания прироста рабочего продукта (т.е. части, а не всего продукта в целом). Как показано на рис. 3.4, в течение каждого спринта полностью выполняется определенная работа по анализу, проектированию, построению, интеграции и тестированию. Преимущество такого единовременного подхода заключается в том, что он позволяет быстро проверить достоверность предположений, сделанных при разработке функциональных средств программного продукта.

Например, можно принять ряд проектных решений, разработать на их основании некоторый код, а затем протестировать код и проектный замысел в течение одного и того же спринта. Выполняя все взаимосвязанные работы в одном спринте, можно быстро доработать функциональные средства, а следовательно, достичь преимуществ итеративной разработки, по существу, не планируя дополнительные итерации.

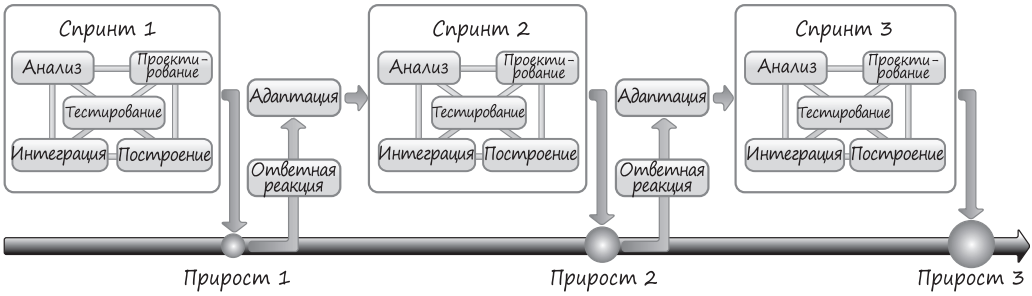


Рис. 3.4. В методике Scrum эффективно используются преимущества как итеративной, так и инкрементной разработки

Неверное употребление понятия спринта состоит в сосредоточении каждого спринта только на одном виде работ, например, спринта 1 — на анализе, спринта 2 — на проектировании, спринта 3 — на программировании, а спринта 4 — на тестировании. Такой подход представляет собой попытку перекрыть методику Scrum структурой разбиения работ в водопадном стиле. Этот ложный подход я нередко обозначаю как *водопадная схватка* (WaterScrum), а другие — как *схваткопад* (Scrummerfall).

Работа в Scrum выполняется не поэтапно, а по очереди над каждым функциональным средством в отдельности. Таким образом, в конце спринта создается ценный прирост продукта (некоторые, но не всего его функциональные средства). Этот прирост включается или интегрируется и тестируется с любыми разработанными ранее функциональными средствами, а иначе он не считается завершенным. Например, в прирост 2 на рис. 3.4 включаются функциональные средства из прироста 1. В конце спринта может быть получена ответная реакция на вновь завершенные функциональные средства в контексте уже готовых функциональных средств. Это помогает представить разрабатываемый продукт в более общей перспективе, чем можно было бы иначе ожидать.

Получая ответную реакцию на результаты спринта, мы можем адаптироваться, выбирая другие функциональные средства для работы в следующем спринте или изменяя процесс для разработки следующего ряда функциональных средств. Иногда из полученной ответной реакции мы можем узнать, что данный прирост продукта не настолько хорош, как мог бы быть, хотя он формально и отвечает

требованиям. В таком случае мы можем запланировать доработку в последующем спринте как часть обязательств в отношении итеративной разработки и непрерывного усовершенствования. Это помогает преодолеть препятствие, когда заранее неизвестно, сколько проходов потребуется для усовершенствования. Методика Scrum не требует определять количество итераций заранее. Непрерывный поток ответных реакций направит нас на проведение подходящего и экономически обоснованного количества итераций, если программный продукт разрабатывается постепенно.

Эффективное использование изменчивости путем обследования, адаптации и прозрачности

Плановые процессы разработки коренным образом отличаются от Scrum в нескольких измерениях, как показано в табл. 3.1, составленной на основании измерений, предложенных в [Reinertsen, Donald G. 2009a]. В плановом, последовательном процессе разработки допускается незначительная изменчивость на выходе или же полное ее отсутствие. Этот процесс выполняется в течение вполне определенных этапов, а отзывы на завершающей его стадии употребляются незначительно. С другой стороны, в Scrum учитывается тот факт, что в разработке продукции требуется некоторый уровень изменчивости, чтобы создать что-нибудь новое. Кроме того, в Scrum предполагается, что процесс, необходимый для создания продукта, сложен, а следовательно, его нельзя полностью определить заранее. Более того, на ранней стадии данного процесса довольно часто возникает ответная реакция, обеспечивающая правильное создание нужного продукта.

Таблица 3.1. Сравнение плановых и Scrum-процессов

Измерение	Плановый процесс	Scrum-процесс
Степень определения процесса	Вполне определенный ряд последовательных этапов	Сложный процесс, который нельзя полностью определить заранее
Произвольность на выходе	Незначительная изменчивость на выходе или же полное ее отсутствие	Изменчивость ожидаема, поскольку не предпринимаются повторные попытки создать то же самое
Количество используемых отзывов	Мало и поздно	Часто и рано

В основу Scrum положены принципы *обследования*, *адаптации* и *прозрачности*, совместно называемые в [Schwaber, Ken, and Mike Beedle. 2001] *управлением эмпирическим процессом*. Обследованию и адаптации в Scrum подлежит не только создаваемый продукт, но и порядок его создания (рис. 3.5).

Чтобы сделать это правильно, следует опираться на принцип прозрачности, который состоит в том, что все сведения, имеющие значение для выпуска

продукта, должны быть доступны для тех, кто задействован в его создании. Благодаря прозрачности оказывается возможным обследование, которое, в свою очередь, требуется для адаптации. Прозрачность позволяет также всем заинтересованным лицам наблюдать и понимать происходящее. Она способствует большей общительности и доверию как в самом процессе, так и среди членов команды разработчиков.

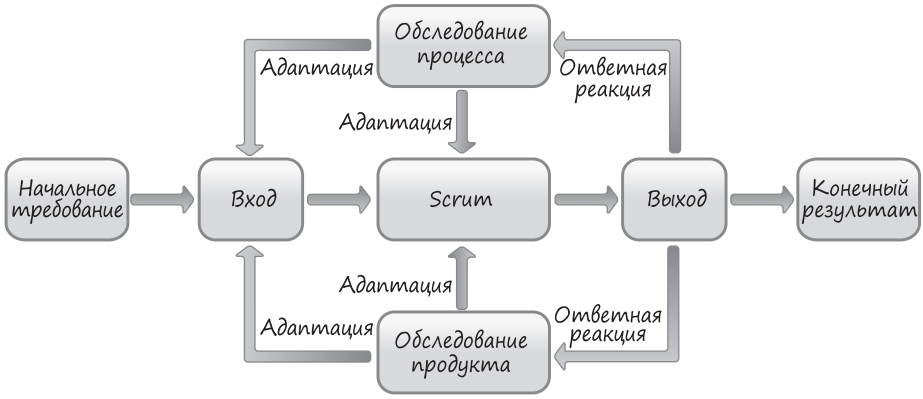


Рис. 3.5. Модель Scrum-процесса

Одновременное сокращение всех форм неопределенности

Разработка новых продуктов — сложное предприятие с высокой степенью неопределенности. Эту неопределенность можно разделить на следующие две обширные категории [Laufer, Alexander. 1996].

- **Конечная неопределенность** (т.е. неопределенность *чего-то*) — это неопределенность вокруг функциональных средств конечного продукта.
- **Неопределенность средств** (т.е. неопределенность *как-то*) — это неопределенность вокруг процесса и технологий, применяемых для разработки продукта.

В конкретных средах или у отдельных продуктов может быть также *неопределенность потребителя* (т.е. неопределенность *кого-то*). Например, в организациях, находящихся на стадии развития, в том числе и в крупных организациях, занимающихся разработкой новой продукции, могут иметься только предположения о конкретных потребителях их продукции. Такая неопределенность должна быть устранена, иначе можно создать замечательную продукцию, но не для того рынка сбыта.

В традиционных, последовательных процессах разработки первоначально упор делается на полном исключении конечной неопределенности, для чего

заранее и полностью определяется то, что требуется создать, и только после этого устраняется неопределенность средств. Такой упрощенный, прямолинейный подход к сокращению неопределенности не годится для разработки продукции в комплексной области, где наши действия и среда, в которой мы работаем, ограничивают друг друга. Рассмотрим следующий пример.

- Сначала мы решаем создать функциональное средство (это наше действие).
- Затем демонстрируем это функциональное средство заказчику, который после его просмотра меняет свое решение относительно того, что ему действительно требуется, или осознает, что он неточно сообщил требования к данному средству (т.е. наше действие вызывает ответную реакцию среды).
- Мы вносим коррективы в проект, исходя из ответной реакции (т.е. ответная реакция среды вынуждает нас предпринять непредусмотренное действие).

Методика Scrum не накладывает никаких ограничений на полное устранение одного из видов неопределенности, прежде чем устранить другой ее вид. Вместо этого мы можем предпринять более целостный подход и одновременно сделать акцент на всех видах неопределенности (конечной, средств, потребителя и т.д.). Разумеется, в любой момент времени мы можем сосредоточить основное внимание на одном, а не на другом виде неопределенности. Одновременно устранение нескольких видов неопределенности упрощается благодаря итеративной и инкрементной разработке и направляется постоянным обследованием, адаптацией и прозрачностью. Такой подход позволяет рационально опробовать и исследовать среду, чтобы выявить и узнать о *неизвестных неизвестных* по мере их появления (т.е. тех факторах, о которых мы еще знаем, что они нам неизвестны).

Прогнозирование и адаптация

Применяя методику Scrum, мы постоянно идем на компромисс между прогнозированием и потребностью в адаптации. К этой категории относятся следующие пять принципов гибкой разработки.

- Наличие свободы выбора.
- Способность принять, что нельзя все сделать сразу правильно.
- Предпочтение адаптивного, исследовательского подхода.
- Учет изменений экономически обоснованным способом.
- Оптимальное сочетание предварительно прогнозирующих работ с адаптивными своевременными работами.

Наличие свободы выбора

Для плановой, последовательной разработки требуется, чтобы важные решения в таких областях, как постановка требований или проектирование, принимались, оценивались и утверждались на соответствующих стадиях процесса. Более того, эти решения должны приниматься до перехода к следующей стадии, даже если они и основываются на ограниченных знаниях.

Методика Scrum настаивает на том, что мы вообще не должны принимать преждевременное решение просто потому, что общий процесс диктует подходящий для этого момент. Напротив, применяя методику Scrum, мы предпочитаем стратегию свободного выбора. Нередко данный принцип называется *последним ответственным моментом* [Poppendieck, Mary, and Tom Poppendieck. 2003]. Это означает, что мы задерживаем принятие на себя обязательств и не делаем важные и необратимые решения вплоть до последнего ответственного момента. Когда же наступает этот момент? Когда непринятие решения обойдется дороже, чем его принятие (рис. 3.6). Именно в этот момент и принимается решение.

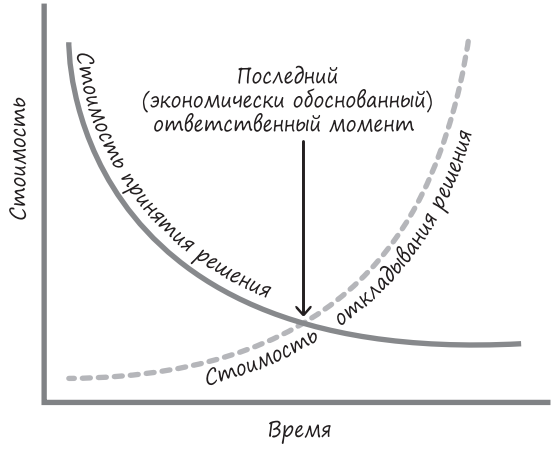


Рис. 3.6. Решения следует принимать в последний ответственный момент

Чтобы оценить данный принцип по достоинству, рассмотрим следующий пример. В первый рабочий день разработки продукта у нас имеются, по крайней мере, сведения о том, что мы делаем. На следующий рабочий день разработки продукта мы узнаем больше. Зачем же тогда вообще принимать все наиболее важные, а возможно, и необратимые решения в первый день, т.е. так рано? Большинство из нас предпочитают ждать до тех пор, пока не будут получены дополнительные сведения, чтобы принять более взвешенное решение. Если принять важное или необратимое решение слишком рано или неверно, этот момент будет соответствовать

экспоненциальной части кривой стоимости принятия решений, приведенной на рис. 3.6. По мере того, как ситуация с принятием решения разъясняется, стоимость его принятия убывает (вероятность принятия неудачного решения снижается потому, что повышается уверенность в рыночной или технической целесообразности). Именно поэтому мы должны ждать до тех пор, пока не будет получено больше информации, чтобы перейти к принятию решения.

Способность принять, что нельзя все сделать сразу правильно

Плановые процессы не только предписывают составление полного перечня требований или готового плана. Они также предполагают, что мы можем сделать это правильно заранее. Но в действительности маловероятно, чтобы все требования были получены сразу, а подробные планы — правильно и заранее составлены на основании этих требований. Хуже того, когда требования изменяются, нам приходится корректировать первоначальные требования и планы, чтобы согласовать их с текущей действительностью (подробнее об этом — в главе 5).

Применяя методику Scrum, мы признаем, что не можем составить все требования или планы заблаговременно. В действительности мы считаем, что любые попытки сделать это могут оказаться опасными, поскольку мы можем упустить важные сведения, а следовательно, составить немало требований низкого качества (рис. 3.7).

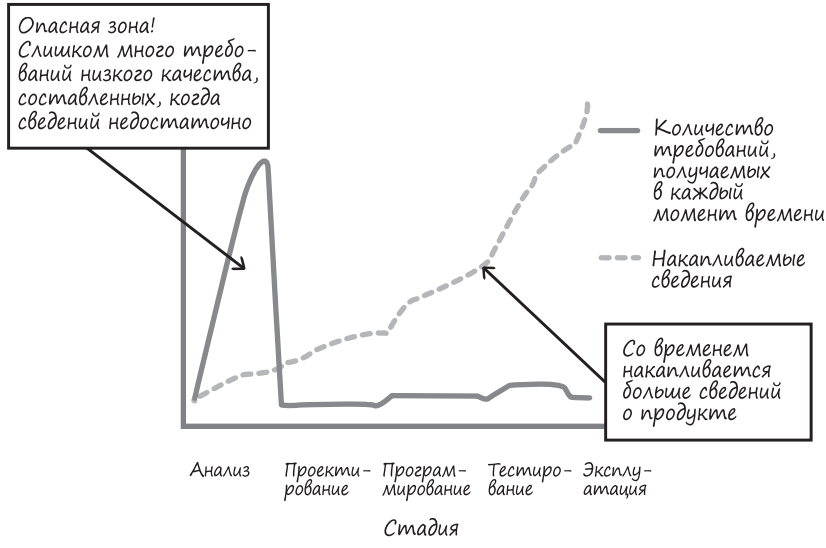


Рис. 3.7. Получение требований по плану относительно сведений о разрабатываемом продукте

На этом рисунке наглядно показано, что если применяются плановые, последовательные процессы, то бо́льшая часть требований составляется рано, когда еще не накоплено достаточно сведений о разрабатываемом продукте. Такой подход рискован, поскольку он создает иллюзию, будто мы устранили конечную неопределенность. Кроме того, он потенциально оказывается весьма расточительным по мере того, как наши представления о разрабатываемом продукте расширяются, а ситуация изменяется (подробнее об этом — ниже).

Несмотря на то что, работая по методике Scrum, нам по-прежнему приходится составлять некоторые требования и планы заранее, это делается лишь в достаточной степени, исходя из предположения, что подробности этих требований и планов будут уточнены по мере получения дополнительных сведений о разрабатываемом продукте. Ведь даже если мы полностью уверены в том, что нужно разрабатывать и как заранее организовать для этого работу, мы все равно узнаем, где мы ошиблись, как только попытаемся внедрить ранние постепенно наращиваемые варианты конечного продукта в той среде, где он должен существовать. И в этот момент все проявляющие неудобные реалии принудят нас внести изменения.

Предпочтение адаптивного, исследовательского подхода

Плановые, последовательные процессы сосредоточены на использовании того, что в настоящий момент известно, и прогнозировании того, что еще неизвестно. Методика Scrum отличается более адаптивным подходом проб и ошибок, основанном на соответствующем применении результатов исследования.

Исследование означает момент времени, когда мы предпочитаем приобрести знания, необходимые для некоторой деятельности, например, для построения прототипа, создания опытного образца, проведения изысканий или экспериментов. Иными словами, сталкиваясь с неопределенностью, мы приобретаем необходимые знания и сведения путем экспериментирования.

Применяемые нами инструментальные средства и технологии оказывают существенное влияние на стоимость исследования. Исторически сложилось так, что исследование разработки программного обеспечения всегда обходилось дорого, поскольку в нем применялся более предсказуемый подход, состоявший в попытке сразу добиться правильного результата (рис. 3.8).

В качестве примера я вспоминаю себя первокурсником Технологического института в шт. Джорджия в начале 1980-х годов, когда я пользовался перфокартами — таким же неудобным (вплоть до отвращения) средством для внесения любых корректив и исправления ошибок, как и пишущая машинка. Мне было трудно принять такой подход: “Быстро попробуем и посмотрим, что из этого получится”, когда для опробования каждого возможного решения приходилось усердно создавать перфокарты, становиться в очередь на доступ к ЭВМ

и ждать целые сутки результатов проверки правильности своего решения. Ведь даже простая опечатка обходилась минимум в сутки потраченного календарного времени. Водопадный процесс, допускавший тщательное рассмотрение текущих сведений и прогнозирование при наличии неопределенности в попытке найти правильное решение, просто имел экономический смысл.

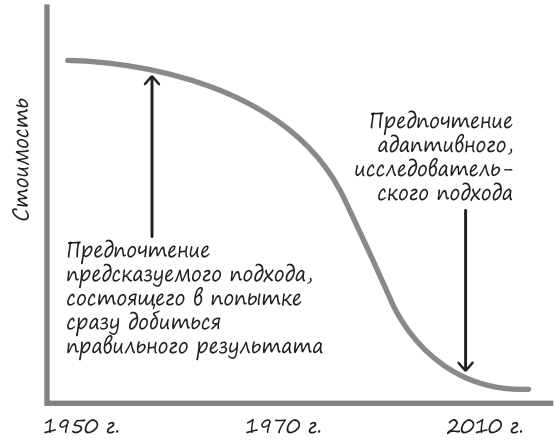


Рис. 3.8. Исторически сложившаяся стоимость исследования

К счастью, инструментальные средства и технологии усовершенствовались, а стоимость исследования сократилась. Теперь ничто экономически не сдерживает проведение исследования. В действительности оказывается намного дешевле приспосабливаться к ответной реакции пользователей, быстро разрабатывая программный продукт, чем вкладывать средства в попытки сразу сделать все правильно. И это совсем не плохо, поскольку контекст (окружающие технологии), в котором должны существовать решения, постоянно усложняется.

Если, применяя методику Scrum, мы обладаем достаточными знаниями и сведениями, чтобы совершить осознанный, обоснованный шаг вперед, мы продвигаемся дальше в поисках правильного решения. Но когда мы сталкиваемся с неопределенностью, то вместо попыток спрогнозировать, мы проводим недорогое исследование, чтобы приобрести подходящие сведения и воспользоваться ими для совершения осознанного, обоснованного шага вперед. Ответная реакция на наши действия поможет нам решить, стоит ли продолжать исследование и когда это нужно сделать.

Учет изменений экономически обоснованным способом

Как мы уже знаем, чем позднее вносятся изменения при последовательном подходе к разработке, тем дороже они обходятся. Это обстоятельство наглядно иллюстрирует рис. 3.9, созданный по материалам [Boehm, Barry W. 1981].

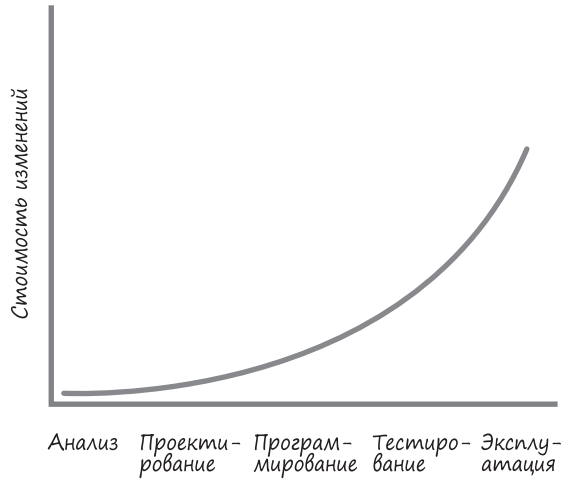


Рис. 3.9. Чем позже вносятся изменения, тем дороже они обходятся

Например, изменения, вносимые на стадии анализа, могут стоить 1 доллар, а те же самые изменения на более поздней стадии тестирования — 1000 долларов. Почему это именно так? Если мы совершаем ошибку и обнаруживаем ее на стадии анализа, то ее исправление обойдется недорого. А если та же самая ошибка обнаруживается уже на стадии проектирования, то придется исправить не только неверно составленное требование, но, возможно, и отдельные части проекта, в зависимости от того, насколько неверно было составлено требование. Такое накапливание ошибок продолжается на каждой последующей стадии. В итоге мелкая поначалу ошибка, которую нетрудно было исправить на стадии анализа, превращается в более крупную ошибку на стадии тестирования или эксплуатации.

Чтобы избежать поздних изменений, в последовательных процессах осуществляется поиск способов тщательного контроля и сведения к минимуму любых изменений в требованиях или проектных решениях. С этой целью повышается точность прогнозов относительно того, что и как должна делать проектируемая система.

К сожалению, чрезмерное прогнозирование на ранних стадиях процесса разработки нередко приводит к противоположным результатам. Этим не только не удается исключить изменения, но и фактически задержать выпуск продукции с превышением бюджета. В чем же парадоксальность этой истины? Во-первых, стремление исключить дорогостоящие изменения вынуждает нас чрезмерно вкладывать средства на каждой стадии, выполняя больше работы, чем нужно и целесообразно. И во-вторых, мы сосредоточены на принятии решений, опираясь на важные предположения на ранней стадии процесса, прежде чем мы сможем проверить эти предположения по ответной реакции участников проекта

и с учетом наших рабочих ресурсов. В итоге мы производим крупный запас рабочих продуктов, исходя из этих предположений. А в дальнейшем этот запас, скорее всего, придется исправлять или отвергать по мере проверки достоверности (или недостоверности) наших предположений или происходящих изменений (например, появления новых или развития существующих требований), как это обычно и бывает. Это вполне согласуется с классическим образцом сбывающегося пророчества (рис. 3.10).

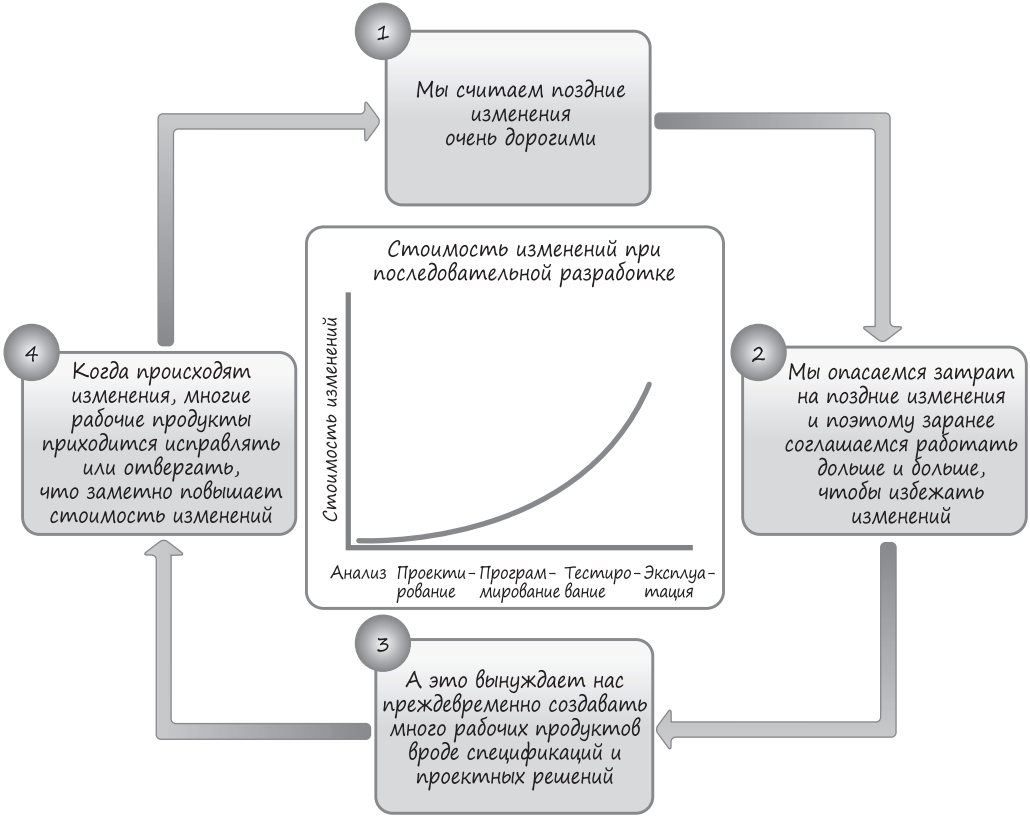


Рис. 3.10. Сбывающееся пророчество

Действуя по методике Scrum, мы допускаем изменения как норму. Мы считаем, что вряд ли сможем предвидеть неопределенность, свойственную разработке продукции, если заранее согласимся работать дольше и больше. Следовательно, мы должны быть готовы учесть изменения. А когда эти изменения происходят, нам нужно выбрать более привлекательный и экономичный подход, чем традиционная разработка, даже если изменения происходят на поздней стадии разработки продукции.

Таким образом, наша цель — сохранить как можно более пологой кривую стоимости изменений, т.е. сделать экономически обоснованными даже поздние изменения. Этот замысел наглядно демонстрируется на рис. 3.11. Данной цели мы можем добиться, управляя объемом незавершенных работ и ходом этих работ таким образом, чтобы время оказывало меньшее влияние на стоимость изменений в проекте, когда мы пользуемся методикой Scrum по сравнению с последовательным подходом к разработке.

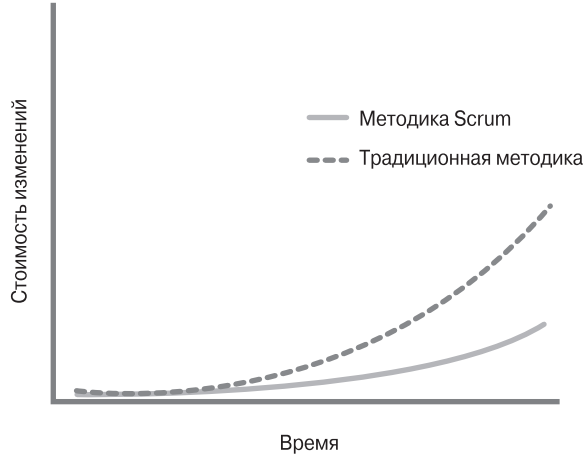


Рис. 3.11. Спрямление кривой стоимости изменений

Независимо от подхода, применяемого для разработки продукции, нам нужно добиться следующего соотношения: мелкие изменения в требованиях должны приводить к пропорционально мелким изменениям в реализации, а следовательно, и в стоимости (очевидно, что чем крупнее изменения, тем больше их ожидаемая стоимость). Еще одно важное свойство данного соотношения состоит в том, что оно должно оставаться в силе независимо от того, когда именно делается запрос на изменение.

Применяя методику Scrum, мы производим много рабочих продуктов, в том числе подробных требований и контрольных примеров, в срок, избегая создания потенциально ненужных артефактов. Когда вносится изменение, то в конечном итоге, как правило, оказывается намного меньше артефактов или ограничивающих решений, основанных на предположениях, которые могут быть отвергнуты или доработаны. Таким образом, стоимость сохраняется более пропорциональной объему запрашиваемых изменений.

Если разработка выполняется по последовательной методике, создание артефактов на ранней стадии и побуждение к преждевременному принятию окончательных решений означает, что стоимость изменений быстро возрастает со

временем по мере накопления запасов. Это приводит к тому, что точка перегиба рано появляется на традиционной кривой, приведенной на рис. 3.11, где линия начинает резко подыматься вверх.

Оптимальное сочетание предварительно прогнозирующих работ с адаптивными своевременными работами

С точки зрения плановой методики разработки подробное заблаговременное составление требований и планов имеет решающее значение и должно быть выполнено перед тем, как перейти к последующим стадиям процесса. А с точки зрения методики Scrum считается, что заблаговременно выполняемая работа должна быть полезной, но не чрезмерной.

Применяя методику Scrum, мы признаем, что получить сразу требования и планы невозможно. Означает ли это, что мы не должны составлять никаких требований или планов заранее? Разумеется, нет! Методика Scrum состоит в том, чтобы найти золотую середину между предварительно прогнозирующими работами и адаптивными своевременными работами, как показано на рис. 3.12, адаптированном из иллюстрации в [Cohn, Mike. 2009].

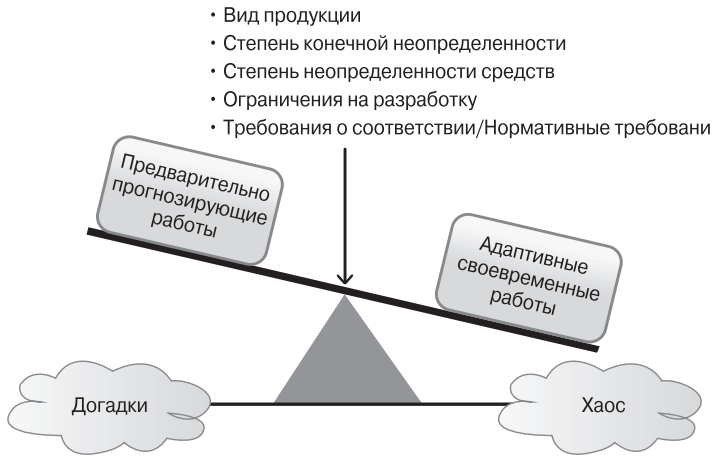


Рис. 3.12. Оптимальное сочетание предварительно прогнозирующих работ с адаптивными своевременными работами

При разработке продукции точка равновесия должна быть установлена экономически обоснованно, чтобы увеличить до максимума объем работ по непрерывной адаптации, исходя из быстрой ответной реакции, но свести к минимуму объем работ по предварительному прогнозированию. При этом должны быть удовлетворены требования о соответствии, нормативные требования и достигнуты корпоративные цели.

Конкретный способ достижения такого равновесия отчасти определяется видом разрабатываемой продукции, степенью неопределенности, существующей в том, что именно требуется создать (т.е. конечной неопределенности) и как это сделать (т.е. неопределенности средств), а также ограничениями, накладываемыми на разработку. Если слишком увлечься прогнозированием, это потребует от нас сделать немало допущений при наличии значительной неопределенности. А если слишком увлечься адаптацией, то мы можем оказаться в состоянии постоянных перемен, что сделает нашу работу неэффективной и хаотичной. Для быстрой разработки передовых продуктов нам нужно работать в пространстве, где адаптивность уравновешивается достаточным прогнозированием, чтобы не скатиться в хаос. Инфраструктура Scrum отлично действует в этой точке равновесия порядка и хаоса.

Утвержденное обучение

Применяя методику Scrum, мы организуем работу таким образом, чтобы быстро достичь состояния *утвержденного обучения* — термина, предложенного в [Ries, Eric. 2011]. Мы проходим утвержденное обучение, когда получаем сведения, подтверждающие или опровергающие сделанное предположение. К этой категории относятся следующие три принципа гибкой разработки.

- Быстрое утверждение важных предположений.
- Эффективное использование нескольких параллельных петель обучения.
- Организация рабочего процесса для получения быстрой ответной реакции.

Быстрое утверждение важных предположений

Предположение — это догадка или мнение о том, что сделанное допущение истинно, реально или определенно, даже если отсутствует утвержденное обучение, позволяющее узнать, что это истина. Плановая разработка намного более терпима к долгосрочным предположениям, чем методика Scrum. Применяя методику плановой разработки, мы составляем обширные предварительные требования и планы, скорее всего, вносящие многие важные предположения, которые вряд ли будут проверены до тех пор, пока не наступит довольно поздняя стадия разработки.

Предположения представляют значительный *риск* для разработки. Применяя методику Scrum, мы пытаемся свести к минимуму количество важных предположений, существующих в любой момент. Мы также не должны допускать долгое существование важных предположений без проверки их достоверности. Сочетание итеративной и инкрементной разработки вместе с сосредоточением внимания на недорогом исследовании может быть использовано для быстрой

проверки предположений. В итоге, если мы, применяя методику Scrum, сделаем совершенно неверное предположение, то, скорее всего, быстро обнаружим свою ошибку и получим возможность ее исправить. Если же, применяя методику плановой, последовательной разработки, мы сделаем такое же неверное предположение, то проверка его достоверности на поздней стадии процесса может привести к значительному сбою или полному провалу проектных работ.

Эффективное использование нескольких параллельных петель обучения

Обучение происходит и в том случае, если применяется методика последовательной разработки. Но важная форма обучения происходит только после создания, интеграции и тестирования функциональных средств. Это означает, что существенное обучение происходит ближе к концу выполняемых работ. Позднее обучение приносит меньшие выгоды, поскольку мало времени для эффективного обучения или стоимость его слишком высока.

Применяя методику Scrum, мы понимаем, что постоянное обучение служит залогом успеха. При этом мы выявляем и применяем петли обратной связи, чтобы повысить эффективность обучения. Повторяющаяся форма такого вида разработки продукции состоит в том, чтобы сделать сначала предположение (или поставить цель), создать что-нибудь (выполнить некоторые виды работ), получить ответную реакцию (или отзыв) на созданное, а затем воспользоваться этой ответной реакцией, чтобы обследовать то, что мы сделали, сравнивая его с тем, что мы предполагали сделать. Далее мы адаптируем продукт, процесс и/или наши представления, исходя из того, чему мы научились (рис. 3.13).

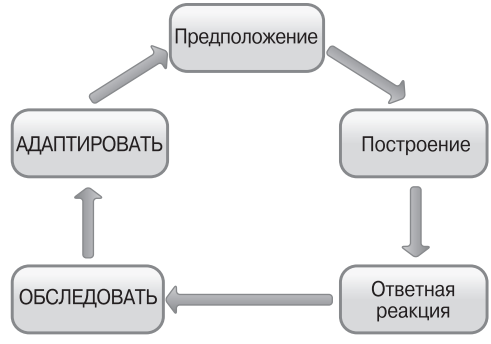


Рис. 3.13. Образец петли обучения

Методика Scrum позволяет эффективно использовать несколько предопределенных *петель обучения*. Например, ежедневная летучка является ежедневной петлей обучения, а подведение итогов спринта — петлей обучения на уровне

итерации. Эти разновидности петель обучения подробнее рассматриваются в последующих главах.

Инфраструктура Scrum достаточно удобна для охвата многих других петель обучения. Например, петли обучения ответной реакции на нормы инженерной практики вроде парного программирования (ответная реакция в течение секунд) и разработки посредством тестирования (ответная реакция в течение минут) нередко применяются в гибкой разработке по методике Scrum, хотя они и не предписываются этой методикой.

Организация рабочего процесса для получения быстрой ответной реакции

Допустимость долгосрочных предположений делает плановые процессы терпимыми и к позднему обучению, а следовательно, в них не делается акцент на *быстрой ответной реакции*. Применяя методику Scrum, мы стремимся достичь ответной реакции как можно быстрее, поскольку это очень важно для отсечения как можно раньше неверных путей и быстрого открытия и исследования оперативно возникающих возможностей.

При плановой разработке каждый вид работ планируется на конкретное назначенное время, исходя из вполне определенной последовательности стадий процесса. Такой подход предполагает, что ранние виды работ могут быть выполнены без реакции, возникающей в ответ на поздние виды работ. В итоге может возникнуть большой промежуток времени между выполнением определенных видов работ и получением ответной реакции на них, замыкающей петлю обучения.

Рассмотрим в качестве примера интеграцию и тестирование компонентов программного продукта. Допустим, что мы разрабатываем параллельно три компонента. В какой-то момент эти компоненты должны быть интегрированы и протестированы, прежде чем будет получен готовый к поставке продукт. До тех пор, пока мы не попробуем выполнить интеграцию компонентов, мы не узнаем, правильно ли были разработаны эти компоненты. Попытка интегрировать компоненты вызовет важную ответную реакцию на результаты их разработки.

Если применяется методика последовательной разработки, то интеграция и тестирование не произойдут до тех пор, пока не настанет заранее последующая стадия, на которой должны быть интегрированы многие или все компоненты. К сожалению, замысел сначала разработать ряд компонентов параллельно, а затем плавно собрать их вместе в единое целое на стадии интеграции вряд ли осуществим. В действительности, даже при наличии хорошо продуманных интерфейсов, определенных перед разработкой компонентов, вполне возможно, что при их интеграции что-нибудь пойдет не так (рис. 3.14).

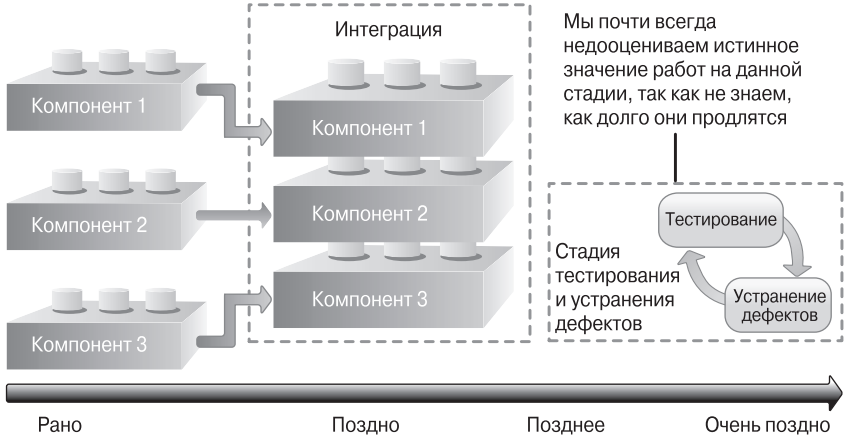


Рис. 3.14. Интеграция компонентов

Виды работ, вызывающие ответную реакцию и происходящие спустя долгое время после разработки, имеют неуместные побочные эффекты, превращающие интеграцию в обширную стадию тестирования и устранения дефектов, поскольку компоненты, разработанные разрозненно, нередко не интегрируются плавно. На данной стадии можно только гадать, сколько времени потребуется для устранения дефектов и во что это обойдется.

Применяя методику Scrum, мы организуем ход работ таким образом, чтобы как можно быстрее получить ответную реакцию, перемещаясь по петле обучения, приведенной на рис. 3.13. При этом мы гарантируем, что виды работ, вызывающие ответную реакцию, происходят очень близко по времени к первоначальной работе. Быстрая ответная реакция дает больше экономических выгод, поскольку ошибки накапливаются, когда ответная реакция задерживается, а в итоге сбои вырастают в геометрической прогрессии.

Вернемся к рассмотренному ранее примеру интеграции компонентов. Разрабатывая компоненты, мы сделали ряд важных предположений о порядке их интеграции. В зависимости от этих предположений мы продвигаемся по пути проектирования. На данной стадии мы еще не знаем, насколько верно был выбран путь проектирования, но имеем лишь наилучшее предположение об этом.

Но как только мы выберем такой путь, то можем принять много других решений, основываясь на данном выборе. И чем дольше мы будем ждать проверки достоверности первоначального проектного предположения, тем больше зависимых решений придется принимать. Если в дальнейшем мы определим (по ответной реакции на стадии интеграции), что первоначальное предположение оказалось неверным, то в конечном в итоге получим в свое распоряжение беспорядочную мешанину. Нам не только придется пересматривать неверные решения,

но и делать это много времени спустя. А поскольку память у людей недолговечна, то им приходится тратить немало времени, чтобы поскорее вернуться к той работе, которую они уже выполняли раньше.

Если принять во внимание общую стоимость пересмотра потенциально неверных зависимых решений, а также стоимость задержки выпуска продукции, то экономические выгоды от быстрой ответной реакции оказываются весьма привлекательными. Быстрая ответная реакция немедленно замыкает петлю обучения, позволяя нам отсекать неверные пути разработки, прежде чем они смогут нанести серьезный экономический ущерб.

Незавершенные работы

Незавершенные работы обозначают работы, которые были начаты, но еще не завершены. В процессе разработки продукции нужно учесть незавершенные работы и организовать надлежащее управление ими. К этой категории относятся следующие четыре принципа гибкой разработки.

- Использование экономически обоснованных объемов партий.
- Учет запасов и управление ими для оптимального хода работ.
- Акцент на простаивающей работе, а не на простаивающих работниках.
- Учет стоимости задержки.

Использование экономически обоснованных объемов партий

Еще одно внутреннее убеждение, на котором основываются плановые, последовательные процессы, состоит в том, что все однотипные виды работ желательно командировать вместе и выполнять их в течение одной стадии. Такой подход по принципу “сначала одно, а затем другое” предполагает, что мы выполняем один вид работ полностью (или в основном), прежде чем начинать другой вид работ. Допустим, что мы составили все требования на стадии анализа. Далее мы переносим партию требований на следующую стадию проектирования. А поскольку мы составили все требования, то *объем* их *партии* в данном случае составляет 100%.

Подход по принципу “сначала одно, а затем другое” отчасти является следствием убеждения в том, что старый принцип экономии от увеличения масштабов производства применим к разработке продукции. В соответствии с этим принципом стоимость производства единицы продукции сокращается по мере увеличения количества производимых единиц (объема партии). Таким образом, методика последовательной разработки основывается на том убеждении, что чем крупнее партии в разработке продукции, чем больше экономия от увеличения масштабов.

Следуя методике Scrum, мы признаем, что догматичное применение принципа экономии от увеличения масштабов к разработке продукции способно нанести значительный экономический ущерб, несмотря на то, что это основополагающий принцип производства. Как ни странно, работа малыми партиями во время разработки продукции имеет немало преимуществ. Подробно объемы партий обсуждаются в [Reinertsen, Donald G. 2009b], а в табл. 3.2 перечислены преимущества малых объемов партий.

Если мелкие партии лучше, чем крупные, то не следует ли просто употреблять партию единичного объема, т.е. работать одновременно только над одним требованием, пропуская его через все виды работ до тех пор, пока оно не будет реализовано и готово для заказчика? Некоторые называют такой подход *потоком единичных изделий*. Как будет показано в последующих главах, единичный объем партии может оказаться пригодным в некоторых случаях, но если поставить целью единичный характер разработки, то поток в частности и вся разработка в целом окажутся экономически неэффективными.

Таблица 3.2. Преимущества малых объемов партий

Преимущество	Описание
Сокращение времени цикла	Чем меньше партии, тем меньшие объемы работ ожидают выполнения, а это, в свою очередь, означает сокращение времени ожидания завершения работ. Таким образом, работы выполняются быстрее
Сокращение изменчивости потока	Представим себе ресторан, через который плавным потоком проходят небольшие вечеринки. А теперь представим себе влияние на поток посетителей ресторана, когда подъезжает большой экскурсионный автобус и все его пассажиры (крупной партией) направляются в ресторан
Ускорение ответной реакции	Мелкие партии ускоряют ответную реакцию, умаляя последствия ошибок
Сокращение риска	Мелкие партии представляют собой меньшие запасы, подверженные изменениям. Чем меньше партии, тем меньше вероятность сбоев в работе (риск сбоев в десяти отдельных видах работ выше, чем в пяти)
Сокращение издержек	Управление крупными партиями влечет за собой значительные издержки. Например, управлять перечнем из 3 тысяч видов работ труднее, чем перечнем из 30 видов работ
Повышение мотивации и настойчивости	Мелкие партии побуждают к сосредоточенности и чувству ответственности. Понять последствия задержек и сбоев в работе намного легче, имея дело с мелкими, а не крупными партиями
Сокращение затрат и сроков выполнения работ	Ошибки в крупных партиях оказываются крупными в отношении затрат и сроков. А если работы выполняются мелкими партиями, то и ошибки намного мельче

Учет запасов и управление ими для оптимального хода работ

В этой главе постоянно подчеркивается, что производство и разработка продукции — это не одно и то же, а следовательно, и подходы к ним должны быть разные. Но из опыта производства следует извлечь один полезный для разработки урок, что делается нечасто. Этот урок связан с высокой стоимостью запасов, которые называются также незавершенным производством или незавершенными работами. В области бережливой разработки продукции с нулевыми запасами значение незавершенных работ известно уже давно [Reinertsen, Donald G. 2009b; Poppendieck, Mary, and Tom Poppendieck. 2003]. И Scrum-команды принимают это понятие во внимание.

Производители отчетливо осознают важность запасов и их финансовые последствия. А как же иначе? Запасы быстро накапливаются, ожидая обработки. Производственные запасы не только физически осязаемы, но и финансово ощутимы. Если вы спросите у финансового директора производственной компании, каковы производственные запасы (или незавершенное производство) на его предприятии и насколько они изменились за последний месяц, то получите точный ответ.

Ни один грамотный производитель не сидит на крупных запасах. Детали, сложенные на полу производственного помещения в ожидании сборки в готовую продукцию, обесцениваются в бухгалтерских книгах. Еще хуже, если закупить целый груз деталей, а затем изменить конструкцию изготовляемой продукции. Что в таком случае делать со всеми этими деталями? Может быть, доработать детали, чтобы подогнать их под новую конструкцию или просто отказаться от них, поскольку они уже никуда не годятся, что еще хуже? А может быть, все-таки избежать напрасных расходов на уже приобретенные детали, не изменяя конструкцию, даже если это и верное проектное решение, чтобы использовать эти детали, хотя и с риском произвести менее удовлетворительную продукцию?

Очевидно, что если сидеть на крупных запасах и затем изменить что-нибудь в производимой продукции, то значительных убытков не избежать. Чтобы свести к минимуму подобные риски, грамотные производители управляют запасами экономически обоснованным способом. Они держат в своем распоряжении некоторые запасы, но руководствуются здравым смыслом для своевременного управления запасами.

Вообще говоря, организации, занимающиеся разработкой продукции, и близко не отдают себе отчет в своих незавершенных работах. Отчасти это объясняется тем, что, разрабатывая продукцию, мы имеем дело с ресурсами знаний, которые не так физически осязаемы, как сборочные детали на полу производственного помещения. Ресурсы знаний вроде кода на диске, документа в картотеке или панели визуального отображения на стене не столь навязчивы.

Запасы в разработке продукции, как правило, не ощутимы и финансово. Если спросить финансового директора организации, занимающейся разработкой продукции, какие запасы существуют в его организации, он, скорее всего, бросит озадаченный взгляд и ответит: “Никаких”. И если финансовый отдел организации отслеживает другие количественные показатели проектных работ, то он вряд ли следит за такого рода запасами в разработке продукции.

К сожалению, запасы (или незавершенные работы) очень важны, и поэтому ими нужно каким-то образом управлять в процессе разработки продукции. В традиционных подходах к разработке продукции такому управлению вообще не уделяется внимание. Как упоминалось выше при обсуждении объемов партий, устанавливая довольно крупный (зачастую 100%) объем партии, традиционная методика разработки продукции отдает предпочтение созданию крупных запасов. Важное следствие из наличия немалого объема незавершенных работ в процессе разработки продукции состоит в том, что он оказывает значительное влияние на упоминавшуюся ранее кривую стоимости изменений (см. рис. 3.9).

Если мы приступаем к разработке, нам нужны некоторые, хотя и не *все* требования. А если у нас имеется много требований, то мы, скорее всего, расточим запасы, когда требования изменятся. С другой стороны, если у нас имеются недостаточные запасы требований, мы нарушим быстрый ход работ, что также является формой расточительства. Наша цель в Scrum — найти золотую середину между достаточными и чрезмерными запасами.

Очень важно понять, что требования являются лишь одной из форм запасов, существующих в разработке продукции. В процессе разработки продукции имеется немало мест и моментов, где имеются незавершенные работы. Поэтому нам нужно заранее выявить их и управлять ими.

Акцент на простаивающей работе, а не на простаивающих работниках

Следуя методике Scrum, мы считаем, что простаивающая работа намного более расточительна и экономически убыточна, чем простаивающие работники. *Простаивающая работа* — это такая работа, которую требуется выполнить (например, построить или протестировать что-нибудь), но нельзя сделать, поскольку нам что-то мешает. Возможно, мы бездействуем в ожидании результатов работы другой команды, и до тех пор, пока она не завершит ее, мы не сможем приступить к своей работе. А может быть, у нас столько работы, что мы просто не в состоянии сделать все сразу. В таком случае часть работы простаивает до тех пор, пока мы сможем приступить к ней. С другой стороны, *простаивающие работники* — это люди, имеющие *возможность* работать больше, но задействованы не полностью.

Во многих организациях, занимающихся разработкой продукции, больше внимания уделяется простаивающим работникам, чем убыткам от простаивающей работы. Принято, например, считать, что если нанять на работу тестировщика, то он должен уделять все свое рабочее время тестированию. Если же он занимается тестированием не все свое рабочее время, то наниматель несет убытки из-за того, что работник простаивает, когда он мог бы заниматься тестированием. И во избежание этого наниматель привлекает работника к большему числу проектов, чтобы задействовать его полностью.

К сожалению, такой подход позволяет сократить лишь одну форму расточительства (простаивание работников), но в то же время увеличивает другую (простаивание работы). И чаще всего стоимость простаивания работы оказывается намного выше, чем стоимость простаивания работников. Рассмотрим подробнее, почему это именно так.

Чтобы проиллюстрировать это положение, применим стратегию полной занятости работников к олимпийской эстафете в беге на 100 метров. С точки зрения этой стратегии такая эстафета совершенно неэффективна. Спортсменам платят за бег, а они бегают лишь четверть всей дистанции в эстафете, простаивая остальное время. И это неправильно! Если они получают полную зарплату, то должны бегать все время. Может быть, заставить их бегать вверх и вниз по трибунам или задействовать их в эстафете по параллельной дорожке, пока они не несут эстафетную палочку? В таком случае они будут полностью заняты бегом.

Как известно, победить в эстафете нельзя, если задействовать бегунов полностью. Победа присуждается лишь тому, кто первым пересечет финишную линию с эстафетной палочкой. Из этого можно сделать следующий важный вывод: “Следить нужно за эстафетной палочкой, а не за бегунами” [Larman, Craig, and Bas Vodde. 2009]. В контексте разработки продукции эстафетная палочка, лежащая на земле, равнозначна работе, готовой к выполнению, но простаивающей в ожидании нужных ресурсов. Выиграть эстафету (т.е. выпустить продукцию) нельзя, если эстафетная палочка лежит на земле. Мне лично по душе аналогия с эстафетной палочкой и бегунами, поскольку она наглядно показывает, что мы должны следить за работой, а не за работниками. Но ведь всякая аналогия хромает. В данном случае эстафетный подход к передаче работы как раз и является одним из аспектов традиционного, последовательного способа разработки, которого нам хотелось бы избежать!

Кроме того, всем известны последствия полного задействования ресурсов. Если позаимствовать график из теории массового обслуживания или очередей, то можно обнаружить очевидный ущерб, который наносит стремление задействовать ресурсы полностью (рис. 3.15).

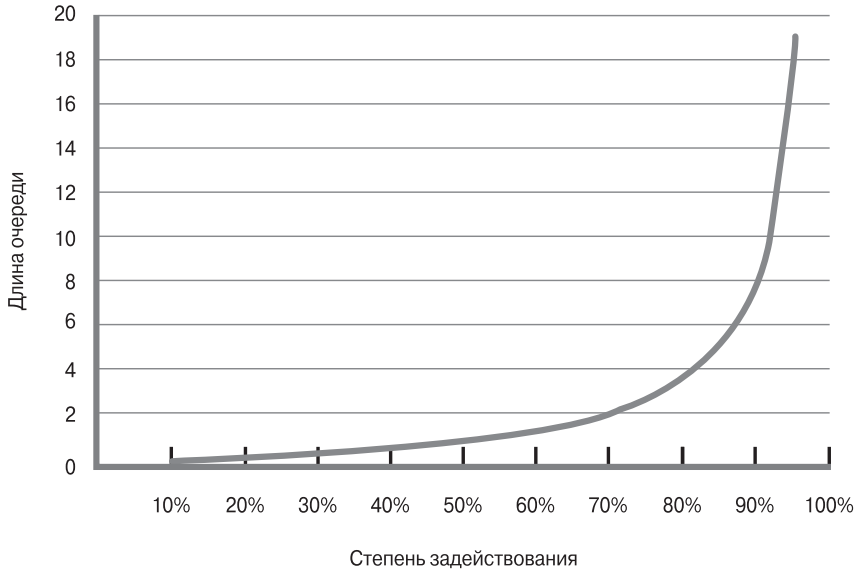


Рис. 3.15. Влияние, которое оказывает задействование ресурсов на длину очереди (задержку обслуживания)

Этот график понятен всякому владельцу компьютера. Что произойдет, если задействовать все вычислительные ресурсы компьютера (процессор и оперативную память)? Он попытается выполнить сразу все задания, а в итоге каждое из них замедлится. Иными словами, компьютер, выполняющий сразу много заданий, на самом деле работает менее производительно. Войдя в такое состояние, очень трудно выйти из него — разве что начать удалять задания по очереди или перезагрузить компьютер. Намного эффективнее компьютер будет работать в том случае, если его вычислительные ресурсы будут задействованы не более, чем на 80%. Длина очереди на рис. 3.15 приравнивается к задержке обслуживания, а задержка — к эстафетной палочке, лежащей на земле.

Объем простаивающей (задержанной) работы вырастет в геометрической прогрессии, если задействовать ресурсы в достаточно высокой степени. И эта простаивающая работа может обойтись очень дорого, зачастую во много раз дороже, чем простаивание работников (характерный пример стоимости задержки приведен в следующем разделе). Следовательно, применяя методику Scrum, мы отчетливо осознаем, что поиск узких мест в ходе работ и сосредоточение внимания и усилий на их устранении являются намного более экономически обоснованными действиями, чем попытки задействовать всех полностью.

Учет стоимости задержки

Стоимость задержки — это финансовые затраты, связанные с задержкой работы или достижением промежуточного этапа. Как показано на рис. 3.15, по мере задействования большего количества ресурсов длина очереди и задержка обслуживания возрастают. Следовательно, сокращая убытки от простаивания работников (т.е. увеличивая степень их занятости), мы одновременно увеличиваем убытки, связанные с простаиванием работы, ожидающей своей очереди на обслуживание. На основании стоимости задержки мы можем рассчитать, какие из этих убытков наносят больший экономический ущерб.

К сожалению, в 85% организаций стоимость задержки вообще не определяется количественно [Reinertsen, Donald G. 2009b]. Если добавить к этому тот факт, что в большинстве организаций, занимающихся разработкой, вообще не осознают, что у них имеется накопившаяся работа (запасы), ожидающая своей очереди на выполнение, то нетрудно понять, почему они обычно сосредоточивают все внимание на исключении видимых убытков от простаивания работников.

Рассмотрим простой пример, иллюстрирующий причины, по которым стоимость простаивания работы, как правило, выше стоимости простаивания работников. С этой целью зададимся следующим вопросом: следует ли назначить документатора для команды в начале или в конце разработки? Оба эти варианта сравниваются в табл. 3.3, хотя возможны и другие варианты.

Таблица 3.3. Пример расчета стоимости задержки

Параметр	Значение
Длительность при наличии полностью занятого документатора	**12** месяцев
Длительность при наличии документатора, назначенного в конце разработки, когда выполнены все работы, кроме документирования	**14** месяцев
Время цикла документирования в конце разработки	**2** месяца
Стоимость задержки в месяц	**250** тыс. долларов
Общая стоимость задержки	**500 тыс. долларов**
Полностью понесенные годовые затраты на содержание документатора	**90** тыс. долларов
Месячные затраты на содержание документатора	**7,5** тыс. долларов
Затраты на содержание полностью занятого документатора	**90** тыс. долларов
Затраты на содержание документатора, нанятого в конце разработки	**15** тыс. долларов
Приростные затраты на содержание полностью занятого документатора	**75 тыс. долларов**

Допустим, что документатор назначен работать над продуктом полный рабочий день в течение 12 месяцев, даже если он и не требуется все время. В этом случае приростные затраты на его содержание составят 75 тыс. долларов (их можно отнести к убыткам от простаивания работника), помимо затрат на содержание документатора, если нанять его на два месяца в конце разработки, когда выполнены все работы, кроме документирования.

Если нанять документатора для составления всей документации в конце разработки, его полная занятость потребуется лишь в течение двух месяцев. Но в то же время это задержит выпуск продукта на те же самые два месяца. А если задержать поставку продукта на два месяца, то рассчитанная стоимость задержки составит 500 тыс. долларов от прибыли в течение срока эксплуатации (*прибыль в течение срока эксплуатации* — это общая потенциальная прибыльность продукта в течение срока его эксплуатации; в данном примере она сокращается на 500 тыс. долларов).

В данном примере затраты на содержание простаивающего работника составляют 75 тыс. долларов, а стоимость простаивающей работы — 500 тыс. долларов. Если уделить внимание оптимизации занятости документатора, то можно значительно снизить общий экономический эффект от выпуска продукта. В процессе разработки нам приходится постоянно сталкиваться с подобного рода компромиссами. И стоимость задержки является одним из самых важных факторов, которые придется учитывать, принимая экономически обоснованные решения.

Прогресс

Применяя методику Scrum, мы определяем достигнутый прогресс по тому, что было произведено и проверено, а не по тому, как удалось продвинуться в соответствии с предопределенным планом, и насколько далеко удалось продвинуться на конкретной стадии или этапе разработки. К этой категории относятся следующие принципы гибкой разработки.

- Адаптация к информации в реальном времени и перепланирование.
- Определение достигнутого прогресса путем проверки достоверности рабочих ресурсов.
- Акцент на доставке ценности.

Адаптация к информации в реальном времени и перепланирование

В плановом, последовательном процессе план служит авторитетным источником того, как и когда должна быть выполнена работа. Отсюда вполне ожидаемое соответствие плану. С другой стороны, следуя методике Scrum, мы считаем,

что неистовая вера в план нередко ослепляет, заслоняя от нас тот факт, что план может оказаться неверным.

Цель гибкой разработки по методике Scrum — не придерживаться некоторого плана, т.е. предварительного прогноза относительно того, что, на наш взгляд, может последовать. Напротив, цель гибкой разработки — быстрое перепланирование и адаптация к потоку экономически важной информации, которая постоянно поступает в процессе разработки.

Определение достигнутого прогресса путем проверки достоверности рабочих ресурсов

Прогресс в процессе плановой, последовательной разработки демонстрируется завершением одной стадии и разрешением перейти к следующей. В итоге, если каждая стадия начинается и завершается так, как и предполагалось, может сложиться впечатление, будто процесс разработки продукции продвигается успешно. Но в его конце продукт, созданный в полном соответствии с намеченным планом, может иметь намного меньшую потребительскую ценность, чем предполагалось. Как можно заявлять об успешном окончании проекта, если он не отвечает ожиданиям заказчика, хотя и завершился вовремя и в рамках бюджета?

Следуя методике Scrum, мы определяем прогресс, формируя проверенные рабочие ресурсы, которые доставляют ценность и могут служить для проверки достоверности важных предположений. Это дает нам ответную реакцию, чтобы знать, какой правильный шаг будет следующим. Когда мы следуем методике Scrum, то для нас важен не первоначальный объем работы, а ценность завершенной работы для заказчика.

Акцент на доставке ценности

Разработка по плановой, последовательной методике основывается на прилежном соответствии процессу. По своему характеру интеграция и выпуск функциональных средств происходят в конце процесса последовательной разработки (рис. 3.16). При таком подходе существует риск исчерпания ресурсов (времени и средств), прежде чем вся важная ценность будет доставлена заказчикам.

С этим связано еще одно убеждение, бытующее в традиционной разработке и состоящее в том, что ценны сами артефакты планирования и документирования, получаемые по ходу выпуска функциональных средств. Если эти артефакты действительно ценны, то чаще всего они оказываются ценными только для последующего процесса, а не для заказчиков. Если же они ценны и для заказчика, то их ценность возникает только в том случае, если желательный продукт окончательно доставляется заказчику. А до тех пор эти артефакты не представляют непосредственной ценности для заказчика.

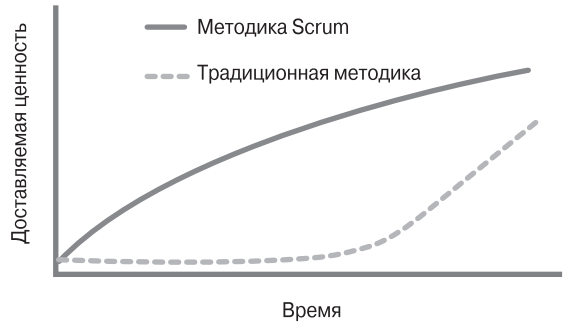


Рис. 3.16. Доставка как можно раньше высокоценных функциональных средств

С другой стороны, в гибкой разработке по методике Scrum делается акцент на потребительской ценности. Этот принцип опирается на приоритетную, инкрементную модель доставки, где наиболее ценные функциональные средства непрерывно создаются и доставляются в следующей итерации. В итоге заказчики получают как можно раньше непрерывный поток высокоценных функциональных средств.

Ценность в Scrum формируется путем доставки рабочих ресурсов заказчикам, проверки достоверности важных предположений или приобретения ценных знаний. Следуя методике Scrum, мы считаем, что промежуточные артефакты не представляют ощутимой ценности для заказчиков, а до конца процесса являются лишь средствами, если они сами непригодны для формирования важной ответной реакции или приобретения ценных знаний.

Исполнение

Применяя методику Scrum, мы ожидаем конкретные характеристики, связанные с исполнением. К этой категории относятся следующие три принципа гибкой разработки.

- Быстрое продвижение, но без спешки.
- Упор на качество.
- Сведение церемоний к минимуму.

Быстрое продвижение, но без спешки

В разработке по плановой методике существует убеждение, что если строго следовать плану и все сразу делать правильно, то можно избежать доработки, отнимающей время и средства. Разумеется, быстро продвигаться от одной стадии процесса к другой желательно, но не принципиально.

Одна из главных целей в Scrum состоит в том, чтобы действовать умело, адаптивно и быстро. Чем быстрее мы работаем, тем быстрее доставляем ценность, получаем ответную реакцию и раньше передаем ценность в руки заказчиков. Умение быстро учиться и реагировать позволяет нам скорее приносить доход и/или сокращать расходы.

Не следует, однако, путать быстроту со спешкой. Время в Scrum играет важную роль, но не нужно спешить с завершением работ. Ведь иначе будет нарушен принцип *постоянного темпа*, позволяющий людям работать размеренно в течение продолжительного периода времени. Кроме того, спешка отрицательно сказывается на качестве.

Рассмотрим пример, проясняющий разницу между быстротой и спешкой. Допустим, что речь идет об обучении тайскому боксу. Как и в большинстве других боевых искусств, исполнение приемов тайского бокса требует быстроты. Умение проворно и точно выполнять движения ката или вести тренировочный бой доставляет больше удовольствия и результатов от занятий этим видом спорта. А спешка в выполнении движений с целью добиться искомого результата значительно снижает эффективность и может привести к серьезным травмам во время тренировочного боя. Выполняя приемы тайского бокса, следует двигаться проворно, ловко и не торопясь, быстро приспосабливаясь к ситуации. Иными словами, нужно действовать быстро, но без спешки.

Упор на качество

При разработке по плановой методике существует убеждение, что благодаря тщательному, последовательному исполнению работы можно получить высококачественный продукт. Но на самом деле мы не можем проверить его качество до тех пор, пока не проведем тестирование интегрируемого продукта, что обычно происходит на поздней стадии процесса. Если тестирование должно свидетельствовать о недостаточном качестве, то нам придется перейти к дорогостоящей стадии проверки и устранения дефектов, чтобы осуществить контроль качества. А поскольку на каждой стадии процесса зачастую работают разные команды, то нередко считается, что за качество достигнутого результата отвечает команда тестирования.

Качество в Scrum — это не результат, получаемый командой тестирования в конце процесса, а нечто, принадлежащее межфункциональной Scrum-команде, непрерывно создаваемое и проверяемое в каждом спринте. Каждый прирост доставляемой ценности завершается с высоким уровнем уверенности в достигнутом качестве и имеет потенциал для передачи в эксплуатацию или поставки заказчикам (подробнее о критерии готовности речь пойдет в главе 4). В итоге значительно сокращается потребность в тщательном тестировании на поздней стадии процесса для достижения нужного качества.

Сведение церемоний к минимуму

Плановые процессы проявляют тенденцию к строгому соблюдению церемоний, тщательному документированию и сильному акценту на самом процессе. Побочный эффект сосредоточения методики Scrum на доставляемой ценности состоит в том, что церемониям вокруг процесса уделяется очень мало внимания. Нельзя сказать, что все церемонии плохи. Например, "церемония" посещения паба для общения и сближения каждую пятницу после работы может стать хорошей традицией в коллективе. В данном контексте церемония рассматривается как *излишняя формальность*. Некоторые могут назвать ее "процессом ради процесса". Такая церемония обходится дорого, но приносит мало или вообще не приносит никакой пользы. Иными словами, церемония — это своего рода расточительство.

Примерами церемоний могут служить излишние формальности, включая следующее.

- Трехдневный обременительный процесс, требующийся для утверждения и переноса кода из среды разработки в среду контроля качества, прежде чем приступать к тестированию.
- Протоколирование всех аномалий программными средствами для отслеживания и извещения о них, даже если достаточно сказать коллеге, что разработанное им функциональное средство не работает и его нужно исправить, чтобы продолжить работу дальше.
- Составление документации только потому, что настало заранее установленное время составлять документацию, несмотря на то, что никто не может объяснить, зачем эта документация вообще нужна и каковая ее ценность.

Наша цель в Scrum — исключить излишние формальности. Следовательно, мы сводим церемонии к достаточному или приемлемому минимуму. Разумеется, у каждой организации свои критерии достаточности или приемлемости. Так, если создается новый веб-сайт для социальной сети, то потребность в церемониях может оказаться исключительно низкой. А если конструируется кардиостимулятор с учетом государственных норм, то определенного рода церемоний потребуется больше, чем на минимально достаточном уровне (рис. 3.17).

Нередко акцент в Scrum на минимально достаточных церемониях неверно трактуется как "полное пренебрежение документацией". Применяя методику Scrum, мы не пренебрегаем документацией, а просто оцениваем, какая именно документация требуется с экономической точки зрения. Если мы составляем документацию как нечто бесполезное и не приносящее никакой пользы, то напрасно тратим время и средства на составление мертвого документа. Но не все

документы мертвые. Например, документ может быть составлен при следующих условиях.

- Документ является поставляемой частью продукта (например, инструкции по установке, руководство пользователя и т.д.).
- Наша цель — зафиксировать важное обсуждение, решение ли соглашение, чтобы в дальнейшем можно было легко восстановить в памяти, что именно было обсуждено, решено или согласовано.
- Документирование очень полезно, чтобы помочь новым членам команды быстро войти в курс дела.
- Имеется требование, регламентирующее составление определенной документации (затраты на эксплуатацию предприятия в регулируемой отрасли).

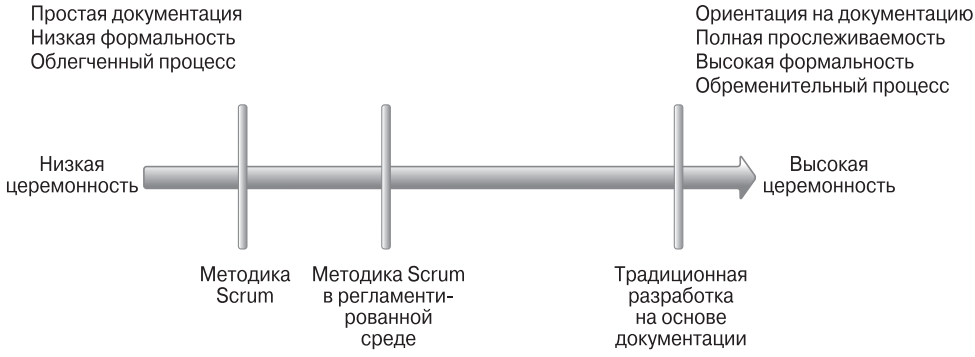


Рис. 3.17. Шкала церемоний

Следуя методике Scrum, мы пытаемся не делать работу, которая не принесет никакой экономической выгоды в краткосрочной или долгосрочной перспективе. Мы считаем, что время и средства лучше потратить на доставку потребительской ценности.

Заключение

В этой главе основное внимание было уделено описанию базовых принципов гибкой разработки — основополагающих убеждений, которыми мы руководствуемся, применяя методику Scrum. Эти убеждения были рассмотрены в сравнении с другими убеждениями, господствующими в методике традиционной, плановой, последовательной разработки, а результаты подытожены в табл. 3.4.

Таблица 3.4. Результаты сравнения принципов плановой и гибкой разработки

Вопрос	Плановый принцип	Гибкий принцип
Сходство разработки и производства	И то и другое следует вполне определенному процессу	Разработка не является производством, но составляет рецепт для продукции
Структура процесса	Разработка выполняется последовательно и стадиями	Разработка должна быть итеративной или инкрементной
Степень изменчивости процесса и продукции	Попытка исключить изменчивость процесса и продукции	Эффективное использование изменчивости путем обследования, адаптации и прозрачности
Управление неопределенностью	Исключение сначала конечной неопределенности, а затем неопределенности средств	Одновременное устранение неопределенностей
Принятие решений	Каждое решение принимается на соответствующей стадии процесса	Свобода выбора
Возможность все делать сразу правильно	Предполагается заблаговременное наличие всех сведений, требующихся для составления требований и планов	Нельзя все делать сразу правильно
Исследование в сравнении с использованием	Использование того, что уже известно, и прогнозирование того, что еще неизвестно	Предпочтение адаптивного, исследовательского подхода
Изменения и проявления	Изменения нарушают планы и обходятся дорого, и поэтому их следует избегать	Учет изменений экономически обоснованным способом
Прогнозируемость в сравнении с адаптируемостью	Процесс сильное прогнозируемый	Оптимальное сочетание предварительно прогнозирующих работ с адаптивными своевременными работами
Предположения (неутвержденные знания)	Допустимость долгосрочных предположений в процессе	Быстрая проверка достоверности важных предположений
Ответная реакция	Важное обучение происходит в одном цикле анализа, проектирования, программирования и тестирования	Эффективное использование нескольких параллельных петель обучения

Окончание табл. 3.4

Вопрос	Плановый принцип	Гибкий принцип
Быстрая ответная реакция	В процессе допускается позднее обучение	Организация рабочего процесса с учетом быстрой ответной реакции
Объем партии (определяет объем выполненной работы перед тем, как переходить к следующей работе)	Партии крупные и зачастую полные, т.е. они составляются по принципу "сначала одно, а затем другое". Потребность в экономии от увеличения масштабов	Применение мелких, экономически обоснованных объемов партий
Запасы или незавершенные работы	Запасы не являются частью системы убеждений, и поэтому им не уделяется должное внимание	Учет запасов и управление ними для оптимального хода работ
Убытки от простаивания людей или работы	Распределение трудовых ресурсов с целью их максимального использования	Акцент на простаивающей работе, а не на простаивающих работниках
Стоимость задержки	Стоимость задержки редко принимается во внимание	Стоимость задержки всегда принимается во внимание
Соответствие плану	Соответствие плану считается главным средством достижения хороших результатов	Адаптация и перепланирование вместо соответствия плану
Прогресс	Демонстрация достигнутого прогресса продвижением от одних стадий или этапов к другим	Определение достигнутого прогресса проверкой достоверности рабочих ресурсов
Сосредоточенность	Сосредоточение всего внимания на самом процессе и неуклонном следовании ему	Акцент на доставке ценности
Скорость	Следование процессу: делать сразу все правильно и продвигаться быстро	Быстрое продвижение, но без спешки
Когда достигается высокое качество	Качество достигается в конце процесса, после обширной стадии тестирования и устранения дефектов	Качество достигается с самого начала процесса
Формальность (церемонность)	Формальность (вполне определенные процедуры и контрольные точки) очень важна для эффективного исполнения	Применение минимально достаточных церемоний

Цель такого сравнения — не убедить вас в том, что водопадная методика разработки плохая, а методика Scrum хорошая, но в том, чтобы продемонстрировать, что основополагающие убеждения в водопадной методике больше подходят доя решения другой категории задач, чем в Scrum. Вы можете сами оценить, какого рода задачи стоят перед вашей организацией, а следовательно, выбрать наиболее подходящую для нее методику. В последующих главах книги будет подробно описано, каким образом эти принципы упрочивают друг друга, обеспечивая эффективный подход к разработке продукции.

Глава 4
Спринты

Работа по методике Scrum организуется итерациями или рабочими циклами длительностью до одного календарного месяца. Они называются *спринтами*, ограничиваются по времени, имеют краткую и постоянную продолжительность и цель, которую нельзя изменить, как только спринт будет начат, а также должны достигнуть конечного состояния, определяемого командой по критерию готовности.

Краткий обзор

Спринты образуют остов инфраструктуры Scrum (рис. 4.1).

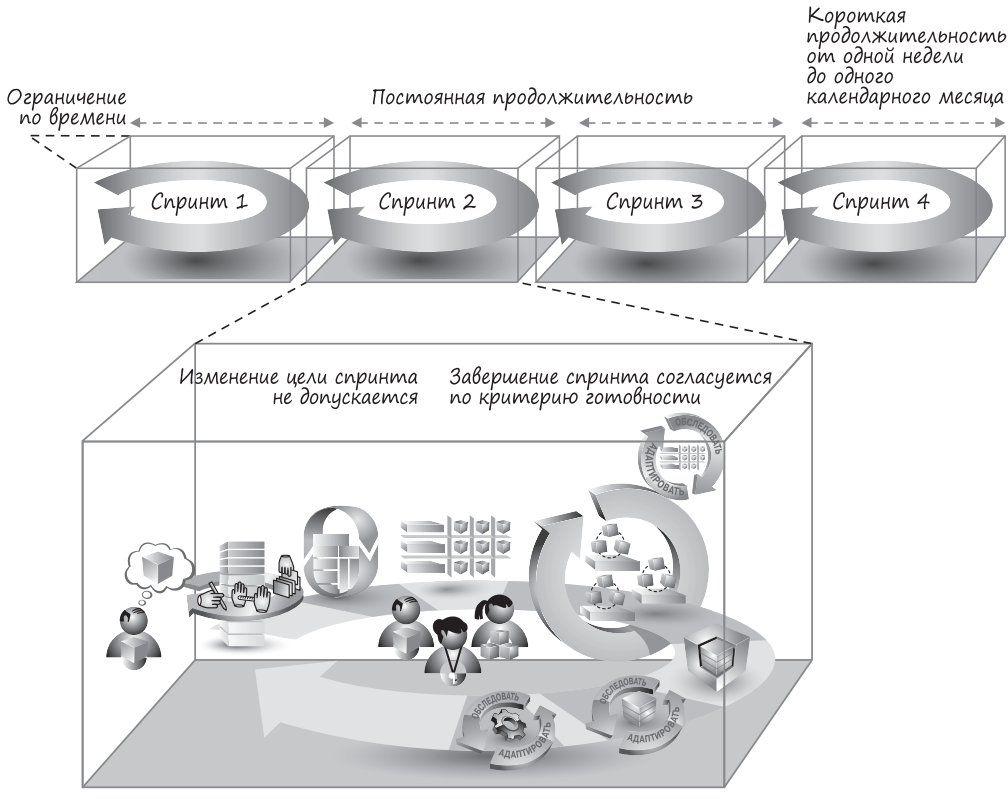


Рис. 4.1. Спринты образуют остов инфраструктуры Scrum

Крупной серой стрелкой, образующей петлю, простирающейся от задела продукта через весь цикл исполнения спринта и охватывающей всех членов Scrum-команды, на рис. 4.1 обозначается спринт, где другие артефакты и виды деятельности в Scrum показаны ориентированными по относительному времени их появления в спринте. Несмотря на то что выполнение спринта нередко путают с самим спринтом, на самом деле это лишь один из видов деятельности, происходящих в течение спринта, наряду с планированием, подведением итогов и ретроспективой спринта.

Все спринты ограничены по времени, а следовательно, имеют фиксированные даты начала и завершения. Кроме того, спринты должны быть короткими: от одной недели до одного календарного месяца в зависимости от конкретных обстоятельств. Как правило, любые изменения в объеме работ или кадрах в течение спринта не допускаются. И наконец, прирост потенциально готового к поставке продукта завершается в соответствии с согласованным Scrum-командой критерием готовности.

Несмотря на то что всякая организация реализует методику Scrum по-своему, упомянутые выше характеристики спринта (за некоторыми рассматриваемыми далее исключениями) предназначены для применения в каждом спринте и в каждой команде. Рассмотрим их по отдельности, чтобы стало понятнее их назначение.

Ограничение по времени

В основу спринтов положено понятие *ограничения по времени* — методики планирования времени, помогающей организовать исполнение работ и управлять их объемом. Каждый спринт происходит в определенных временны*х* пределах с конкретными датами начала и завершения, называемых *временными рамками*. В этих рамках команда должна работать в постоянном темпе, чтобы завершить выбранный ряд работ в соответствии с целью спринта. Ограничивать спринт по времени важно по нескольким причинам (рис. 4.2).

Установление предела для незавершенных работ

Ограничение по времени — это методика установления предела на объем незавершенных работ, которые представляют собой початые, но еще не оконченные запасы. Неумение правильно управлять объемом незавершенных работ может повлечь за собой серьезные экономические последствия. Команда планирует работать только над теми элементами из задела продукта, которые она считает способной начать и завершить в течение спринта, и поэтому ограничение по времени устанавливает предел для незавершенных работ.

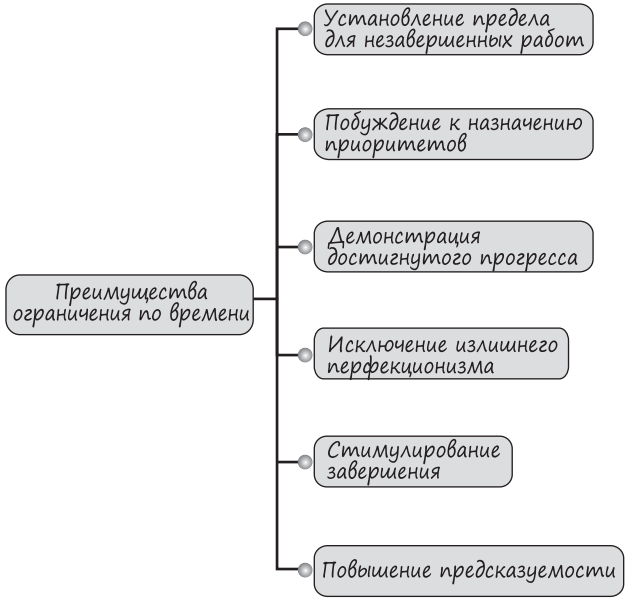


Рис. 4.2. Преимущества ограничения по времени

Побуждение к назначению приоритетов

Ограничение по времени вынуждает назначить приоритеты и начать с того небольшого объема работ, который она считает нужным выполнить в первую очередь. Благодаря этому заостряется внимание на быстром получении чего-то ценного.

Демонстрация достигнутого прогресса

Ограничение по времени помогает также продемонстрировать существенный прогресс, завершив и проверив важные части работы к известной дате (как правило, к концу спринта). Такого рода прогресс сокращает организационный риск смещением акцента с ненадежных форм извещения о достигнутом прогрессе вроде соответствия плану. Кроме того, ограничение по времени помогает продемонстрировать прогресс в отношении крупных функциональных средств, для завершения которых требуются не одни временны*е* рамки. Завершение работы по отношению к этим функциональным средствам обеспечивает ценный, ощутимый прогресс, достигнутый в каждом спринте. Оно помогает также участникам проекта и команде выяснить, что еще остается сделать, чтобы выпустить полностью функциональное средство.

Исключение излишнего перфекционизма

Ограничение по времени помогает избежать излишнего перфекционизма. Всем нам так или иначе приходилось когда-то тратить слишком много времени на попытки добиться "идеала", сделать что-нибудь "совершенное", хотя было бы достаточно и вполне приличного результата. Ограничение по времени побуждает завершить потенциально бесконечную работу к фиксированной дате окончания спринта, когда должно быть получено приемлемое решение.

Стимулирование завершения

Ограничение по времени стимулирует также завершение. Как показывает мой опыт, дело оказывается чаще всего сделанным, если команде известна дата окончания. Тот факт, что конец спринта влечет за собой жесткий срок, побуждает членов команды к прилежанию, чтобы завершить работу вовремя. Не зная дату окончания, вряд ли имеет смысл торопиться завершить работу.

Повышение предсказуемости

Ограничение по времени повышает предсказуемость. И хотя нельзя с большой определенностью предсказать, что проект будет завершен через год, вполне обоснованно предположить, что конкретная работа будет сделана в течение следующего спринта.

Краткосрочность спринтов

Краткосрочные спринты приносят немало выгод (рис. 4.3).

Простота планирования

Краткосрочные спринты проще планировать. Работу проще спланировать на несколько недель, чем на несколько месяцев. Кроме того, планирование на столь короткий период времени требует меньших затрат труда и выполняется намного точнее, чем планирование на длительный период.

Быстрота ответной реакции

Краткосрочные спринты способствуют быстрой ответной реакции. В течение каждого спринта создается работоспособное программное обеспечение, а затем появляется возможность обследовать созданный продукт, соответственно адаптировав его, а также порядок его создания. Быстрая ответная реакция позволяет сразу отсечь нежелательные пути дальнейшей разработки программного

продукта и подходы к ней, прежде чем будет принято неудачное решение, которое повлечет за собой много других решений. Кроме того, быстрая реакция позволяет своевременно раскрывать и использовать возникающие возможности, не терпящие отлагательства.

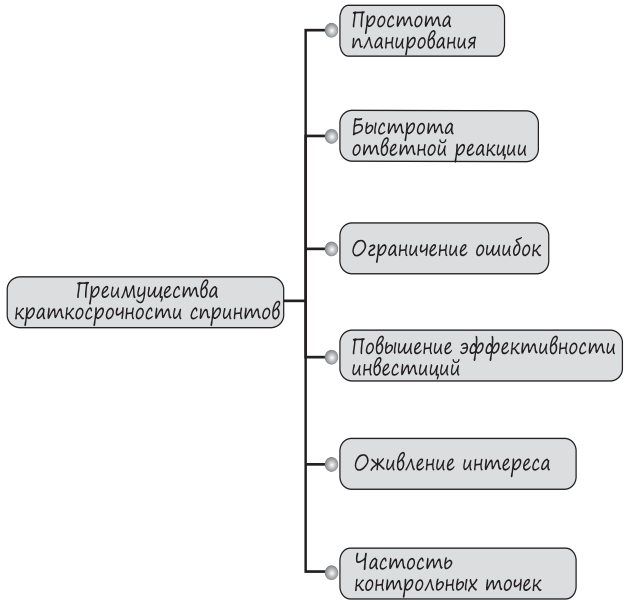


Рис. 4.3. Преимущества краткосрочности спринтов

Повышение эффективности инвестиций

Краткосрочные спринты не только улучшают экономические показатели через быструю ответную реакцию, но и позволяют раньше и чаще выпускать конечный продукт. В итоге появляется возможность получить доход раньше, а следовательно, повысить эффективность инвестиций в целом (характерный тому пример приведен в главе 14).

Ограничение ошибок

Краткосрочные спринты также ограничивают ошибки. Насколько можно ошибиться в течение двухнедельного спринта? Даже если мы испортим все дело, то потеряем только две недели. Мы настаиваем на краткосрочных спринтах, поскольку они обеспечивают частую координацию и ответную реакцию. Таким образом, если мы и ошиблись, то, по крайней мере, ненамного.

Оживление интереса

Краткосрочные спринты способствуют оживлению интереса. Человеку свойственно ослаблять интерес и энтузиазм, долго ожидая удовлетворения (рис. 4.4).

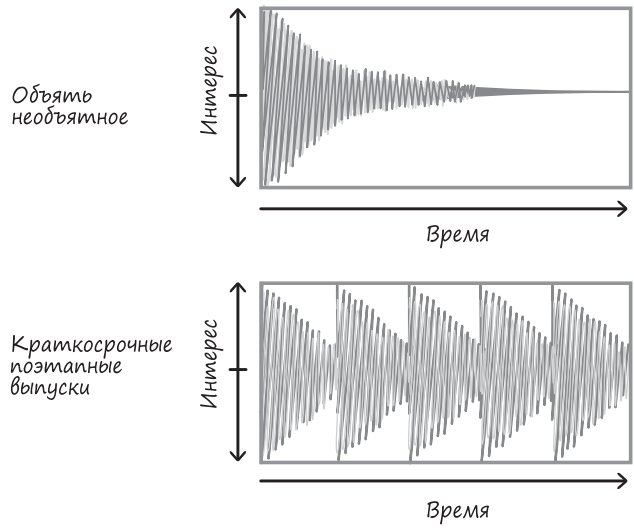


Рис. 4.4. Ослабление интереса со временем

Если мы будем работать над слишком долгосрочным проектом, то не только потерпим неудачу, но и, скорее всего, утратим в конечном итоге энтузиазм в проектных работах. (Когда я работал в компании IBM, такие проекты мы называли "объять необъятное", поскольку для их выполнения требовалось очень много времени и труда, если их вообще можно было завершить подобно попыткам объять необъятное.) В отсутствие видимого прогресса и заметного конца работы люди постепенно теряют к ней интерес. А ближе к концу проекта у них может даже возникнуть желание заплатить кому-нибудь, лишь бы перейти к другому проекту!

Краткосрочные спринты позволяют сохранить интерес участников проекта на высоком уровне, если часто доставляются рабочие ресурсы. Удовлетворение от раннего и частого получения конечного результата оживляет наш интерес и желание продолжать работать в направлении поставленной цели.

Частость контрольных точек

Краткосрочные спринты обеспечивают также многие содержательные контрольные точки (рис. 4.5).

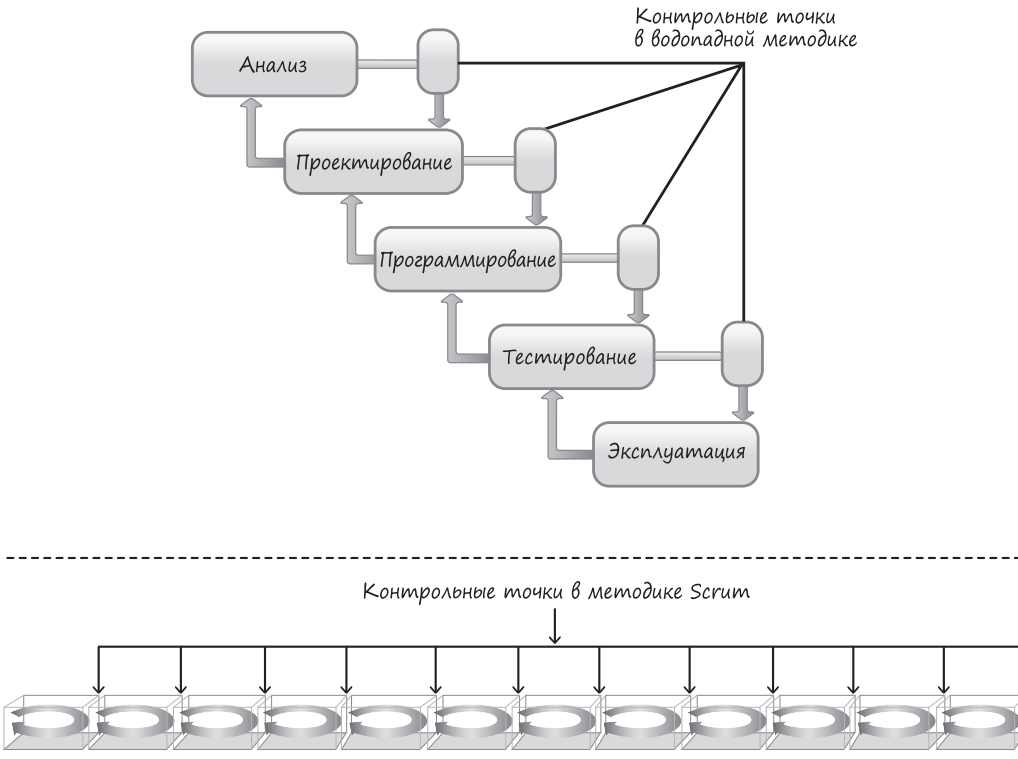


Рис. 4.5. Сравнение контрольных точек

Одним из ценных аспектов проектов, разрабатываемых по последовательной методике, является ряд вполне определенных промежуточных этапов. В течение срока действия проектов эти промежуточные этапы предоставляют руководителям известные контрольные точки, которые обычно связаны с принятием решения относительно перехода к следующей стадии процесса. И хотя эти промежуточные этапы полезны для руководства проектами, как пояснялось в главе 3, они дают ненадежное представление об истинном положении дел в доставке потребительской ценности.

Методика Scrum предоставляет руководителям, участникам проекта, владельцам продуктов и прочим заинтересованным лицам намного больше контрольных точек, чем в проектах, выполняемых по последовательной методике. В конце каждого короткого спринта появляется содержательная контрольная точка (подведение итогов спринта), позволяющая всем заинтересованным лицам основывать свои решения на демонстрируемых, рабочих функциональных средствах. Людям проще иметь дело со сложной средой, когда у них имеется больше возможностей в намечаемых контрольных точках для обследования и адаптации.

Постоянная продолжительность

Как правило, для конкретного вида проектных работ команде приходится выбирать постоянную продолжительность спринта, не изменяя ее, если на то отсутствуют веские основания. К таким веским основания могут быть отнесены следующие.

- Принимается решение перейти от четырехнедельных спринтов к двухнедельным, чтобы чаще получать ответную реакцию. Но для того чтобы принять окончательное решение, нужно попробовать провести хотя бы пару двухнедельных спринтов.
- Принимая во внимание ежегодные праздники или конец финансового года, практиковать трехнедельные спринты удобнее, чем двухнедельные.
- Продукт выпускается за неделю, и поэтому двухнедельный спринт был бы расточительством.

Тот факт, что команда не в состоянии выполнить всю работу в течение текущего спринта, нельзя считать веским основанием для увеличения продолжительности спринта. Недопустимо также дотянуть до последнего дня спринта, чтобы понять, что работа не будет сделана, и пытаться продлить спринт еще на день или неделю. Все это свидетельствует о нарушениях в нормальной организации труда и возможностях ее улучшения, но не служит веским основанием для изменения продолжительности спринта.

Преимущества размеренного ритма

Спринты одинаковой продолжительности обеспечивают *размеренный ритм* проектных работ по методике Scrum. Такой размеренный ритм дает Scrum-команде в частности и организации вообще возможность научиться работать ритмично, когда события происходят быстро, чтобы достичь быстрого и гибкого потока доставки коммерческой ценности. Как показывает мой опыт, размеренный ритм проведения спринтов позволяет людям “войти в курс дела”, “быть на коне” или “войти в колею”. На мой взгляд, это происходит потому, что размеренный ритм вырабатывает привычку к обыденным, но необходимым видам деятельности, и благодаря этому высвобождаются умственные способности для постоянного сосредоточения на интересной, повышающей ценность работе.

Наличие короткого размеренного ритма проведения спринта способствует также выравниванию интенсивности работы. В отличие от проекта, выполняемого по традиционной последовательной методике, где наблюдается резкое увеличение интенсивности работ на последних стадиях, у каждого спринта имеется такой же профиль интенсивности, как и у других спринтов. Как поясняется

в главе 11, размеренный ритм спринта позволяет командам работать в постоянном темпе.

Проведение спринтов в размеренном ритме значительно сокращает издержки на координацию. Если спринты имеют фиксированную длину, то мы можем предсказуемо запланировать график проведения мероприятий по планированию, подведению итогов и ретроспективе одновременно многих спринтов. А поскольку всем известно, когда эти мероприятия произойдут, то издержки на планирование графика их проведения для многих спринтов значительно сокращаются.

Так, если мы проводим двухнедельные спринты в ходе проектных работ, длящихся целый год, то можем запланировать повторяющееся событие в календаре всех заинтересованных лиц на подведение итогов следующих 26 спринтов. Если же мы допустим варьирование продолжительности спринтов, то нетрудно представить, что для согласования графиков работы участников проекта потребуются дополнительные усилия вместо простого извещения за одну или две недели о предстоящем подведении итогов спринта! Это также означает, что мы могли бы даже найти удобное время для основных участников проекта в их плотном графике, запланированном, скорее всего, на многие недели вперед.

И наконец, если над одним и тем же проектом работают многие команды, организация их труда в сходном размеренном ритме позволит синхронизировать работу всех команд. Подробнее об этом речь пойдет в главе 12.

Упрощенное планирование

Соблюдение постоянной продолжительности спринта упрощает также планирование видов деятельности. Если все спринты имеют одинаковую продолжительность, хотя и могут отличаться на один или два дня из-за праздников, команде удобнее выполнить объем работы в типичном спринте, что обычно называется *скоростью* ее работы. Как правило, скорость нормализуется по спринту. Если же продолжительность спринта может варьироваться, это означает, что единица нормализации спринта отсутствует, и поэтому нельзя сказать, что средняя скорость работы команды составляет 20 очков за спринт.

Скорость работы команды можно, безусловно, рассчитать, даже если команда пользуется спринтами переменной продолжительности, хотя сделать это будет труднее. Если же придерживаться постоянной продолжительности спринта, то расчеты скорости упрощаются, поскольку они опираются на данные из предыстории работы команды.

Постоянная продолжительность спринтов упрощает также остальные расчеты при планировании. Так, если мы работаем над *выпуском с жестко заданными сроками*, а продолжительность спринтов не меняется, то рассчитать количество спринтов можно непосредственно по календарю. Ведь нам известна текущая

дата, а также дата выпуска и то, что все спринты имеют одинаковую продолжительность. Если же продолжительность спринтов допускается варьировать, то расчет количества спринтов к дате выпуска может быть значительно усложнен, поскольку придется иметь дело с обширным заблаговременным планированием, потребует лишних издержек и, скорее всего, окажется намного менее надежным, чем при постоянной продолжительности спринтов.

Запрет на изменение цели спринта

Методика Scrum устанавливает важное правило. Оно состоит в том, что как только цель спринта будет установлена и начнется его выполнение, никакие изменения, способные существенно повлиять на поставленную цель спринта, не разрешаются.

Что такое цель спринта

Каждый спринт может быть подытожен по цели, описывающей коммерческое назначение и ценность спринта. Как правило, спринт преследует ясную, едино направленную цель, например, следующее.

- Поддержка составления первоначального отчета.
- Загрузка и обработка картографических данных Северной Америки.
- Демонстрация возможности посылать текстовые сообщения посредством интегрированного, встроенного программного обеспечения и аппаратно реализованного стека.

Иногда цель спринта может быть разноплановой, например: “Добиться элементарного вывода на печать и поддержки поиска по дате”. Во время планирования спринта команда разработчиков должна помогать уточнению и согласованию цели спринта, чтобы с ее помощью определить те элементы из задела продукта, которые могут быть завершены к концу спринта (подробнее об этом речь пойдет в главе 19). Эти элементы задела продукта служат для дальнейшей выработки цели спринта.

Взаимное обязательство

Цель спринта служит основанием для взаимного обязательства, которое берет на себя команда разработчиков и владелец продукта. В частности, команда обязуется достичь намеченной цели к концу спринта, а владелец продукта — не изменять эту цель на протяжении спринта.

Это взаимное обязательство демонстрирует особое значение спринтов для гибкой приспособляемости коммерческих потребностей к изменениям, позволяя в то же время команде сосредоточить основное внимание на эффективном применении способностей ее членов создавать ценность в течение короткого фиксированного периода времени. Определяя цель спринта и придерживаясь ее, Scrum-команда способна сохранять основное внимание на конечной цели, имеющей отчетливую ценность.

Изменение в сравнении с уточнением цели спринта

Несмотря на то что цель спринта фактически изменить нельзя, ее можно *уточнить*. Рассмотрим подробнее, чем изменение цели отличается от ее уточнения.

В чем, собственно, заключается изменение? Изменение — это любое внесение поправок в работу или ресурсы, способное нанести экономически ощутимый ущерб, опасно нарушить ход работ или значительно увеличить объем работ в пределах спринта. Как правило, изменение состоит в добавлении или удалении элемента задела продукта из спринта или значительном расширении состава элемента из задела продукта, уже находящегося в спринте. Следующий пример наглядно демонстрирует сущность изменения.

> **Владелец продукта:** "Когда я сказал, что мы должны организовать поиск в полицейской базе данных малолетнего преступника, я имел в виду не только имя и фамилию, но и возможность поиска в базе данных по рисунку татуировок на теле подозреваемого!"

Добавление возможности поиска по рисунку потребует значительно большего труда и скорее всего отрицательно скажется на способности команды выполнить взятое на себя обязательство реализовать поиск по имени и фамилии. В данном случае владелец продукта должен принять во внимание, что для реализации функционального средства поиска по рисунку в задел продукта придется ввести новый элемент и запланировать работу над ним в следующем спринте.

А в чем, собственно, заключается уточнение? Под уточнением понимаются дополнительные подробности, предоставляемые в течение спринта, чтобы помочь команде достичь намеченной цели спринта. Как будет показано в главе 5, в начале спринта могут быть известны не все подробности, связанные с элементами из задела спринта. Поэтому члены команды могут вполне обоснованно задавать в течение спринта уточняющие вопросы, на которые владельцу продукта придется отвечать. Приведенный ниже пример наглядно демонстрирует сущность уточнения. Подобным образом владелец продукта может и должен обеспечить уточнение цели спринта.

Команда разработчиков: "Когда вы сказали, что совпадения, полученные в результате поиска малолетнего преступника, должны отображаться списком, имели ли вы в виду, что такой список лучше было бы упорядочить?"

Владелец продукта: "Да, их нужно отсортировать в алфавитном порядке по фамилии".

Команда разработчиков: "Ладно, мы можем это сделать".

Последствия изменения цели спринта

На первый взгляд, правило, запрещающее изменять цель спринта, вступает в непосредственный конфликт с основным принципом Scrum непременно принимать изменения во внимание. Изменения, безусловно, следует принимать во внимание, но делать это следует взвешенно и экономически обоснованно. Экономические последствия изменения цели спринта возрастают по мере повышения уровня капиталовложений в изменившуюся работу (рис. 4.6).

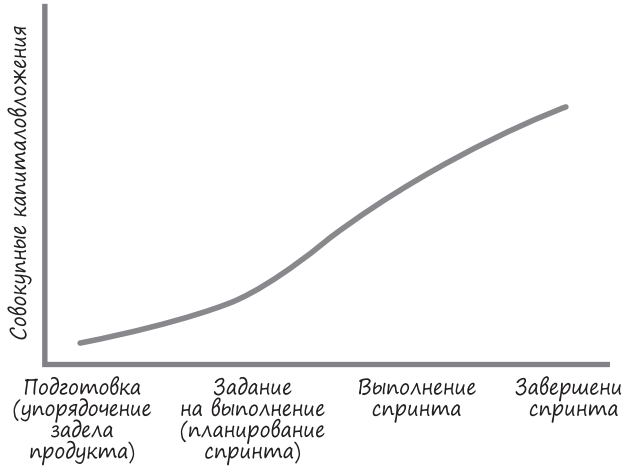


Рис. 4.6. Совокупные капиталовложения в разных состояниях

Мы вкладываем средства в элементы из задела продукта, чтобы подготовить их к работе в течение спринта. Но как только спринт начинается, наши капиталовложения в эти элементы возрастают, поскольку при планировании спринта мы тратим время на их обсуждение на уровне отдельных задач. Если же нам потребуется внести изменения после планирования спринта, мы не только рискуем нарушить планы, но и понести дополнительные расходы на перепланирование любых изменений в течение спринта.

Кроме того, как только мы приступим к выполнению спринта, наши капиталовложения в работу возрастают еще больше по мере прохождения элементов

из задела продукта через разные состояния: подготовки к работе (работа еще не начата), выполнения (работа не завершена) и готовности (работа завершена).

Допустим, что функциональное средство X, включенное в настоящее время в обязательство на спринт, требуется заменить функциональным средством Y, отсутствующим во взятом обязательстве. Даже если мы еще не начали работать над функциональным средством X, то все равно понесем убытки от планирования. Кроме того, у функционального средства X могут быть зависимости от других функциональных средств в спринте, и поэтому изменения в нем или полная его замена другим средством может повлиять на другие функциональные средства, а следовательно, и на конечную цель спринта.

Если же работа над функциональным средством X уже началась, то, помимо упомянутых выше убытков, мы можем понести и другие убытки. Например, вся работа, уже выполненная над функциональным средством X, может быть отвергнута полностью. Кроме того, мы можем понести дополнительные убытки от удаления потенциально завершенной работы над функциональным средством X, которым, возможно, так и не придется воспользоваться в дальнейшем. В частности, мы не собираемся включать частично завершенную работу в прирост потенциально готового к поставке продукта в конце спринта. И, разумеется, если функциональное средство X уже завершено, мы можем понести убытки, связанные с вложением средств в создание полностью готового функционального средства X. Все это только усугубляет убытки!

Помимо непосредственных экономических последствий от понесенных убытков, на экономический эффект может повлиять и возможное падение мотивации команды и доверия, которое может сопутствовать изменениям. Когда владелец продукта сначала берет на себя обязательство не изменять цель спринта, а затем нарушает взятое обязательство, то вполне естественно, что команда теряет мотивацию, что практически всегда отрицательно сказывается на желании ее членов прилежно работать над завершением элементов из задела продукта. Кроме того, нарушение обязательства может нанести вред атмосфере доверия, царящей в Scrum-команде, поскольку команда разработчиков потеряет доверие к желанию владельца продукта придерживаться взятых на себя обязательств.

Прагматичность

Запрет на изменение цели спринта — это всего лишь правило, а не закон, к которому Scrum-команда должна относиться прагматично. Что, если коммерческие условия изменятся таким образом, что внесение изменений в цель спринта покажется вполне оправданным? Допустим, что конкурент выпустит свой новый продукт во время спринта. В результате анализа этого нового продукта мы приходим к выводу, что нам придется изменить цель, постановленную

для текущего спринта, поскольку мы создаем в настоящий момент экономически менее ценный продукт, чем у конкурента. Стоит ли в таком случае слепо соблюдать правило запрета на изменение цели спринта? Вероятно, не стоит.

Что, если очень важная производственная система сильно нарушена и к ее наладке придется привлечь некоторых или даже всех членов команды? Следует ли прервать спринт, чтобы наладить производственную систему? Стоит ли говорить руководству организации, что в следующем спринте мы сначала наладим производство? Вероятно, не стоит.

В конечном счете здоровый прагматизм берет верх над соблюдением правила, запрещающего изменять цель спринта. Это способны оценить все члены Scrum-команды. Если внести изменения в текущий спринт, то они повлекут за собой отрицательные экономические последствия, как пояснялось выше. Но если экономические последствия изменений намного менее значительны, чем экономические последствия откладывания этих изменений, то их внесение считается разумным деловым решением. А если экономические последствия от изменений едва ли ощутимы по сравнению с их отсутствием, то изменять цель спринта не следует.

Что же касается мотивации и доверия, то, как показывает мой опыт, когда владелец продукта откровенно обсуждает с командой экономическую целесообразность и необходимость изменений, большинство членов команды лучше понимают и правильно оценивают подобную необходимость. Поэтому очень важно сохранять мотивацию вместе с доверием.

Ненормальное завершение спринта

Если цель спринта окажется совершенно неверной, Scrum-команда может прийти к решению, что продолжать текущий спринт не имеет смысла, и порекомендовать владельцу продукта преждевременно завершить спринт. Если текущий спринт завершается ненормально, он сразу же подходит к концу, и Scrum-команда собирается для проведения ретроспективы спринта. Для планирования следующего спринта она встречается с владельцем продукта, ставит другую цель и выбирает из задела продукта другой ряд элементов.

Спринт завершается преждевременно, если происходит важное с экономической точки зрения событие. Например, действия конкурента могут свести на нет цель спринта, а условия финансирования разработки продукта — существенно измениться.

Несмотря на то что владелец продукта резервирует возможность для отмены каждого спринта, как показывает мой опыт, владелец продукта редко пользуется такой возможностью. Зачастую Scrum-команда может принять менее решительные меры, чтобы исправить сложившееся положение. Напомним, что спринты

имеют короткую продолжительность, и в среднем команда оказывается на полпути к завершению спринта, когда возникает ситуация, побуждающая вносить в него изменения. Так, до конца спринта может остаться около недели, когда в него требуется внести изменения, и поэтому преждевременное завершение спринта может оказаться менее желательным, чем его продолжение. А нередко имеется возможность внести менее резкие изменения вроде прекращения разработки функционального средства с целью высвободить средства для устранения неполадок в важном производственном процессе вместо того, чтобы преждевременно прекращать спринт.

Очень важно понять, что преждевременное прекращение спринта оказывает не только отрицательное влияние на моральное состояние команды, но и серьезно нарушает быстрый и гибкий поток выпуска функциональных средств и сводит к минимуму многие преимущества упоминавшейся ранее постоянной продолжительности спринтов. К преждевременному прекращению спринта следует прибегать как к последнему средству.

Если спринт завершается преждевременно, Scrum-команде придется определить продолжительность следующего спринта (рис. 4.7).

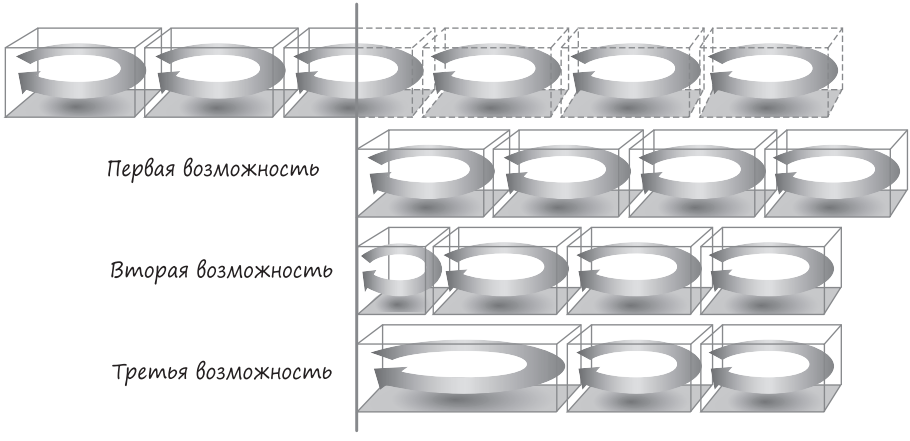


Рис. 4.7. Принятие решения относительно продолжительности следующего спринта после преждевременного завершения текущего спринта

Для этого имеются следующие очевидные возможности.

1. Оставить исходную продолжительность спринта. Это дает преимущество сохранить единообразную продолжительность спринта в течение всего процесса разработки, за исключением, разумеется, преждевременного прекращения спринта. Если несколько Scrum-команд сотрудничают над выполнением одних и тех же проектных работ, то вследствие того,

что используется первоначальная продолжительность спринта, Scrum-команда, преждевременно завершившая спринт, теряет синхронность в работе с другими командами.

2. Сделать следующий спринт достаточно продолжительным, чтобы достичь конечной даты преждевременно завершенного спринта. Так, если Scrum-команда преждевременно завершила двухнедельный спринт в конце первой недели, в следующем спринте ей потребуется около недели, чтобы снова войти в первоначальный размеренный ритм работы в спринте.
3. Сделать следующий спринт более продолжительным, чем обычный спринт, чтобы охватить время, оставшееся в преждевременно завершившемся спринте, плюс время, отводимое на следующий полноценный спринт. Таким образом, в предыдущем примере следующий спринт можно сделать продолжительностью три недели, чтобы восстановить исходный размеренный ритм выполнения спринта.

Если проектные работы выполняются несколькими командами, то выбрать лучше вторую или третью возможность. Но в любом случае придется принять во внимание конкретный контекст, чтобы выяснить, какую именно возможность следует выбрать.

Критерий готовности

Как пояснялось в главе 2, в результате каждого спринта должен быть получен прирост потенциально готового к поставке продукта. Там же упоминалось, что термин “потенциально готовый к поставке” совсем не означает, что разработанный продукт или его часть должны быть фактически поставлены. Поставка — это коммерческое решение, которое нередко принимается в разном темпе. В некоторых организациях может оказаться нецелесообразным поставлять продукт в конце каждого спринта.

Понятие “потенциально готовый к поставке” лучше описывает состояние уверенности в том, что продукт или функциональные средства, которые должны были быть разработаны в спринте, фактически готовы. Это означает отсутствие ощутимого объема незавершенной работы (например, важного тестирования или интеграции), которую нужно завершить, прежде чем поставить результаты спринта, если поставка является коммерческой целью. Чтобы определить, является ли разработанный продукт или функциональное средство потенциально готовым к поставке, Scrum-команда должна руководствоваться вполне определенным, согласованным критерием готовности.

Что такое критерий готовности

В принципе *критерий готовности* представляет собой контрольный список видов работ, которые команда предполагает успешно завершить, прежде чем объявить результаты своей работы потенциально готовыми к поставке (табл. 4.1).

Таблица 4.1. Пример контрольного списка по критерию готовности

Критерий готовности
❑ Проект проанализирован
❑ Код завершен
❑ Код реорганизован
❑ Код в стандартном формате
❑ Код снабжен комментариями
❑ Код проверен
❑ Код обследован
❑ Документация для конечного пользователя обновлена
❑ Протестировано
❑ Проведено модульное тестирование
❑ Проведено комплексное тестирование
❑ Проведено регрессивное тестирование
❑ Проведено тестирование платформы
❑ Проведено тестирование языка
❑ Дефекты не обнаружены
❑ Проведено приемочное тестирование
❑ Действует на рабочих серверах

Очевидно, что отдельные элементы контрольного списка будут зависеть от количества следующих факторов.

- Характер разрабатываемого продукта.
- Технологии, применяемые для его разработки.
- Организация, разрабатывающая продукт.
- Текущие препятствия, мешающие достижению того, чего можно было бы достигнуть.

Чаще всего минимальный критерий готовности должен определять готовую часть разработанных, построенных, интегрированных, протестированных и документированных функциональных возможностей программного продукта,

способных доставить потребителю утвержденную ценность. Но для составления полезного контрольного списка виды работ, перечисляемые на верхнем уровне, должны быть дополнительно уточнены. Например, что означает тестирование вообще, модульное и комплексное тестирование в частности, тестирование системы, платформы и интернационализации? Можно, конечно, придумать и много других форм тестирования специально для разрабатываемого программного продукта. Но стоит ли включать все эти виды тестирования в критерий готовности?

Следует иметь в виду, что если важный вид тестирования происходит не в каждом спринте (например, тестирование производительности), то его следует выполнять хотя бы иногда. Так, если в будущем предполагается выделить один из спринтов только для тестирования производительности, которое должно быть непременно выполнено, чтобы считать продут действительно готовым, то прирост потенциально готового к поставке продукта не происходит в каждом спринте. Хуже того, если тестирование производительности выполняется позднее и не по плану, то далее в процессе не только обнаружится серьезное затруднение, но и придется потратить намного больше времени и средств, чтобы исправить данное затруднение, чем в том случае, если бы тестирование производительности выполнялось раньше.

Иногда тестирование может отнять больше времени, чем отведено на спринт. Если это происходит потому, что команда разработчиков накопила большой технический долг по тестированию, ей придется автоматизировать свои тесты, чтобы выполнять тестирование в течение спринта. А если это происходит из-за характера самого теста, то придется мириться с тем, что такой тест будет начинаться в одном спринте и завершаться в другом. Например, одна организация, где я проводил обучение, проектировала устройство, состоявшее из аппаратных средств, встроенного и прикладного программного обеспечения. Один из стандартных тестов в этой организации состоял в отбраковочном испытании данного устройства в течение 1500 часов на выявление отказа. Такой тест нельзя было выполнить в течение одного двухнедельного спринта, и поэтому Scrum-команда решила внести коррективы в критерий готовности, чтобы спринт можно было считать завершенным, несмотря на то что 1500-часовой тест еще не завершен.

Меня нередко спрашивают: “Если существенный дефект остается в последний день спринта, то можно ли считать готовым соответствующий элемент из задела продукта?” Нет, нельзя! Как правило, спринты не выходят за спланированные заранее временны*е* рамки, и поэтому текущий спринт нельзя продлить на один или два дня, чтобы устранить дефект в этом спринте. Вместо этого по окончании спринта в запланированный срок незавершенный элемент задела продукта переносится из текущего спринта обратно в задел продукта в надлежащем порядке с учетом других элементов, которые в настоящий момент находятся в заделе продукта. Незавершенный элемент может быть затем завершен в последующем спринте.

Scrum-команды должны иметь надежное определение критерия готовности, обеспечивающее высокую степень уверенности в том, что разработанный продукт или его часть обладает высоким качеством и подлежит поставке. Отсутствие такой уверенности лишает организацию возможности поставлять продукт по своему усмотрению и может привести к накоплению технического долга, как поясняется в главе 8.

Возможное развитие критерия готовности во времени

Критерий готовности можно рассматривать как способ определения состояния работ в конце спринта. Для многих высокопроизводительных команд целевое конечное состояние работ позволяет судить о потенциальной готовности к поставке. И это состояние остается относительно постоянным в течение всего срока разработки.

Например, когда я выполнял роль владельца продукта в проекте переделки веб-сайта организации Scrum Alliance в 2007 году, мы выполняли работы в течение недельных спринтов. Конечное состояние по нашему критерию готовности можно было подытожить как “действует на рабочих серверах”. Мы вместе с командой разработчиков пришли к выводу, что это совершенно обоснованное состояние для достижения цели каждого спринта. Это конечное состояние мы определили в самом начале проектных работ и не меняли его все время, пока я был владельцем продукта для веб-сайта.

Но многие команды начинают работать с критерием готовности, который не оканчивается на состоянии, где все функциональные средства завершены до такой степени, что они способны действовать или быть поставленными. Существуют определенные препятствия, мешающие некоторым командам достичь такого состояния в начале разработки, хотя оно и является их конечной целью. В итоге они могут (по необходимости) начать с меньшего конечного состояния, допуская дальнейшее развитие критерия готовности во времени по мере устранения организационных препятствий.

Например, я посещал организацию, занимавшуюся разработкой клинических информационных систем. Ее продукт устанавливался в поликлинике и накапливал разнообразную клиническую информацию (иногда даже непосредственно из оборудования, выполнявшего диагностические тесты). Команде разработчиков было известно, что клинические испытания, подразумевавшие установку программного продукта в клинической лаборатории с целью убедиться в его работоспособности на клиническом оборудовании, потребуется провести прежде, чем поставлять продукт. Но поскольку у команды не было регулярного доступа в лабораторию, они не включили поначалу клиническое испытание в свой критерий готовности. Вместо этого они решили проводить спринты клинических испытаний по завершении каждого выпуска.

Из обсуждения я узнал, что в отделе маркетинга и команде разработчиков терпеть не могли эти клинические испытания перед выпуском, поскольку никто не мог предвидеть, сколько спринтов потребуется для устранения всех дефектов, а продукт нельзя было выпустить, не устранив дефекты. Для коллективной выработки возможных решений мы привлекли технического директора. И тогда он задал команде следующий вопрос: "Если бы у вас был доступ в лабораторию, смогли бы вы проводить клинические испытания в каждом спринте?"

Обсудив этот вопрос, члены команды нашли на него следующий ответ: "Да, могли бы, но это означало бы, что мы завершали бы в каждом спринте меньше функциональных средств". Технический директор согласился устранить препятствие, организовав доступ команды в клиническую лабораторию местного университета. Он также согласился с тем, что завершение меньшего количества функциональных средств в каждом спринте было бы разумным компромиссом ради того, чтобы доставляемые функциональные средства прошли клинические испытания. В тот момент команде удалось развить свой критерий готовности, чтобы добиться состояния потенциальной готовности к поставке, и благодаря этому у всех появилось больше уверенности в том, что намеченная работа будет завершаться в каждом спринте.

Иногда команде может быть известно препятствие, которое она просто не в состоянии устранить сразу. Таким образом, ей известно, что критерий готовности в течение проектных работ по необходимости будет развиваться. В качестве характерного примера может служить разработка продукта, состоящего из аппаратных и программных средств. Мне не раз приходилось видеть применение методики Scrum в разработке многих подобных продуктов, и зачастую я слышал от разработчиков программного обеспечения следующее: "Оборудование всегда поступает с опозданием!" В подобных случаях у команды, занимающейся разработкой программного обеспечения, отсутствует оборудование, на котором они могли бы тестировать свое программное обеспечение, а следовательно, команда не может заявить, что результаты, достигнутые в конце спринта, потенциально готовы к поставке. В лучшем случае она может заявить о готовности для эмулятора, поскольку тестирование в течение первых спринтов обычно выполняется на программном эмуляторе конкретного оборудования. А когда это оборудование появится, критерий готовности получит дальнейшее развитие в направлении потенциальной готовности или хотя бы чего-нибудь близко ее напоминающего.

Критерии готовности и приемки

Критерий готовности применяется к приросту продукта, разрабатываемого в течение спринта. Прирост продукта состоит из ряда элементов задела продукта, и поэтому каждый такой элемент должен быть завершен в соответствии с видами работ, перечисленными в контрольном списке критерия готовности.

Как поясняется в главе 5, каждый элемент, переносимый из задела продукта в спринт, должен иметь ряд *условий удовлетворения* (т.е. критерии приемки этого элемента), определяемых владельцем продукта. Эти *критерии приемки* в конечном итоге проверяются в ходе *приемочных испытаний*, в ходе которых владелец продукта может убедиться, что элемент задела спринта функционирует должным образом. Так, если элемент задела продукта должен позволять заказчику оплачивать покупки кредитной карточкой, то условия удовлетворения должны состоять в возможности работать с кредитными карточками AmEx, Visa и MasterCard. Таким образом, у каждого элемента из задела спринта имеются свои критерии приемки. Эти характерные для него критерии дополняют, но не подменяют собой критерии готовности, определяемые в контрольном списке на все элементы из задела продукта.

Элемент задела продукта может считаться готовым только в том случае, если удовлетворяются характерные для него критерии приемки (например, возможность работать со всеми видами кредитных карточек), а также отдельные критерии готовности на уровне спринта (например, способность действовать на рабочем сервере) из контрольного списка. Если элементы задела продукта, удовлетворяющие их критериям приемки, неудобно называть *готовыми*, их можно обозначать как *завершенные* или *принятые*.

Критерий готовности и неполная готовность

В некоторых командах принято употреблять понятие *готовности* в сравнении с *неполной готовностью*. Неполная готовность означает, что предполагается сделать больше, чем сделано. Употреблять оба этих разных понятия в командах совсем не обязательно, но следует признать, что я сам пользуюсь ими, когда проверяю домашнюю работу своего сына. Когда я спрашиваю, сделал ли он свое домашнее задание, он обычно отвечает утвердительно. А когда я спрашиваю у учительницы на родительском собрании в школе, сдает ли мой сын домашнюю работу готовой на проверку, она отвечает, что его домашняя работа совсем не готова!

В результате бесед с моим сыном с целью установить истину я пришел к выводу, что он пользуется следующим критерием готовности: “Я выполнил домашнюю работу настолько, насколько был готов к ней!” И с этого момента я стал пользоваться термином *неполная готовность*, который, как мы с ним согласились, означает, что домашняя работа сделана настолько, насколько учительница посчитает ее готовой.

Команды не привыкли добиваться полной готовности рано и зачастую опираются на неполную готовность, как на костыли. Они пользуются понятием неполной готовности, чтобы отличать состояние полной готовности (т.е. столько

работы, сколько они готовы сделать) от состояния неполной готовности (т.е. столько работы, сколько требуется сделать, чтобы заказчики посчитали ее готовой). А тем командам, которые усвоили, что полной готовности можно достичь только в том случае, если сделать всю работу, необходимую для того, чтобы удовлетворить заказчиков, оба эти состояния готовности не нужны. Им все едино: что полная, что неполная готовность!

Заключение

В этой главе была рассмотрена решающая роль спринтов в инфраструктуре Scrum. Спринты служат главным остовом Scrum, на который могут опираться все остальные виды деятельности и артефакты в Scrum. Спринты имеют короткую, ограниченную по времени и постоянную продолжительность. Как правило, они определяются своей целью, которую нельзя изменять без особых на то экономических оснований. В результате спринтов должен быть получен прирост потенциально готового к поставке продукта в соответствии с согласованным критерием готовности. А в следующей главе основное внимание будет уделено исходным данным для спринта — требованиям и их общим представлением в виде пользовательских историй.

Глава 5
Требования и пользовательские истории

В этой главе обсуждаются отличия Scrum-проекта от традиционного проекта в трактовке требований. В этом контексте описывается назначение пользовательских историй в качестве общей формы представления элементов коммерческой ценности. Сначала основное внимание в этой главе уделяется сущности пользовательских историй, их способности представить коммерческую ценность на разных уровнях абстракции и критериям оценки пригодности пользовательских историй. Затем поясняется, как обращаться с нефункциональными требованиями и организовать работу по приобретению знаний в Scrum-проекте. И в завершение подробно рассматриваются две методики сбора пользовательских историй.

Краткий обзор

Требования трактуются в Scrum совершенно иначе, чем в последовательной методике разработки продукции. При последовательном подходе к разработке продукции требования не обсуждаются, уточняются заранее и рассматриваются отдельно. А в Scrum требования подробно рассматриваются при обсуждениях, которые постоянно происходят в течение разработки и конкретизируются *своевременно* и в *достаточной степени*, чтобы команды могли приступить к разработке функциональных средств, отвечающих этим требованиям.

При последовательной разработке продукции требования трактуются во многом так же, как и в производстве: они представляют собой обязательные, не обсуждаемые спецификации, которым должна соответствовать продукция. Такие требования составляются заранее и передаются команде разработчиков в форме подробнейшего документа, а задача команды разработчиков — произвести продукцию, соответствующую подробным требованиям.

Если требуется внести изменения в первоначальный план, это делается в формальном процессе контроля изменений. А поскольку главная цель — соответствие спецификациям, то подобные отклонения нежелательны и недешевы. Ведь бо́льшую часть незавершенных работ в форме подробнейших требований (и всей основанной на них работы) придется переделать или отвергнуть.

С другой стороны, требования в Scrum рассматриваются как важная степень свободы, которая позволяет удовлетворить коммерческим целям. Так, если мы исчерпаем время или средства, то можем опустить малоценные требования. И если во время разработки появится новая информация, указывающая на то, что требование стало намного менее подходящим в отношении затрат и выгоды, то мы можем убрать его из продукта. А если появится новое очень ценное требование, его можно добавить в продукт, отвергнув, вероятнее всего, менее ценное требование, чтобы освободить место.

Многим из нас, вероятно, приходилось составлять документ с "полными" требованиями в начале разработки, а впоследствии обнаруживать отсутствие в нем важного требования. Когда обнаруживается отсутствие такого требования, нередко возникает следующий характерный для данной ситуации диалог:

> **Заказчик:** "Теперь, когда я вижу эти построенные функциональные средства, я понимаю, что мне нужно еще одно средство, отсутствующее в техническом задании".
>
> **Разработчики:** "Если вам потребовалось это функциональное средство, то почему же вы не указали его заранее?"
>
> **Заказчик:** "Я не осознавал, что это средство мне было нужно, до тех пор, пока не увидел продукт в собранном виде".
>
> **Разработчики:** "Если бы вы дольше и тщательнее продумали требования заранее, то обнаружили бы потребность в этом функциональном средстве уже тогда, а не теперь".

Дело в том, что, разрабатывая передовую продукцию, невозможно составить заранее полные требования или проектные решения, даже если работать над ними дольше и прилежнее. Некоторые требования и проектные решения всегда появляются в процессе разработки продукции, и никакая тщательная подготовительная работа не способна этому воспрепятствовать.

Следовательно, применяя методику Scrum, мы не должны уделять много времени и средств конкретизации требований заранее. Если мы предполагаем, что требования могут со временем измениться по мере того, как мы больше узнаем о том, что разрабатываем, то тем самым мы исключаем чрезмерные затраты времени и средств на составление требований, которые можем впоследствии отвергнуть. Вместо того чтобы накапливать крупные запасы требований заранее, мы создаем для требований заменители, называемые *элементами задела продукта*. Каждый элемент задела продукта представляет желательную коммерческую ценность (рис. 5.1).

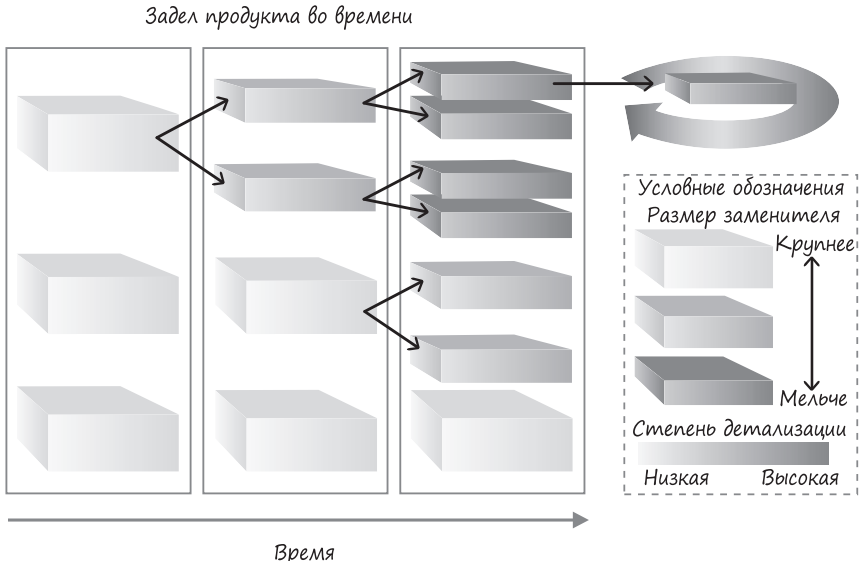


Рис. 5.1. Заменители требований, применяемые в Scrum

Первоначально элементы задела продукта оказываются крупными (на рис. 5.1 они обозначены крупными блоками), и с ними связано мало подробностей. Со временем эти элементы проходят через обсуждения между участниками проекта, владельцем продукта и командой разработчиков, которые уточняют их, превращая в совокупность более мелких, но и более детализированных элементов задела продукта. В конечном итоге элемент задела заменителя оказывается достаточно мелким и детализированным для перемещения в спринт, где он разрабатывается, строится и тестируется. Но даже в течение спринта в обсуждениях между владельцем продукта и командой разработчиков выявляются дополнительные подробности. Как обсуждается в главе 6, задел продукта представляет собой моментальный список текущей совокупности элементов задела продукта и связанных с ними подробностей.

Несмотря на то что в Scrum не предусмотрена стандартная форма элементов задела продукта, многие команды представляют эти элементы как пользовательские истории, что неверно. Одни команды предпочитают прецеденты использования, а другие — собственные формы представления элементов задела продукта.

В этой книге пользовательские истории выбраны в качестве основного средства для представления элементов задела продукта. Более подробно пользовательские истории рассматриваются далее в этой главе. Но даже если вы выберете какую-нибудь другую форму представления элементов задела продукта, обсуждение здесь пользовательских историй окажется вам полезным для лучшего понимания тех характеристик, которыми должно обладать то или иное представление подобных элементов.

Проведение обсуждений

В качестве средства общения требования облегчают общее понимание того, что требуется разработать. Они дают возможность тем, кто понимает, что именно следует создать, ясно передать свои пожелания тем, кому предстоит это создать.

Методика последовательной разработки продукции в значительной мере опирается на составленные в письменном виде требования, которые могут быть истолкованы неверно, хотя и выглядят впечатляюще. Мне вспоминается разговор с вице-президентом, отвечавшим за выпуск продукции в компании, которую я посетил. Я спросил у него, кто руководит всеми специалистами по анализу экономической деятельности компании и как они обращаются с требованиями. В ответ он привел следующий пример: "С первого января моя команда организует проектные работы по техническому заданию, а 31 декабря мы демонстрируем результаты, чтобы посмотреть, чего добились".

Тогда я спросил его, кому из его команды поручено отвечать в течение года на вопросы разработчиков и уточнять требования. Он ответил: "Никому. Все время, отведенное моей команде на данный проект, было потрачено на составление технического задания. Мои аналитики заняты составлением технического задания для других проектов. Но не волнуйтесь, мы составили качественный документ, и разработчики и тестировщики найдут в нем ответы на любые вопросы, внимательно прочитав его".

Маловероятно, чтобы неясности отсутствовали в этом документе на 150 страниц, подробно описывающем прецеденты использования для разработки новой электронной системы медицинского учета. Английский язык не позволяет настолько точно выразить мысль, чтобы избежать неясности и двусмысленности. Но даже если бы он был способен на это, то сами люди не всегда умеют точно выразить мысль в письменном виде. Для того чтобы функциональные средства были построены должным образом, тем, кому они требуются, намного полезнее проводить своевременные обсуждения с теми, кто разрабатывает, строит и тестирует эти средства.

Обсуждения в Scrum служат главным средством для надлежащего обсуждения и сообщения требования. Речевое общение обладает тем преимуществом, что дает быструю реакцию, упрощая и удешевляя достижение общего взаимопонимания. Кроме того, обсуждения способствуют двухстороннему общению, способному порождать идеи по поводу разрешения возникающих затруднений и выявления новых возможностей. Подобные дискуссии вряд ли могут возникнуть в результате чтения документа.

Но обсуждение — это всего лишь средство и не заменяет собой всю документацию. Задел продукта в Scrum является "живым документом", постоянно доступным в течение разработки продукции. Те, кому требуются и нужны

требования, оформленные в виде документа, могут просто составить такой документ из элементов задела продукта и всех связанных с ними подробностей в любом формате и в какой угодно момент.

Постепенное уточнение требований

При последовательном подходе к разработке продукции все требования должны быть детализированы на одном и том же уровне. В частности, каждое требование должно быть указано в утвержденном техническом задании отдельно, чтобы команды, выполняющие работы по разработке, построению и тестированию, смогли понять, как удовлетворить поставленным техническим требованиям. Никакие дополнительные подробности не должны быть оставлены без внимания.

В то же время стремление составить сразу все требования на одном и том же уровне детализации чревато многими недостатками, в том числе следующими.

- Все подробности необходимо предусмотреть на как можно более ранней стадии разработки, когда известно минимум из того, что можно узнать о разрабатываемой продукции.
- Все требования трактуются одинаково независимо от их приоритетности, что вынуждает выделять ценные ресурсы на детализацию требований, которые могут быть вообще не воплощены в продукции.
- Накапливаются крупные запасы требований, которые, вероятнее всего, будут отвергнуты, если изменятся обстоятельства, поскольку их переделка обойдется очень дорого.
- Снижается вероятность проведения обсуждений с целью выработать и уточнить требования, поскольку они уже считаются “полными”.

Как показано на рис. 5.1, не все требования в Scrum должны одновременно обладать одинаковым уровнем детализации. Требования, над которыми придется работать раньше, окажутся более мелкими и детализированными, чем те требования, работу над которыми придется отложить на некоторое время. Для своевременного разбиения крупных слабо детализированных требований на ряд мелких более детализированных требований применяется стратегия *постепенного уточнения*.

Что такое пользовательские истории

Пользовательские истории служат удобной формой выражения требующейся коммерческой ценности для многих видов элементов из задела продукта, особенно функциональных средств. Пользовательские истории вырабатываются таким

образом, чтобы быть понятными как деловым людям, так и техническим работникам. Оно должны иметь простую структуру и служить удобным заменителем обсуждения. Кроме того, пользовательские истории могут быть написаны с разной степенью подробности и должны легко допускать постепенное уточнение.

Как бы хорошо пользовательские истории ни были приспособлены к потребностям разработки, их не следует считать единственным способом представления элементов задела продукта. Они обеспечивают упрощенный подход, вполне согласующийся с базовыми принципами гибкой разработки и потребностями в эффективном и эффектном заменителе. Я лично пользуюсь ими в качестве главного заменителя, который дополняю другой информацией, уместной и полезной для конкретизации требования. Если же я обнаруживаю, что пользовательские истории придется подгонять под конкретную ситуацию (например, для представления некоторых дефектов), то выбираю другой подход. Так, однажды я наблюдал за тем, как команда писала следующую пользовательскую историю: "Мне как заказчику требуется, чтобы система не портила базу данных". Согласитесь, что для представления подобного требования пользовательская история — не самое лучшее средство. Вероятно, было бы проще и уместнее сделать ссылку на дефект в системе обнаружения неисправностей.

Так что же представляют собой пользовательские истории? Рон Джеффриз (Ron Jeffries) предлагает простой, но эффективный способ истолкования пользовательских историй [Jeffries, Ron. 2001]. Он описывает их следующими тремя понятиями: карточка, обсуждение и подтверждение.

Карточка

Принцип действия карточки довольно прост. Первоначально пользовательские истории писали (и до сих пор пишут) прямо на карточках для записей или бумажных наклейках размерами 8×13 см (рис. 5.2).

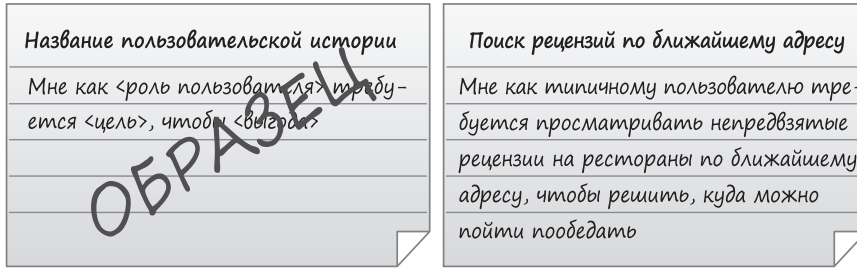


Рис. 5.2. Образец карточки для пользовательской истории

Типичная форма образца для написания пользовательских историй, как показано на рис. 5.2, *слева*, содержит указание категории пользователя (роль

пользователя), чего, собственно, пользователь данной категории хочет добиться (цель), а также причины для достижения поставленной цели (выгода) [Cohn, Mike. 2004]. Часть пользовательской истории, начинающаяся со слова "чтобы", является дополнительной, но не обязательной, но она обычно включается практически в каждую пользовательскую историю, кроме тех случаев, когда назначение истории совершенно очевидно.

Карточка не предназначена для фиксации всей информации, составляющей требование. В действительности мы намеренно пользуемся мелкими карточками с ограниченным местом для записей, чтобы способствовать краткости изложения пользовательских историй. Карточка должна содержать несколько предложений, ясно передающих суть или цель требования. Она служит в качестве заменителя более подробных дискуссий, происходящих между участниками проекта, владельцем продукта и командой разработчиков.

Обсуждение

Подробности требования раскрываются и сообщаются в ходе обсуждения, происходящего между командой разработчиков, владельцем продукта и прочими участниками проекта. А пользовательская история — это всего лишь обещание провести такое обсуждение.

На самом деле обсуждение является не одномоментным событием, а непрерывным диалогом. Первоначально обсуждение может состояться, когда пользовательская история пишется; другое обсуждение — когда она уточняется; очередное обсуждение — когда она оценивается; еще одно обсуждение — во время планирования спринта, когда команда вдается в подробности на уровне отдельных задач; и наконец, непрерывные обсуждения происходят по ходу разработки, построения и тестирования пользовательской истории в течение спринта.

Одно из преимуществ пользовательских историй заключается в том, что они отчасти смещают акцент с написания на обсуждение. Подобные обсуждения служат более богатой формой обмена информацией и сотрудничества с целью правильно выразить требования и сделать их понятными каждому.

Несмотря на то что обсуждения происходят, главным образом, в речевой форме, они нередко могут дополняться документами. Так, в результате обсуждений может быть составлен эскиз пользовательского интерфейса или выработаны бизнес-правила, которые затем записываются в письменной форме. Однажды я посетил организацию, разрабатывавшую программное обеспечение для обработки медицинских снимков. Одна из обсуждавшихся при этом пользовательских историй приведена на рис. 5.3. Обратите внимание на то, что в данной пользовательской истории делается ссылка на целую статью для последующего чтения и обсуждения.

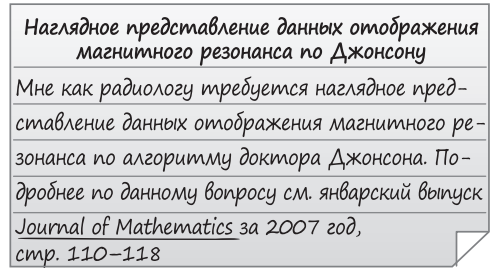


Рис. 5.3. Пользовательская история со ссылкой на дополнительную информацию

Таким образом, мы не отказываемся от всех наших документов ради пользовательских историй и связанных с ними карточек. Пользовательские истории удобны в качестве отправной точки для выявления первоначальной сути требований и в качестве напоминания о необходимости более подробно обсудить требования, когда это уместно. Но пользовательские истории могут и должны быть дополнены любой информацией в письменном виде, которая помогает прояснить требования.

Подтверждение

Пользовательская история содержит также подтверждающую информацию в форме условий ее удовлетворения. Это критерии приемки, проясняющие требуемое поведение. Команда разработчиков руководствуется ими, чтобы лучше понять, что именно требуется разработать, построить и протестировать, а владелец продукта, чтобы подтвердить, что пользовательская история реализована и вполне удовлетворяет его требованиям. Если на лицевой стороне карточки находится описание в несколько строк пользовательской истории, то на тыльной ее стороне можно указать условия удовлетворения данной истории (рис. 5.4).

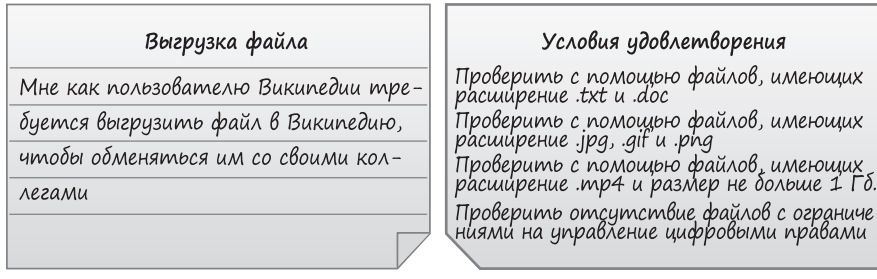


Рис. 5.4. Условия удовлетворения пользовательской истории

Подобные условия удовлетворения могут быть выражены в виде приемочных тестов высокого уровня. Но это не единственные тесты, которые выполняются в процессе разработки пользовательской истории. В действительности для дюжины приемочных тестов, связанных с пользовательской историей, команде придется выполнить намного (возможно, в 10 или 100 раз) больше тестов на детализированном техническом уровне, о чем владелец продукта может вообще не знать.

Приемочные тесты, связанные с пользовательской историей, требуются по нескольким причинам. Прежде всего, они позволяют владельцу продукта выяснить, насколько верно реализована пользовательская история.

Подобные тесты могут быть также полезны для первоначального составления историй и последующего их уточнения по мере выяснения дополнительных подробностей. Такой подход иногда еще называют *спецификацией по образцу* или *разработкой посредством приемочного тестирования* (ATTD). В его основу положен следующий вполне понятный принцип: обсуждения историй могут быть сосредоточены, а зачастую так и происходит, на определении конкретных примеров или требуемых видов поведения. Например, в пользовательской истории с выгрузкой файлов (см. рис. 5.4) обсуждение, скорее всего, велось следующим образом.

> Ограничим сначала размеры выгружаемых файлов величиной не более 1 Гбайт. Убедимся также в правильности выгрузки на примерах типичных текстовых и графических файлов. И на правовых основаниях мы не можем допустить выгрузку в Википедию любых файлов с ограничениями на управление цифровыми правами (DRM).

Если воспользоваться таким инструментальным средством, как Fit или FitNesse, то результаты подобных тестов удобно свести в таблицу. В качестве примера в табл. 5.1 приведены разные размеры файлов и результаты проверки их достоверности. Опираясь на подобные конкретные примеры, можно постепенно продвигаться к цели в процессе составления и уточнения пользовательских историй, имея в своем распоряжении автоматизированные приемочные тесты для каждой пользовательской истории.

Таблица 5.1. Результаты автоматизированного тестирования

Размер файла	Достоверность
`0`	Истинно
`1073741824`	Истинно
`1073741825`	Ложно

Уровень детализации

Пользовательские истории служат отличным средством для переноса элементов потребительской или пользовательской ценности через поток создания ценности в Scrum. Но если имеется только один размер истории, удобно вписывающийся в краткосрочный спринт, то это затруднит планирование на более высоком уровне и не даст возможности воспользоваться преимуществами постепенного уточнения.

Мелкие истории, реализуемые на уровне спринта, слишком малы и многочисленны, чтобы поддерживать планирование и выпуск продукции на более высоком уровне. На подобном уровне требуется меньшее количество элементов, которые менее детализированы и более абстрактны. В противном случае возникает риск увязнуть в болоте совершенно неуместных подробностей. Трудно, например, себе представить, что делать с 500 мелкими историями, если руководство попросит составить описание предлагаемой продукции, чтобы обеспечить финансирование ее разработки, или попытаться разделить эти 500 мелких элементов по приоритетам, чтобы определиться со следующим выпуском.

Кроме того, при наличии только одного (малого) размера истории придется определить все требования на очень мелком уровне детализации задолго до того, как это действительно потребуется. Наличие мелких историй препятствует постепенному уточнению требований своевременно и в достаточной степени. Правда, пользовательские истории можно писать таким образом, чтобы фиксировать потребности заказчиков и пользователей на различных уровнях абстракции (рис. 5.5).

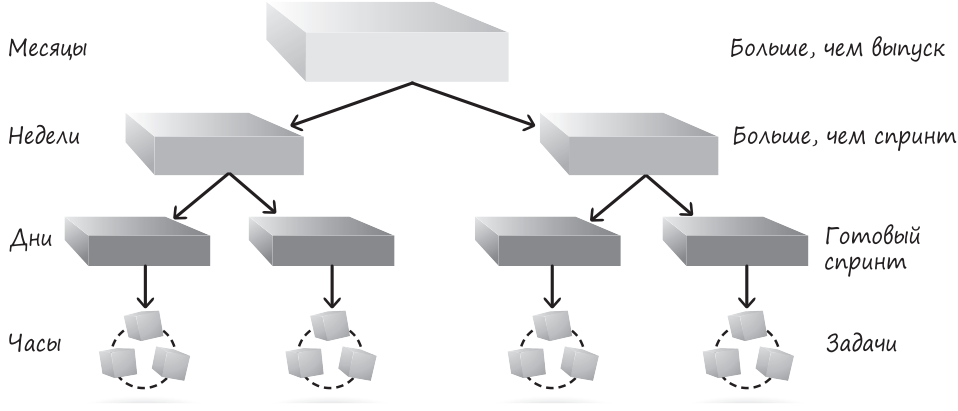


Рис. 5.5. Иерархия абстракции пользовательских историй

На рис. 5.5 пользовательские истории показаны на нескольких уровнях абстракции. Самые крупные истории малочисленны, длятся месяцами и могут охватывать один и даже несколько выпусков. Поэтому многие называют их *эпическими*, намекая на их размеры, сопоставимые с “Илиадой” и “Одиссей” Гомера.

Эпические истории удобны потому, что они дают очень крупную картину и самое общее представление о том, что требуется разработать (рис. 5.6).

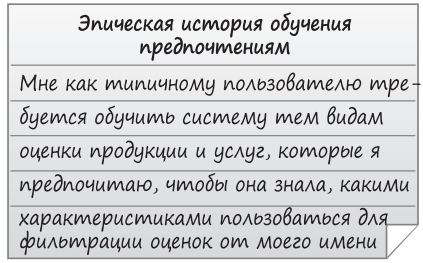


Рис. 5.6. Пример эпической истории

Эпическую историю вообще нельзя вместить в спринт, поскольку она слишком крупная и детализированная для разработки в столь краткие сроки. Поэтому эпические истории отлично заменяют крупные совокупности более детализированных историй, которые предстоит в свое время реализовать в будущем. Пример применения эпических историй будет продемонстрирован при обсуждении планирования продукции в главе 17. Следующие по размеру истории на рис. 5.5 зачастую длятся неделями, и поэтому они слишком велики для одного спринта. В некоторых командах разработчиков они могут называться *функциональными средствами*.

Наименьшие формы пользовательских историй обычно называют просто *историями*. Во избежание путаницы с эпическими историями, функциональными средствами и другими крупными элементами, которые также считаются "историями", некоторые называют эти мелкие истории *реализуемыми* или *поддающимися спринту*, указывая на то, что они длятся днями и достаточно малы, чтобы их можно было реализовать в течение спринта. Пример поддающейся спринту истории приведен на рис. 5.2.

В некоторых командах разработчиков употребляется также термин *тема* для обозначения совокупности связанных вместе историй. Темы позволяют удобно связать вместе истории, имеющие нечто общее, например, одинаковую функциональную область. На рис. 5.7 показана тема, представляющая совокупность историй с подробностями обучения ключевым словам.

Я нередко рассматриваю тему как стопку сложенных вместе и стянутых резинкой карточек для заметок. Под этим подразумевается, что они похожи друг на друга в той области, в которой считаются важными.

На расположенном ниже уровне находятся задачи, над которыми обычно работают один или два человека. Как правило, задачи выполняются в течение нескольких часов. Дойдя на уровня задач, мы указываем, *как* создавать, а не *что* создавать, поскольку для этого служат эпические истории, функциональные средства и просто истории. Задачи не являются историями, и поэтому составляемые истории не следует конкретизировать до уровня задач.

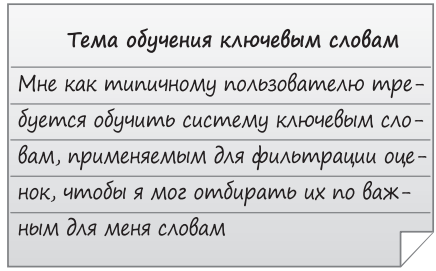


Рис. 5.7. Пример темы

Следует, однако, иметь в виду, что термины *эпическая история*, *функциональное средство*, *история* и *тема* служат лишь для удобства и не являются общепринятыми. На самом деле не так важно, какие именно термины употреблять, если пользоваться ими постоянно. Намного важнее понимать, что истории могут существовать на нескольких уровнях абстракции и что это обстоятельство способствует планированию на нескольких уровнях абстракции и постепенному уточнению крупных элементов доведением их со временем до мелких элементов.

Критерии INVEST оценки пригодности историй

Как же узнать, насколько пригодны написанные нами истории? Для этой цели Билл Уэйк (Bill Wake) предложил шесть критериев под общим сокращением INVEST. Они оказались удобными для следующей оценки: пригодны ли истории для предполагаемого использования, или же они требуют доработки [Wake, William C. 2003].

Критерии **INVEST** означают, что история должна быть *независимой* (**I**ndependent), *обсуждаемой* (**N**egotiable), *ценной* (**V**aluable), *оцениваемой* (**E**stimatable), *мелкой* (**S**mall; т.е. она должна иметь подходящий размер) и *тестируемой* (**T**estable). На основании информации, получаемой по каждому из этих критериев, можно составить общее представление о тех дополнительных изменениях, которые требуется внести в историю, если это вообще придется делать. Рассмотрим каждый из этих критериев по отдельности.

Независимость

С практической точки зрения пользовательские истории должны быть *независимы* или, по крайней мере, слабо связаны друг с другом. Те истории, которые проявляют высокую степень взаимозависимости, усложняют оценивание, назначение приоритетов и планирование. В качестве примера на рис. 5.8, *слева*, показана история **#10**, которая зависит от многих других историй.

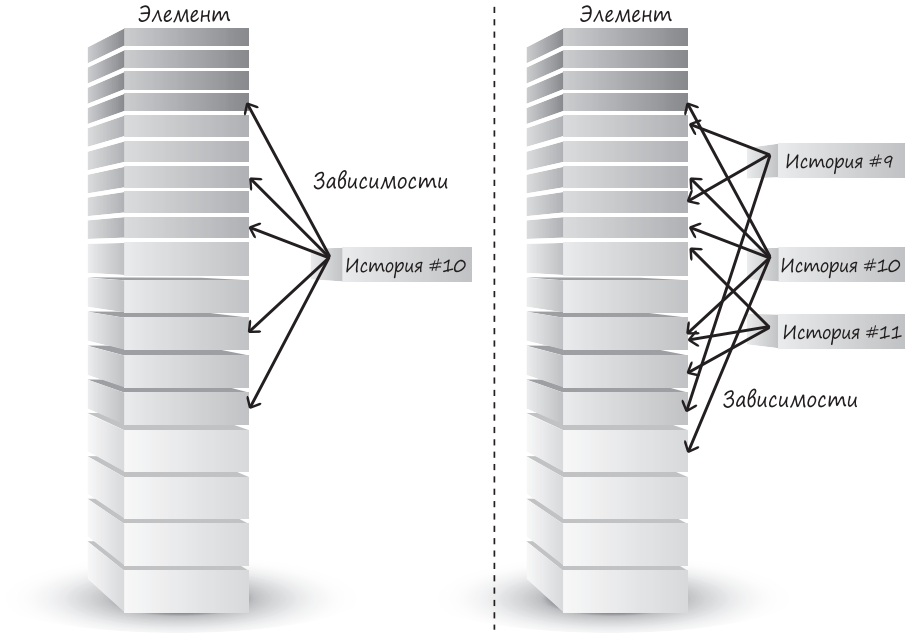


Рис. 5.8. Сильно зависимые истории

Прежде чем приступить к работе над историей **#10**, придется реализовать все остальные зависящие от нее истории. В данном конкретном случае это, возможно, не так уж и плохо, но представьте, что у вас имеется много разных историй с высокой степенью взаимозависимости, как показано на рис. 5.8. Пытаться назначить приоритеты для всех этих историй и решить, над какой из них следует работать в спринте, было бы, по меньшей мере, трудно. Когда применяется критерий независимости истории, то преследуется цель не исключить все зависимости, а написать истории таким образом, чтобы свести их зависимости к минимуму.

Обсуждаемость

Подробности истории должны быть *обсуждаемы*. Истории не являются контрактом, составленным в форме заранее подготовленного технического задания. Напротив, истории являются заменителями обсуждений, где согласовываются подробности.

В пригодных историях ясно схвачена суть функциональных возможностей, требующихся для коммерческого применения, а также причины, по которым они требуются. Но такие истории оставляют место для обсуждения владельцу продукта, участникам проекта и команде разработчиков.

Обсуждаемость помогает всем заинтересованным лицам избежать враждебных отношений и выявления виновных, столь характерных для заранее и подробно составленных технических заданий. Если истории *обсуждаемы*, разработчики не смогут сказать: "Если это средство вам требовалось, вы должны были указать его в техническом задании", поскольку подробности требований обсуждаются с разработчиками. А заказчики из деловой сферы не смогут сказать: "Вы просто не поняли техническое задание и поэтому создали не то, что нужно", поскольку заказчикам придется часто вести диалог с разработчиками, чтобы добиться общего взаимопонимания. Написание обсуждаемых историй позволяет избежать осложнений, связанных с заранее и подробно составленными требованиями, поскольку такие истории ясно указывают на необходимость диалога между заказчиками и исполнителями.

Типичным примером нарушения обсуждаемости служит ситуация, когда владелец продукта указывает команде разработчиков, *как* нужно реализовать историю. Истории должны пояснять, что и зачем, а не как нужно делать. Если же не обсуждается, как нужно делать, то возможности команды проявить новаторство уменьшаются. А *потеря новаторства* может привести к катастрофическим экономическим последствиям.

Но иногда для владельца продукта оказывается очень важно, *как* нужно делать. Например, для разработки функционального средства определенным способом могут существовать нормативные обязательства или же наложены коммерческие ограничения, направляющие на применение конкретной технологии. В подобных случаях истории становятся менее обсуждаемыми, поскольку порядок или способ их реализации отчасти задан в требованиях. И в этом нет ничего плохого. Ведь не все истории обсуждаемы полностью, хотя большинство из них должны быть таковыми.

Ценность

Истории должны быть *ценными* для заказчика, пользователя или и того и другого. Заказчики выбирают и оплачивают продукт, а пользователи употребляют его. Если история не представляет никакой ценности ни для тех, ни для других, она не вносится в задел продукта. Трудно себе представить ситуацию, когда какая-нибудь история ни для кого не имеет ценности, но все равно реализуется, поскольку это просто недопустимо. Такую историю нужно переписать или сделать ее ценной для заказчика или пользователя, а иначе ее придется отвергнуть.

А как насчет историй, ценных для разработчиков, но, очевидно, не представляющих никакой ценности для заказчиков или пользователей? Допустимо ли наличие так называемых *технических историй*, аналогичных приведенной на рис. 5.9?

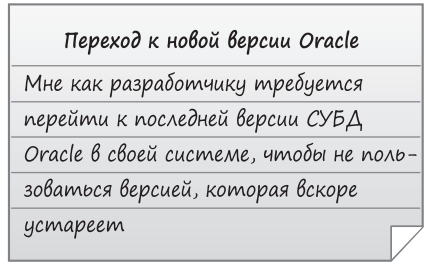


Рис. 5.9. Пример технической истории

Главный недостаток технических историй заключается в том, что владелец продукта может не усмотреть в них никакой ценности, что затруднит или вообще сделает невозможным назначение для них приоритетов по отношению к коммерчески ценным историям. Для того чтобы техническая история существовала, владелец продукта должен ясно понимать, зачем он ее оплачивает, а следовательно, какую ценность она в конечном итоге доставит.

Что же касается истории с переходом на новую версию СУБД Oracle, приведенной на рис. 5.9, то владелец продукта может поначалу не понять, в чем ценность смены версии базы данных. Но как только команда разработчиков объяснит ему риски, связанные с продолжением разработки на неподдерживаемой далее версии базы данных, владелец продукта может решить, что переход на новую версию СУБД является достаточно ценным, чтобы отложить построение новых функциональных средств до завершения такого перехода. Понимая ценность подобной технической истории, владелец продукта может истолковать ее как очередную коммерчески ценную историю и пойти на вполне осознанный компромисс. В итоге данная техническая история может быть включена в задел продукта.

Но на практике большинство технических историй, аналогичных приведенной на рис. 5.10, не должны вноситься в задел продукта. Вместо этого подобные истории должны стать задачами, связанными с завершением коммерчески ценных историй. Если у команды разработчиков имеется строгий критерий готовности, то писать подобные истории не следует, поскольку соответствующие работы подразумеваются критерием их готовности.

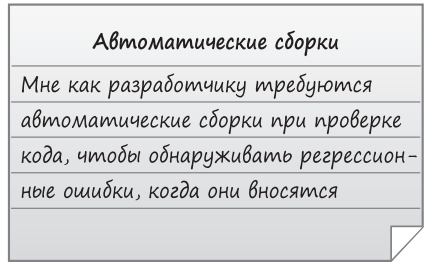


Рис. 5.10. Нежелательная техническая история

Затруднение, связанное с критерием ценности, состоит в том, что все истории в заделе продукта должны быть ценными, т.е. заслуживающими вложения в них средств с точки зрения владельца продукта, который представляет интересы заказчика и пользователя. Не все истории оказываются независимыми и полностью обсуждаемыми, но все они должны быть ценными.

Оцениваемость

Истории должны быть *оцениваемы* командой, которая их разрабатывает, строит и тестирует. Оценки позволяют выяснить размер истории, а следовательно, затраты труда и средств на их реализацию. Чем крупнее история, тем больше труда и средств потребуется на ее реализацию.

Зная размер истории, Scrum-команда получает практическую информацию для принятия решения. С одной стороны, владельцу продукта нужно знать стоимость истории, чтобы определить ее конечный приоритет в заделе продукта. А с другой стороны, команда разработчиков может определить на основании размера истории, требуется ли дополнительное уточнение или разбиение истории на части. Вполне возможно, что крупную историю, к работе над которой планируется вскоре приступить, придется разбить на ряд более мелких историй.

Если команда не в состоянии определить размер истории, это означает, что такая история слишком крупная или неопределенная или что у команды недостаточно знаний, чтобы оценить размер истории. Если история слишком крупная, команде придется потрудиться вместе с владельцем продукта над ее разбиением на более мелкие и поддающиеся управлению истории. А если у команды недостаточно знаний, то для получения недостающей информации потребуется исследовательская деятельность в той или иной форме (подробнее об этом речь пойдет несколько позже).

Подходящий размер (мелкость)

Истории должны быть наделены *подходящим размером* при планировании работы над ними. В частности, истории, над которыми планируется работать в спринтах, должны быть *мелкими*. Так, если планируется спринт в течение нескольких недель, придется работать над несколькими историями, каждая из которых длится несколько дней. А если планируется спринт в течение двух недель, то история не должна длиться две недели, поскольку слишком велик риск не завершить ее в срок.

В конечном счете истории должны быть мелкими, но если история крупная, то в этом нет ничего плохого. Допустим, имеется эпическая история, работать над которой не планируется еще год. Сомнительно, чтобы размер этой истории был выбран подходящим, когда планировалось работать над ней.

В действительности, если потратить время на разбиение данной эпической истории на ряд более мелких историй, такая трата времени может оказаться напрасной. Разумеется, если имеется эпическая история, над которой требуется работать в следующем спринте, то она имеет неподходящий размер, и поэтому придется дополнительно потрудиться над тем, чтобы привести ее к нужному размеру. Применяя данный критерий, следует принимать во внимание момент, *когда* именно будет начата работа над историей.

Тестируемость

Истории должны быть *тестируемы* по двоичному критерию, т.е. они должны проходить или не проходить связанные с ними тесты. Тестируемость означает наличие подходящего критерия приемки, имеющего отношение к условиям удовлетворения и связанного с данной историей. Это, по существу, рассматривавшийся ранее вопрос "подтверждения" пользовательской истории.

Если отсутствуют критерии тестируемости, то как узнать, что история завершена в конце спринта? Кроме того, тесты нередко выявляют в истории важные подробности, и поэтому они могут потребоваться еще до того, как история будет оценена. Но тестировать историю не всегда нужно или вообще возможно. Например, тесты, вероятно, отсутствуют или вообще не нужны для эпических историй, поскольку такие истории не реализуются непосредственно.

Иногда встречается и такая история, которую владелец продукта может посчитать ценной, хотя тестировать ее не имеет никакого практического смысла. Это, скорее всего, связано с нефункциональными требованиями вроде следующего: "Мне как пользователю требуется, чтобы система имела коэффициент использования 99,999%". Несмотря на полную ясность критериев приемки, тесты, выполняемые при вводе системы в эксплуатацию и подтверждающие, что заданный коэффициент использования достигнут, могут отсутствовать. Тем не менее данное требование по-прежнему остается ценным, поскольку оно продвигает процесс разработки.

Нефункциональные требования

Нефункциональные требования представляют собой ограничения, накладываемые на уровне системы. Мне редко приходится составлять нефункциональные требования в виде пользовательских историй, как показано на рис. 5.11, *слева*. Но я не считаю себя обязанным делать это, особенно если составлять их удобнее в другой форме, как показано на рис. 5.11, *справа*.

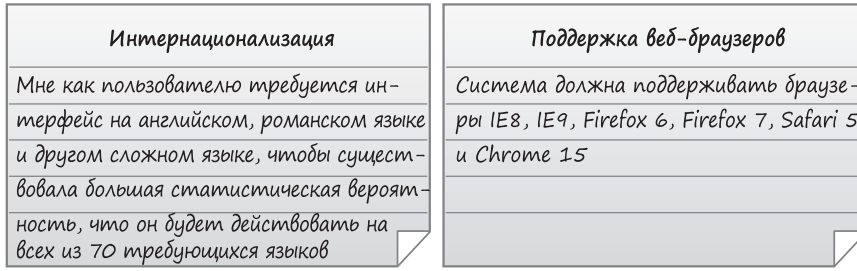


Рис. 5.11. Нефункциональные требования

В качестве ограничений, накладываемых на уровне системы, нефункциональные требования важны потому, что они оказывают влияние на разработку и тестирование большинства или даже всех историй в заделе продукта. Например, нефункциональное требование поддерживать веб-браузеры (см. рис. 5.11, *справа*) является типичным для любого проекта веб-сайта. Если команда разрабатывает функциональные средства веб-сайта, она должна обеспечить способность этих средств нормально работать во всеми указанными браузерами.

Кроме того, команда должна решить, когда следует провести тестирование во всех браузерах. Каждое нефункциональное требование должно быть, прежде всего, включено командой в критерий готовности. Так, если команда включит нефункциональное требование поддерживать веб-браузеры в свой критерий готовности, ей придется протестировать любые из новых функциональных средств, разработанных в спринте, во всех браузерах, перечисленных в данном требовании. Если же тестируемое функциональное средство не работает со всеми поддерживаемыми браузерами, то история считается незавершенной.

В связи с изложенным выше командам рекомендуется включать в свои критерии готовности как можно больше нефункциональных требований. Если же перенести проверку нефункциональных требований на позднюю стадию разработки, то тем самым будет отложено получение быстрой реакции на очень важные характеристики производительности системы.

Истории приобретения знаний

Иногда требуется создать элемент задела продукта, побуждающий к приобретению знаний. Это может быть связано с нехваткой знаний об эксплуатации продукта или процессе его построения для продвижения вперед. Поэтому требуется провести соответствующее исследование, как пояснялось в главе 3. Такое исследование называется по-разному, в том числе *созданием прототипа*, *подтверждением концепции*, *экспериментом*, *исследованием*, *всплеском* и т.д. Все эти термины, по существу, обозначают виды исследовательской деятельности,

направленные на приобретение недостающей информации. В качестве заменителя исследовательской деятельности я нередко употребляю пользовательскую историю (рис. 5.12).

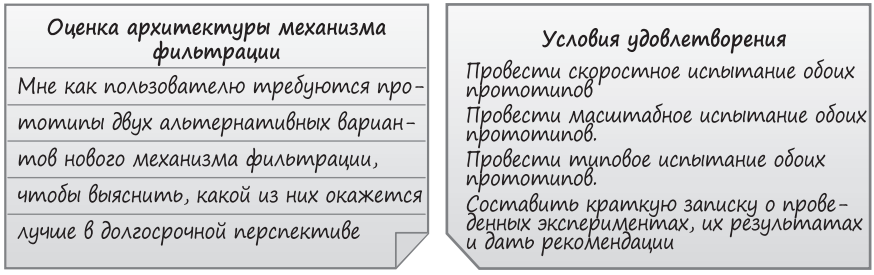


Рис. 5.12. Истории приобретения знаний

В рассматриваемом здесь примере команде разработчиков требуется оценить два возможных варианта архитектуры нового механизма фильтрации. Сначала предлагается создать прототипы обеих архитектур, а затем провести их скоростные, масштабные и типовые испытания. Конечным итогом этой деятельности по исследованию прототипов должна стать краткая записка, в которой описываются проведенные эксперименты, полученные результаты и рекомендации команды относительно того, что и как делать дальше.

Данная конкретная история приобретения знаний похожа на техническую историю, и, как пояснялось ранее, коммерческую ценность любой технической истории должен оценивать владелец продукта. Но поскольку владелец продукта мыслит экономическими категориями, то в подобного рода работах по созданию и испытанию прототипов должна быть какая-то экономическая целесообразность. Вероятнее всего, существует убедительный аргумент в пользу истории приобретения знаний, поскольку команда, как правило, не может продвинуться вперед до тех пор, пока не извлечет из данной истории недостающие знания. И в связи с этим перед командой встает следующий вопрос: насколько ценность полученных знаний превышает затраты на их приобретение?

Отвечая на этот вопрос, Scrum-команда могла бы принять следующий подход. Прежде всего нужно выяснить затраты на создание и испытание прототипов. Ведь ни один владелец продукта не даст согласие на неограниченное исследование. А команда разработчиков не сможет ответить на конкретные вопросы владельца продукта до тех пор, пока не будет принято архитектурное решение. Но в то же время она сможет ответить на вопрос, сколько труда потребуется на приобретение информации, необходимой для принятия архитектурного решения. Таким образом, команде предлагается определить размеры истории создания и испытания прототипов.

Допустим, что оценка размера истории свидетельствует о том, что всей команде придется работать над этой историей в течение одного спринта. А поскольку участники спринта и его продолжительность известны, то известны и затраты на приобретение недостающей информации. Допустим, что они составляют 10 тысяч долларов. Теперь нужно выяснить ценность подобной информации.

Рассмотрим один из способов, позволяющих выяснить, насколько ценной может быть приобретенная информация. Подбросим сначала монету. Если выпадет орел, то разрабатывается архитектура A, если выпадет решка — архитектура B. Затем предложим команде оценить затраты на ошибочное решение. Так, если подбросить монету и выпадет орел, то коммерческие средства будут надстраиваться поверх архитектуры A. Но если эта архитектура окажется ошибочной, то какие затраты повлечет за собой принятие неверного решения и переделка всей надстройки поверх архитектуры B? Допустим, что команда оценила эти затраты в 500 тысяч долларов.

Теперь у нас имеется достаточно информации, чтобы принять обоснованное экономическое решение. Итак, готовы ли мы потратить 10 тысяч долларов на приобретение информации, предполагаемая ценность которой составляет 250 тысяч долларов (т.е. один из вариантов выбора в результате подбрасывания монеты)? Очевидно, что такое решение выглядит вполне обоснованным. И теперь владелец продукта может обосновать причины, по которым данную историю следует включить в задел продукта.

И в качестве завершающего примера, демонстрирующего экономическое обоснование историй приобретения знаний, изменим цифры. Что, если в ответ на вопрос о затратах на ошибочное решение команда оценит их в 15 тысяч долларов? В таком случае решение реализовывать историю создания и испытания прототипов было бы неверным. В самом деле, зачем тратить 10 тысяч долларов на приобретение информации, предполагаемая ценность которой составит лишь 7,5 тысяч долларов? Не лучше ли просто подбросить монету (т.е. сделать эмпирическую оценку), и если мы ошибемся, то просто переделаем работу, воспользовавшись другой архитектурой. На самом деле данный пример не такой уж и надуманный, если принять во внимание постоянно развивающиеся и совершенствующиеся технологии. Это пример стратегии так называемого *быстрого провала*, которая заключается в опробовании какого-нибудь варианта, получении скорой реакции и быстрого обследования и адаптации.

Собирание историй

Откуда же пользовательские истории появляются на свет? Традиционные подходы к собиранию требований предусматривают спрашивать у пользователей, чего они, собственно, хотят. Я лично никогда не достигал особого успеха,

применяя такой подход. Как показывает мой опыт, пользователи намного лучше проявляют себя в роли критиков, чем авторов.

Так, если спросить у пользователя, что ему требуется, он может и не найти ответа на данный вопрос. И даже если он найдет на него ответ и мы разработаем именно то, что он просил, то в конечном итоге пользователь может сказать: "Вы дали мне именно то, что я просил, а теперь, как я это понимаю, мне нужно совсем другое". Уверен, что у многих из вас имеется подобный опыт общения с пользователями.

Более совершенный подход состоит в том, чтобы привлечь пользователей к формулированию того, что требуется разработать, чтобы постоянно оценивать то, что разрабатывается. Чтобы способствовать такому участию пользователей в разработке, во многих организациях устраиваются совещания по написанию пользовательских историй. Такие совещания служат в качестве основных средств для составления, по крайней мере, первоначального ряда пользовательских историй. А в некоторых организациях практикуется также построение карт историй для установления и предоставления контекста историй с ориентацией на пользователей. Каждый из этих подходов рассматривается далее по отдельности.

Совещания по написанию пользовательских историй

Цель *совещания по написанию пользовательских историй* — коллективная выработка коммерческой ценности и создание заменителей того, что должна делать продукция или услуга, в виде пользовательских историй.

В таком совещании обычно принимают участие владелец продукта, Scrum-мастер, команда разработчиков, а также внутренние и внешние участники проекта. Большинство подобных совещаний длятся от нескольких часов до нескольких дней. Мне редко приходилось быть свидетелем более продолжительных совещаний по написанию пользовательских историй, и я не думаю, что они должны длиться дольше. Ведь их назначение — заранее сформировать полный и готовый ряд пользовательских историй (аналогично полному техническому заданию на проект по методике последовательной разработки). В частности, я редко совмещаю подобные совещания с планированием первоначального выпуска, чтобы сформировать ряд историй, подходящих для предстоящего выпуска (подробнее об этом речь пойдет в главе 18).

Если это первое совещание, я обычно начинаю его с анализа роли пользователя. Цель такого анализа — определить перечень ролей пользователя для заполнения тех частей историй, где указывается роль пользователя ("Мне как <*роль пользователя*> требуется...). Разумеется, маркетологи и специалисты по анализу рынка могут отдельно выработать подходящее определение ролей пользователя перед совещанием по написанию пользовательских историй.

Основные характеристики роли можно также расписать в *лицах*. Например, "Лилия" вместе с соответствующим описание может представлять лицо, отвечающее роли юной участницы видеоигры в возрасте от 7 до 9 лет. Определив роль Лилии, можно приступить к написанию историй, указывая Лилию в роли пользователя вместо более абстрактного определения "юная участница видеоигры". Например, "Мне как Лилии требуется выбрать платье из обширного гардероба, чтобы настроить мой аватар по своему вкусу".

Не существует какого-то стандартного способа формирования пользовательских историй во время совещания. Одни команды предпочитают работать над историями по нисходящей, продвигаясь сверху вниз, а другие — снизу вверх по восходящей. Нисходящий подход состоит в том, что команда начинает работу с самой крупной истории вроде эпической, а затем сосредоточивает свои усилия на формировании обоснованного ряда более мелких историй, связанных с эпической историей.

А противоположный, восходящий подход состоит в том, чтобы начать сразу с коллективной выработки историй, связанных со следующим выпуском существующей системы. Ни один из этих подходов не лучше и не хуже другого, поэтому выберите тот, который вам больше всего подходит, а еще лучше — пользуйтесь ими попеременно, чтобы извлечь наибольшую пользу из обоих подходов.

Построение карт историй

Построение карт историй — это методика, распространенная Джеффом Паттоном [Patton, Jeff. 2009] и состоящая в формировании пользовательских историй с точки зрения самих пользователей. Ее основной принцип состоит в том, чтобы разложить высокоуровневую деятельность пользователя на последовательность операций, которую можно далее разложить на ряд детализированных задач (рис. 5.13).

Для описания иерархии в карте историй Паттон пользуется такими терминами, как *деятельность*, *задача* и *подзадача*. Но для согласованности с представленной ранее терминологией далее употребляются термины *эпическая история*, *тема* и *реализуемая история*.

На самом верхнем уровне этой иерархии находятся эпические истории, представляющие крупные виды деятельности, имеющие измеримую экономическую ценность. Примером тому служит эпическая история приобретения продукта (см. рис. 5.13).

Далее следует последовательность операций или типичных пользовательских задач, образующих эпическую историю и распределенных по темам — совокупностям связанных вместе историй. Темы располагаются по оси времени, где темы, происходящие раньше в последовательности операций, располагаются слева от тем, происходящих позднее. Например, тема поиска продукта должна располагаться слева от темы управления тележкой для закупок.

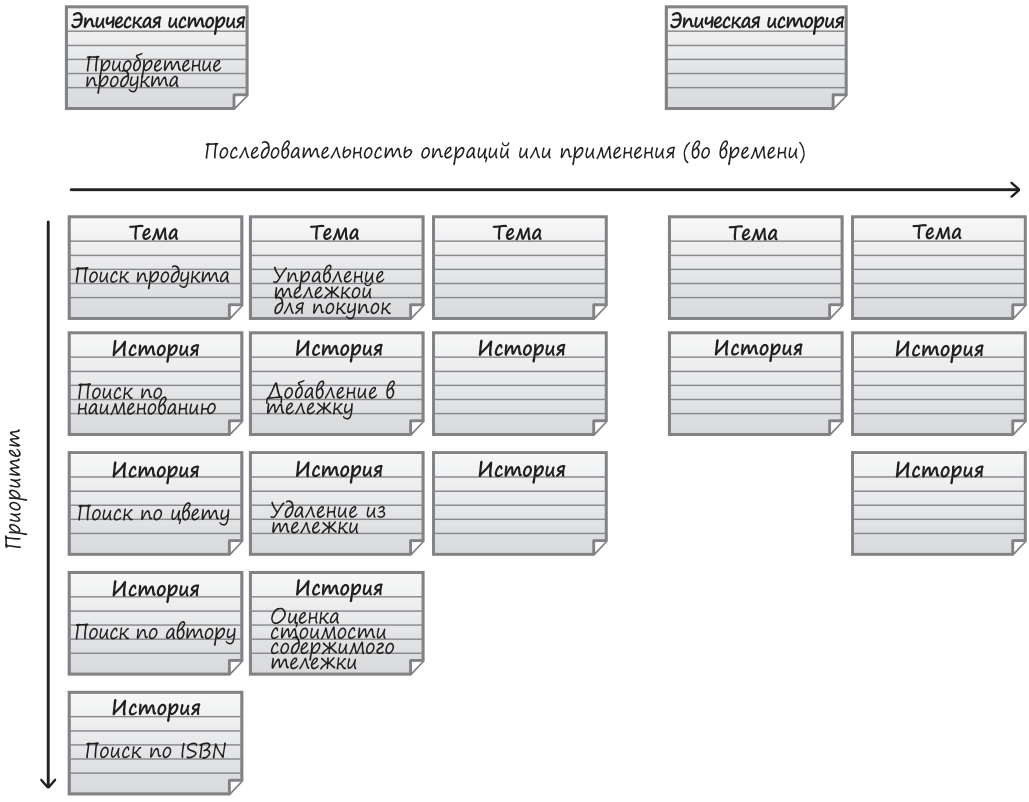


Рис. 5.13. Карта историй

Каждая тема разлагается далее на ряд реализуемых историй, располагаемых вертикально по порядку приоритетов, а на самом деле — по их востребованности, поскольку маловероятно, чтобы истории были уже оценены, а их приоритеты нельзя узнать до тех пор, пока не станут известны затраты. Не все истории из темы должны входить в один и тот же выпуск. Например, история поиска по цвету может быть не намечена для первого выпуска, тогда как история поиска по наименованию — намечена.

Построение карт историй сочетает в себе понятия разработки с ориентацией на пользователей и разложения историй на составляющие. Правильно составленные карты историй отображают поток видов деятельности с точки зрения пользователя и предоставляют контекст для понимания отдельных историй и их взаимосвязи с более крупными единицами потребительской ценности.

Даже если и не выполнять формальное построение карт историй, идея пользоваться последовательностями операций может оказаться плодотворной в течение совещаний по написанию пользовательских историй. Они помогают

сосредоточить обсуждение на написании историй в контексте доставки полной последовательности операций, представляющих ценность для пользователя. Имея в своем распоряжении нужный контекст, намного проще выяснить, были ли пропущены какие-нибудь важные истории, связанные с последовательностью операций.

Отличие традиционных совещаний по написанию пользовательских историй от построения карт историй заключается, в частности, в том, что во время совещания основное внимание уделяется формированию историй, а не их расстановке по приоритетам (т.е. расположению реализуемых историй по вертикали на карте историй). Таким образом, построением карты историй можно пользоваться в качестве дополнения к совещаниям по написанию пользовательских историй для наглядного представления расстановки историй по приоритетам. Карты историй дают двухмерное представление о заделе продукта вместо традиционного линейного (одномерного) его представления.

Заключение

В этой главе обсуждались отличия в трактовке требований к проектам, разрабатываемым по методике Scrum и традиционной последовательной методике. При выполнении проектных работ по методике Scrum создаются заменители требований, называемые элементами задела продукта. Эти элементы нередко выражаются в виде пользовательских историй и проходят через весь Scrum-процесс с явным акцентом на обсуждения как способ доведения требований до нужных подробностей. Помимо этого, применяется стратегия постепенного уточнения и сведения более крупных и менее детализированных историй к более мелким, но и более детализированным историям своевременно и в достаточной степени.

Пользовательские истории были формально представлены в этой главе в контексте понятий карточки, обсуждения и подтверждения. В ходе обсуждения этих понятий было показано, каким образом пользовательские истории могут представлять коммерческую ценность на разных уровнях абстракции. Далее в этой главе пояснялось, каким образом пригодность пользовательских историй определяется по критериям INVEST. Затем были представлены способы обращения с нефункциональными требованиями и видами деятельности по приобретению знаний. И в заключение обсуждались способы собирания пользовательских историй, опирающиеся на проведение совещаний по написанию пользовательских историй и построение карт историй. А в следующей главе речь пойдет о заделе продукта.

Глава 6
Задел продукта

В этой главе описывается важная роль, которую задел продукта играет в проекте, разрабатываемом по методике Scrum. Сначала в ней описываются различные типы элементов, которыми обычно наполняется задел продукта. Затем обсуждаются характеристики пригодного задела продукта и поясняется, каким образом упорядочение задела продукта способствует достижению этих характеристик. Далее описываются причины, по которым задел продукта является главным элементом в управлении быстрым и гибким потоком как на уровне выпуска, так и на уровне спринта. И в завершение поясняется, как определить, какие именно заделы продукта и сколько их нужно иметь.

Краткий обзор

Задел продукта — это список очередности реализации желательных функциональных возможностей продукта. Он обеспечивает централизованное и общее представление о том, что требуется разрабатывать и в каком именно порядке. Этот весьма наглядный артефакт находится в сердцевине инфраструктуры Scrum и доступен для всех участников проекта (рис. 6.1). Задел продукта существует при условии, что для разработки, усовершенствования и сопровождения имеется продукт или система.

Элементы задела продукта

Задел продукта состоит из пунктов, которые называются *элементами задела продукта* или просто *элементами* (рис. 6.2).

Большинство элементов задела продукта составляют функциональные средства, т.е. элементы функциональных возможностей, которые будут иметь ощутимую ценность для пользователя или заказчика. Эти элементы, как правило, пишутся в виде пользовательских историй, хотя конкретная форма элементов задела продукта в Scrum не определяется. К примерам функциональных средств относится нечто совершенно новое (экран регистрации на новом веб-сайте) или изменение в уже имеющемся функциональном средстве (более удобный для пользователей экран регистрации на действующем веб-сайте). А к числу

других элементов задела продукта относятся дефекты, которые нужно устранить, технические усовершенствования, работа по приобретению знаний и любая другая работа, которую владелец продукта считает ценной. Примеры различных типов элементов задела продукта приведены в табл. 6.1.

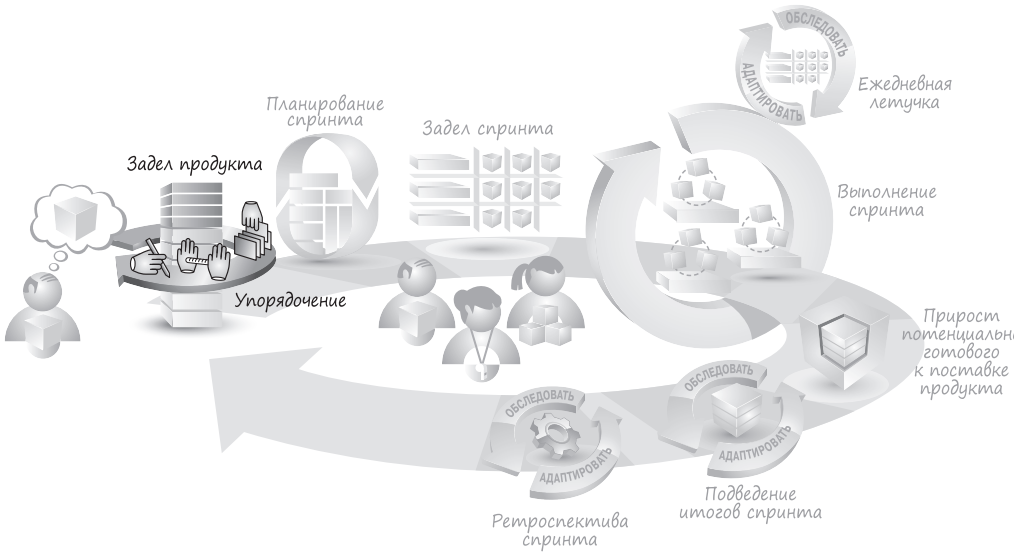


Рис. 6.1. Задел продукта находится в сердцевине инфраструктуры Scrum

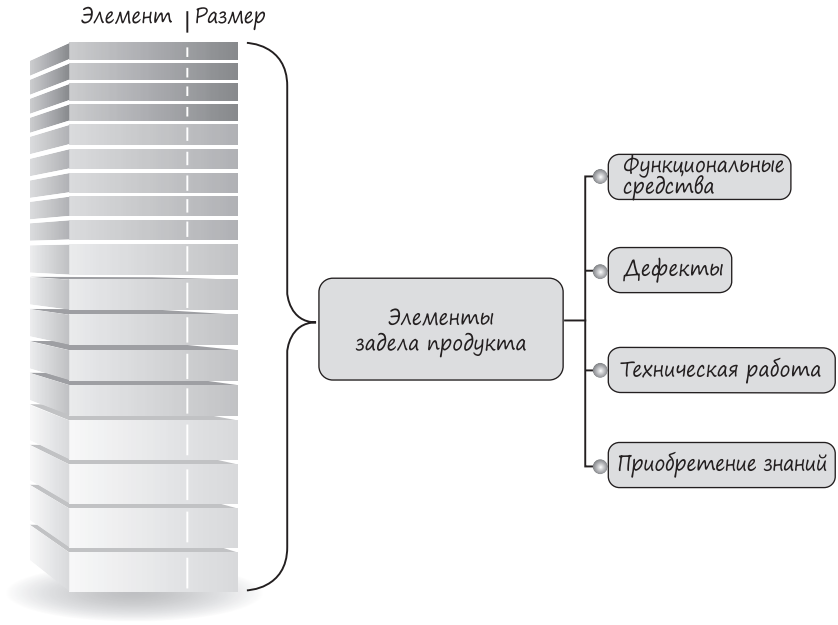


Рис. 6.2. Элементы задела продукта

Таблица 6.1. Примеры элементов задела продукта

Тип элемента задела продукта	Пример
Функциональное средство	Мне как представителю службы поддержки клиентов требуется создать заявку для решения вопросов поддержки клиентов, чтобы я мог регистрировать запросы клиентов на поддержку и управлять ими
Изменение	Мне как представителю службы поддержки клиентов требуется стандартное упорядочение результатов поиска по фамилии, а не по номеру заявки, чтобы упростить поиск заявок на поддержку
Дефект	Устранить дефект **#256** в системе обнаружения неисправностей, чтобы специальные символы в критериях поиска не приводили к аварийному сбою, когда клиент совершает поиск
Техническое усовершенствование	Перейти к последней версии СУБД Oracle
Приобретение знаний	Создать прототип или опытный образец двух архитектур и провести три теста, чтобы выяснить, какая их этих архитектур больше подходит для данного продукта

Характеристики пригодного задела продукта

Все пригодные заделы продуктов проявляют сходные характеристики. Роман Пихлер [Pichler, Roman. 2010] и Майк Кон придумали сокращение *DEEP*, чтобы подытожить характеристики пригодных заделов продуктов, которые должны быть *детализируемыми должным образом* (**D**etailed appropriately), *развивающимися* (**E**mergent), *оцениваемыми* (**E**stimated) и *приоритизированными* (**P**rioritized). Аналогично критериям INVEST, рассматривавшимся в главе 5 для оценки пригодности пользовательских историй, критериями DEEP удобно пользоваться, чтобы выяснить, насколько хорошо структурирован задел продукта. Рассмотрим каждый из этих критериев по отдельности.

Детализация должным образом

Не все элементы в заделе продукта оказываются одновременно детализированными на одном и том же уровне (рис. 6.3).

Те элементы задела продукта, которые скоро предстоит разрабатывать, должны располагаться ближе к вершине задела, иметь малые размеры и быть детализированными настолько, чтобы к работе над ними можно было приступить в ближайшем по срокам спринте. А те элементы задела продукта, которые предстоит разрабатывать нескоро, должны располагаться ближе к дну задела, иметь бо́льшие размеры и быть менее детализированными. Этого должно быть достаточно, поскольку работать над ними не планируется в ближайшее время.

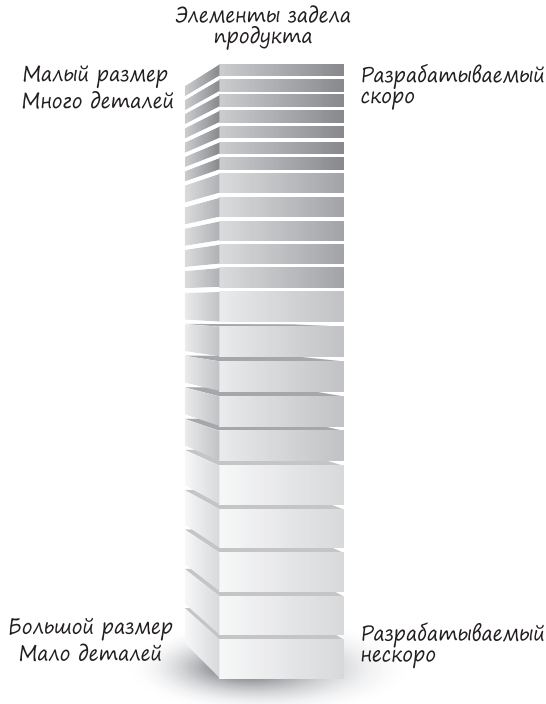


Рис. 6.3. Элементы задела продукта разных размеров

По мере приближения к крупному элементу задела продукта вроде эпической истории эта история разбивается на ряд более мелких, готовых для спринта историй. И это нужно делать своевременно. Ведь если приступить к уточнению слишком рано, то придется потратить немало времени на выяснение подробностей и в конечном итоге так и не реализовать историю. А если ждать слишком долго, то нормальный поток элементов из задела продукта в спринт будет нарушен и работа команды застопорится. Поэтому нужно найти золотую середину между достаточностью и своевременностью подготовки элементов задела продукта к работе.

Развитие

Задел продукта не завершается и не прекращается до тех пор, пока продукт разрабатывается или сопровождается. Напротив, он постоянно обновляется, исходя из экономически ценной информации, которая поступает непрерывным потоком. Например, заказчики могут изменить свое решение в отношении того, что им требуется; конкуренты могут совершить неожиданные, непредсказуемые шаги; а кроме того, могут возникнуть непредвиденные технические трудности. Задел продукта, собственно, и предназначен для того, чтобы адаптироваться к подобным обстоятельствам.

Таким образом, структура задела продукта постоянно развивается во времени. По мере того, как вводятся новые или уточняются уже имеющиеся элементы, владелец продукта должен восстанавливать равновесие и переставлять приоритеты в заделе продукта, принимая во внимание новую информацию.

Оцениваемость

Размер каждого элемента продукта оценивается по затратам труда, которые требуются для разработки данного элемента (рис. 6.4).

Рис. 6.4. Оцениваемость элементов задела продукта

Владелец продукта пользуется полученными оценками как одним из нескольких исходных данных, помогающих определить приоритетность элементов, а следовательно, их расположение в заделе продукта. Кроме того, высокоприоритетный, крупный элемент, располагающийся ближе к вершине задела продукта, служит владельцу продукта сигналом, что данный элемент нужно уточнить, прежде чем переносить его в ближайший по срокам спринт.

Как обсуждается более подробно в главе 7, большинство элементов задела продукта оцениваются в очках за историю или в идеальных днях. Эти оценки размеров элементов должны быть обоснованно, но не чрезмерно точными. Элементы, располагающиеся ближе к вершине задела продукта, оказываются более мелкими, но и более детализированными, и поэтому оценки их размеров будут более низкими и точными. А крупные элементы вроде эпических историй, располагающиеся ближе

к дну задела продукта, не всегда можно оценить численно, и поэтому в некоторых командах они вообще не оцениваются или же для их оценки употребляются размеры футболок (L, XL, XXL и т.д.). По мере уточнения этих крупных элементов и их сведения к ряду мелких элементов каждый мелкий элемент оценивается численно.

Приоритизация

Несмотря на то что задел продукта представляет собой список очередности реализации его элементов, маловероятно, чтобы *все* элементы были расставлены в заделе по приоритетам (рис. 6.5).

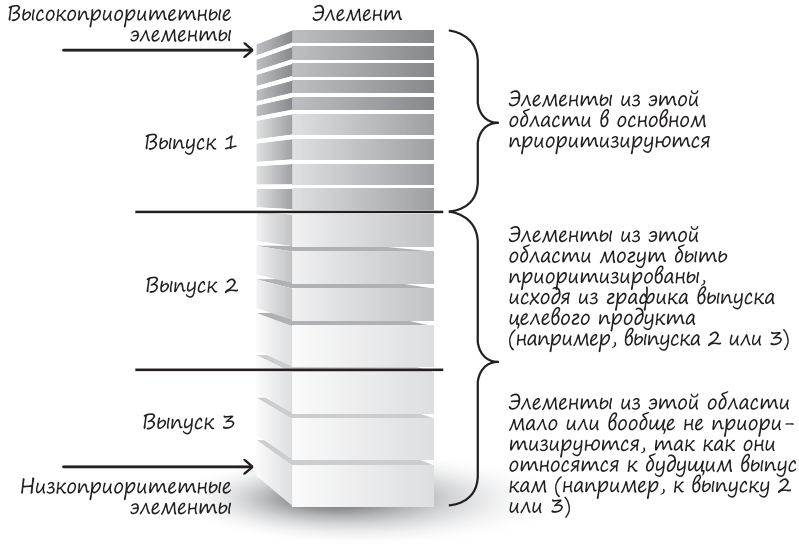


Рис. 6.5. Элементы расставляются в заделе по приоритетам

Целесообразно расставлять по приоритетам ближайшие по срокам элементы, предназначенные для следующего спринта. Это полезно делать в глубину задела до предполагаемого момента первого выпуска. А дальше расстановка приоритетов на более высоком уровне вряд ли стоит затраченного времени.

Например, мы можем объявить, что элемент предназначен для второго или третьего выпуска в соответствии с имеющимся у нас графиком выпуска продукта. Но если мы находимся на ранней стадии разработки функциональных средств первого выпуска, то вряд ли стоит тратить драгоценное время на расстановку по приоритетам тех функциональных средств, которые предполагается разработать во втором или в третьем выпуске. Ведь мы можем так и не дойти до второго или третьего выпуска, или наши замыслы в отношении этих выпусков могут существенно измениться в течение разработки первого выпуска. Следовательно, время на расстановку по приоритетам элементов,

расположенных так далеко в заделе, может с большой долей вероятности оказаться потраченным впустую. Разумеется, по мере появления новых элементов в процессе разработки владелец продукта обязан ввести их в правильном порядке, принимая во внимание те элементы, которые уже находятся в заделе.

Упорядочение задела

Чтобы получить пригодный задел продукта, отвечающий критериям DEEP, нужно с упреждением управлять, организовать, администрировать или, как часто говорят, ухаживать за заделом продукта. Все это предусматривает упорядочение задела.

Что означает упорядочение задела

Упорядочение задела продукта означает три основных вида деятельности: создание и уточнение (дополнение подробностями) элементов задела продукта, их оценку и расстановку по приоритетам. На рис. 6.6 приведен ряд конкретных задач упорядочения и показано их влияние на структуру задела продукта.

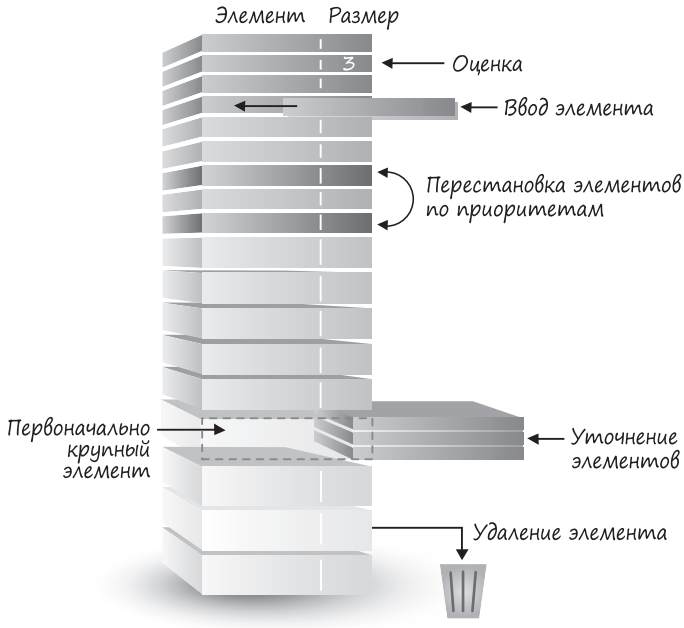


Рис. 6.6. Упорядочение реорганизует задел продукта

В подходящий момент все элементы задела продукта должны быть оценены, чтобы расположить их в заделе по порядку и решить, насколько оправданы дополнительные затраты труда на их уточнение. Кроме того, по мере того,

как становится доступной важная информация, элементы переупорядочиваются в заделе. А по мере приближения к работе над крупным элементом возникает потребность уточнить и свести его к ряду более мелких элементов. Можно также прийти к выводу, что конкретный элемент просто не требуется, и в этом случае он должен быть удален из задела.

Кто упорядочивает задел продукта

Упорядочение задела продукта — это непрерывный коллективный труд под руководством владельца продукта. Оно предусматривает также привлечение внутренних и внешних участников проекта, Scrum-мастера и команды разработчиков (рис. 6.7).

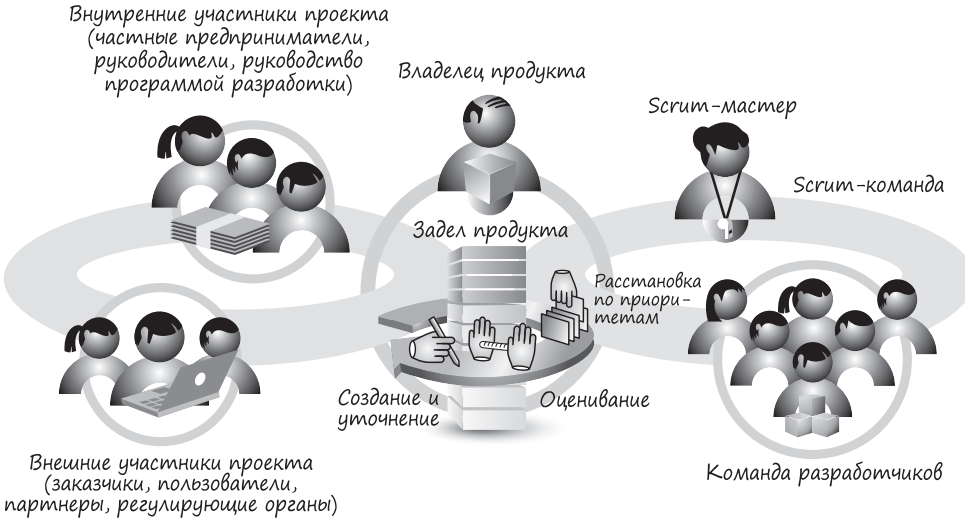


Рис. 6.7. Упорядочение задела продукта — это коллективный труд

В отношении упорядочения задела окончательное решение остается за владельцем продукта. Но грамотные владельцы продуктов понимают, что коллективное упорядочение побуждает к важному диалогу между всеми участниками и, опираясь на коллективный разум и взгляды отдельных лиц и команд, позволяет выявить важную информацию, которая иначе может быть упущена из виду. Грамотные владельцы продуктов также знают, что, привлекая к упорядочению задела разных участников проекта, они могут обеспечить более ясное, общее представление о заделе продукта, а следовательно, меньше времени будет тратиться на недопонимание и передачу ответственности по кругу. Подобный коллективный труд способствует также ликвидации пропасти, исторически сложившейся между деловыми людьми и техническими работниками.

Участники проекта должны выделить достаточно времени упорядочению задела продукта в зависимости от характера деятельности организации и типа проекта. Как правило, команда разработчиков должна выделять до 10% своего времени в каждом спринте на помощь владельцу продукта в работе над упорядочением задела. Это время команда потратит на помощь в создании или анализе элементов, возникающих в заделе продукта, а также на постепенное уточнение крупных элементов и их сведение к более мелким элементам. Команда может также оценить размеры элементов задела продукта и помочь владельцу продукта расставить их по приоритетам, исходя из технических зависимостей и ограничений на ресурсы.

Когда происходит упорядочение задела продукта

В инфраструктуре Scrum только обозначается потребность в упорядочении задела продукта, но не указывается, *когда* именно оно должно происходить. Так когда же следует упорядочивать задел? Если мы следуем методике последовательной разработки, то пытаемся заранее зафиксировать полное и подробное описание требований, и поэтому их упорядочение планируется мало или вообще не предусматривается после того, как они будут утверждены. Во многих организациях такие принятые за основу требования могут быть изменены только в отдельном процессе контроля изменений, который перемежается с основным ходом проектных работ (рис. 6.8).

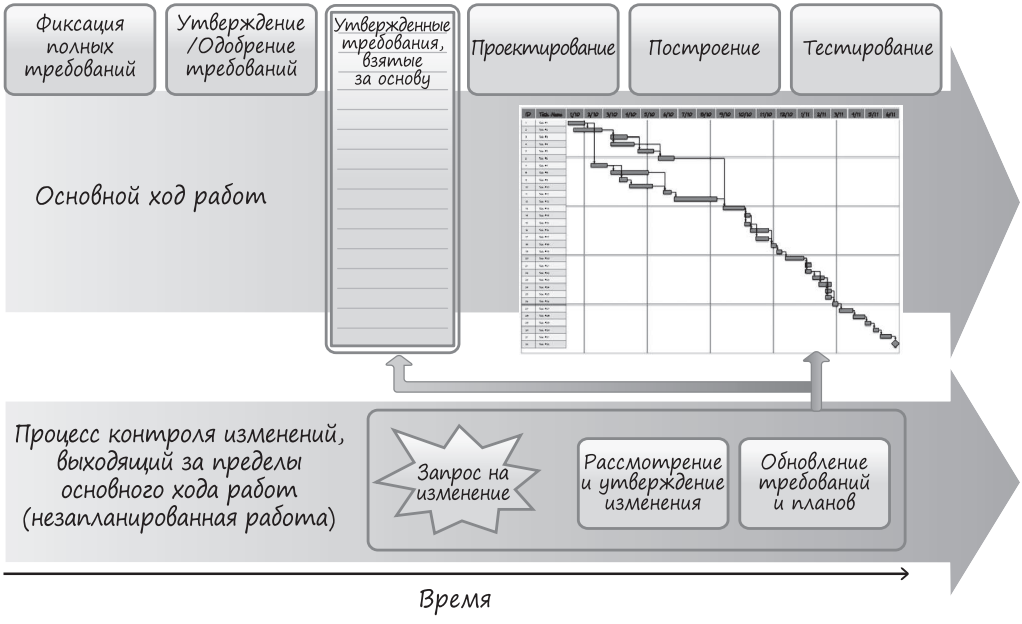


Рис. 6.8. Упорядочение, выходящее за пределы основного хода работ над последовательно разрабатываемыми проектами

Таким образом, упорядочение в течение последовательной разработки является исключительным, незапланированным видом деятельности, выходящим за пределы основного хода работ. Упорядочение происходит лишь в том случае, если в нем возникает насущная потребность, что нарушает быстрый поток доставки коммерческой ценности.

Применяя методику Scrum, мы допускаем наличие неопределенной обстановки, а следовательно, должны быть готовы к постоянному обследованию и адаптации. Мы предполагаем, что задел продукта постоянно развивается, а не зафиксирован на ранней стадии и подлежит изменению только в ходе вторичного процесса обработки исключительных, нежелательных событий. Таким образом, мы должны непременно сделать виды упорядочивающей деятельности важной и неотъемлемой частью управления ходом выполняемых нами работ. Различные моменты времени, когда может быть выполнено упорядочение задела, приведены на рис. 6.9.

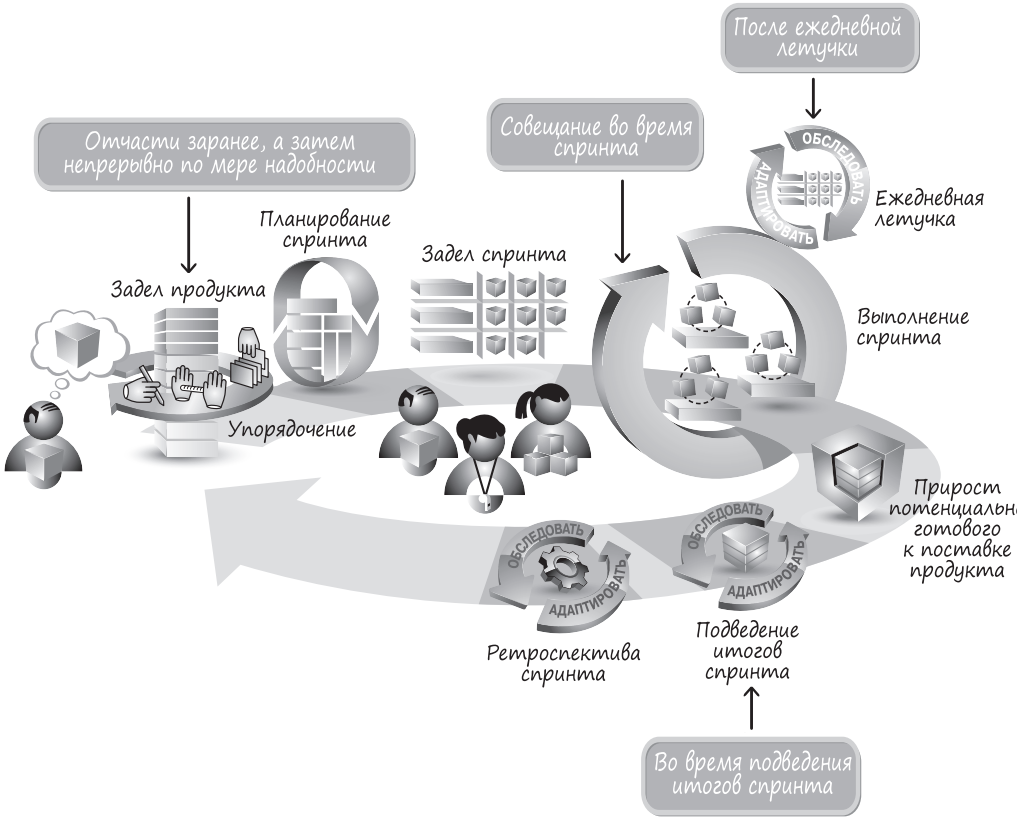


Рис. 6.9. Моменты, когда происходит упорядочение задела продукта

Первоначальное упорядочение задела продукта происходит как часть деятельности по планированию выпуска (подробнее об этом — в главе 18). Во время

разработки продукта его владелец встречается с участниками проектами настолько часто, насколько это требуется для непрерывного упорядочения задела.

Работая с командой разработчиков, владелец продукта может запланировать проведение совещаний по упорядочению задела каждую неделю или один раз в течение спринта. Благодаря этому гарантируется регулярное упорядочение задела по графику, а команда может выделить необходимое для этого время при планировании спринта. Помимо этого, сокращаются убытки от попыток запланировать внеочередные заседания (например, определить момент времени, когда все заинтересованные лица присутствуют, найти подходящее место для совещания и т.д.).

Иногда команды предпочитают распределять упорядочение задела по всему спринту, а не заранее намечать для этого конкретное время. Они уделяют немного времени после ежедневных летучек для постепенного упорядочения задела. Такое упорядочение не требует обязательного присутствия всех членов команды. Например, после ежедневной летучки владелец продукта может попросить ее участников помочь ему уточнить крупную историю. Те члены команды, которые знакомы с этой историей и заинтересованы в ней, остаются и помогают владельцу продукта. А в следующий раз ему могут помочь другие члены команды.

Даже если команды регулярно проводят запланированные совещания или уделяют каждый день некоторое время для анализа задела продукта, большинство команд считают, что естественным образом упорядочивают задел при подведении итогов спринта. По мере того, как все заинтересованные лица начинают лучше понимать, в каком состоянии находится продукт и в каком направлении продвигается работа над ним, зачатую создаются новые элементы задела продукта, переставляются по приоритетам уже имеющиеся элементы или удаляются тех из них, которые больше не нужны. Для гибкой и быстрой доставки коммерческой ценности важнее не момент, когда упорядочивается задел продукта, а обеспечение его плавной интеграции в ход проектных работ по методике Scrum.

Критерий подготовленности

Упорядочение задела продукта должно гарантировать, что элементы, расположенные на вершине задела, готовы к переносу в спринт. И тогда команда разработчиков может с уверенностью взять на себя обязательство завершить их к концу спринта.

Некоторые Scrum-команды предпочитают формализовать данное требование, установив *критерий подготовленности*. Критерии подготовленности и готовности (см. главу 4) можно рассматривать как два состояния элементов задела продукта в течение рабочего цикла спринта (рис. 6.10).

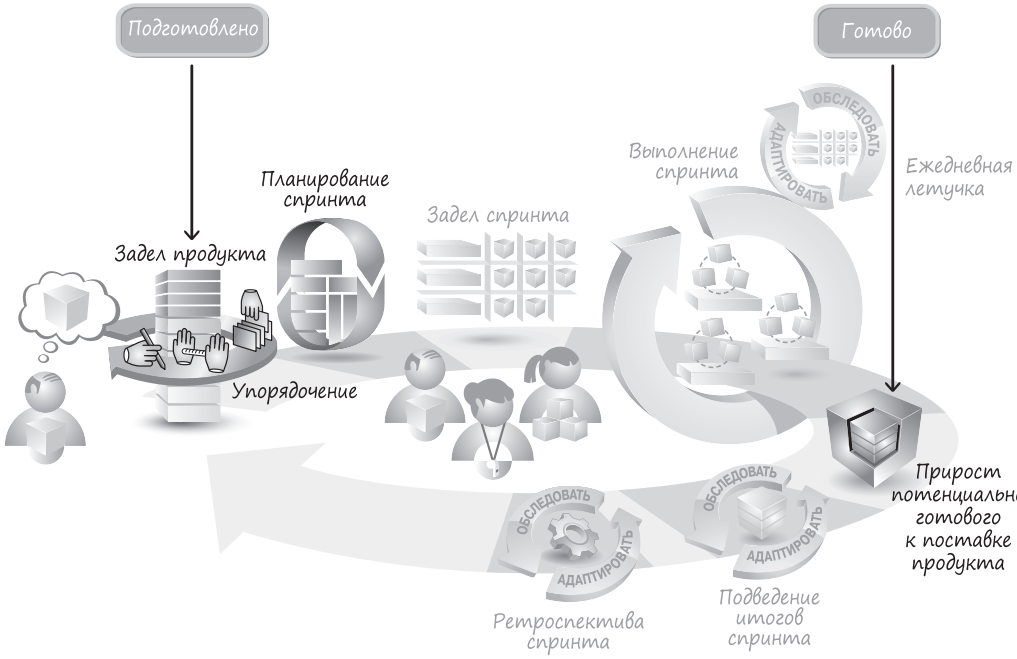


Рис. 6.10. Критерий подготовленности

Оба критерия подготовленности и готовности представляют собой контрольные списки работ, которые должны быть завершены, прежде чем элемент задела продукта можно будет считать находящимся в соответствующем состоянии. Пример контрольного списка по критерию подготовленности элементов задела продукта приведен в табл. 6.2. Строгий критерий подготовленности существенно повышает шансы Scrum-команды успешно достичь поставленной цели спринта.

Таблица 6.2. Пример контрольного списка по критерию подготовленности

Критерий подготовленности
❑ Коммерческая ценность ясно сформулирована
❑ Подробности понятны для команды разработчиков настолько, что она может обоснованно решить, способна ли она завершить данный элемент задела продукта
❑ Зависимости выявлены, и ни одна из внешних зависимостей не помешает завершить данный элемент задела продукта
❑ Команда укомплектована должным образом, чтобы завершить данный элемент задела продукта
❑ Данный элемент задела продукта оценен и достаточно мал, чтобы его было удобно завершить в течение одного спринта
❑ Критерии соответствия ясны и тестируемы
❑ Критерии качества, если таковые имеются, определены и тестируемы
❑ Scrum-команде понятно, как продемонстрировать данный элемент задела продукта при подведении итогов спринта

Управление потоками

Задел продукта служит важным средством, позволяющим Scrum-команде достигать быстрого и гибкого потока доставки ценности в присутствии неопределенности, которую нельзя устранить из разработки продукции. Необходимо признать, что поток экономически важной информации будет постоянно поступать, и поэтому придется организовать работу и руководить ею (т.е. управлять заделом продукта) таким образом, чтобы эту информацию можно было обработать быстро и экономично, поддерживая поток в нормальном состоянии. Рассмотрим роль задела продукта в поддержании нормального потока выпуска и потока спринта.

Управление потоком выпуска

Задел продукта должен быть упорядочен таким образом, чтобы поддерживать непрерывное планирование выпуска (т.е. поток функциональных средств в пределах выпуска). Как показано на рис. 6.5, выпуск можно наглядно представить в виде прямой линии, проходящей через задел продукта. Все элементы задела продукта, расположенные выше этой линии, предназначаются для данного выпуска; а все элементы задела продукта, расположенные ниже, — не предназначаются. Задел продукта удобно разделить двумя прямыми линиями для каждого выпуска, как показано на рис. 6.11.

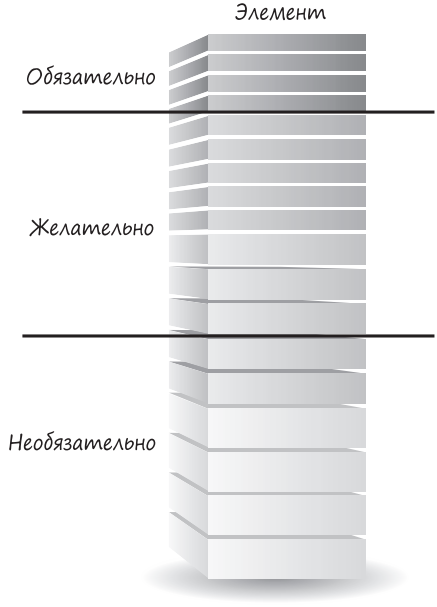


Рис. 6.11. Представление задела продукта на уровне выпуска

Этими двумя линиями задел разделяется на три области: *обязательную*, *желательную* и *необязательную*. *Обязательные функциональные средства* представляют те элементы, которые просто должны присутствовать в предстоящем выпуске, иначе он окажется непригодным для заказчика. *Желательные функциональные средства* представляют те элементы, которые предназначены для следующего выпуска, но их хотелось бы включить в текущий выпуск. Но если для этого не хватит времени и ресурсов, то желательные средства можно исключить, что не помешает выпустить вполне пригодный продукт. И наконец, *необязательные функциональные средства* представляют те элементы, которые не должны быть включены в текущий выпуск. Нижняя линия, отделяющая необязательные функциональные средства от остальных на рис. 6.11, соответствует линии, обозначающей первый выпуск на рис. 6.5. Поддержание задела продукта в таком состоянии помогает лучше выполнять непрерывное планирование выпуска, как поясняется в главе 18.

Управление потоком спринта

Упорядочение задела продукта имеет большое значение для эффективного планирования спринта и возникающего в итоге потока функциональных средств, направляемого в спринт. Если задел продукта детализирован должным образом, элементы, расположенные на вершине задела, должны быть ясно описаны и тестируемы.

Упорядочивая задел продукта для пригодного потока спринта, удобно представить задел в виде трубопровода, по которому требования направляются потоком в спринт на проектирование, построение и тестирование командой разработчиков (рис. 6.12).

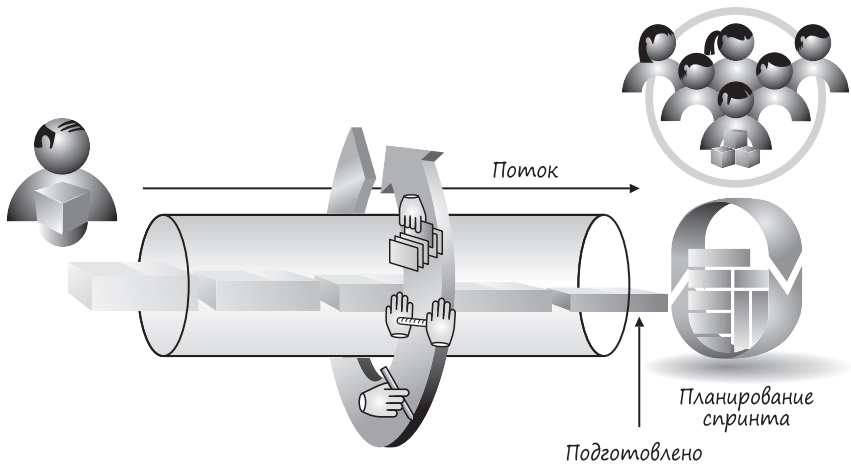


Рис. 6.12. Представление задела продукта в виде трубопровода, по которому требования направляются потоком в спринт

Как показано на рис. 6.12, более крупные и менее понятные требования поступают в трубопровод. По мере их прохождения по трубопроводу и приближения к тому моменту, когда они выходят из него и направляются на разработку, они постепенно уточняются в ходе упорядочивающей деятельности. С правого края трубопровода находится команда разработчиков. К тому моменту, когда элемент задела продукта выходит из трубопровода, он должен быть подготовлен, т.е. детализирован настолько, чтобы команда смогла его понять и спокойно выпустить в течение спринта.

Если входной и выходной потоки не согласованы или не одинаковы, то возникает затруднение. Так, если поток упорядоченных, детализированных и подготовленных к реализации элементов слишком медленный, то трубопровод в конечном итоге опорожнится и команда не сможет спланировать и выполнить следующий спринт (это означает главное нарушение потока или потери в Scrum). А если ввести в трубопровод слишком много элементов для уточнения, то образуется большой запас детализированных требований, которые, возможно, придется переделать или отвергнуть, узнав о них больше (это означает главный источник потерь). Следовательно, в идеальном случае в запасе должно быть достаточное количество элементов задела спринта, чтобы создать равномерный поток, но не так много, чтобы привести к потерям.

Для управления потоком спринта Scrum-команды применяют подход, который заключается в том, чтобы образовать в заделе подходящий запас упорядоченных и подготовленных к реализации элементов. Для многих команд подходит такой приближенный критерий: в заделе должно быть историй на два или три спринта. Так, если команда способна обычно реализовать около 5 элементов задела продукта в одном спринте, то она упорядочивает свой задел продукта таким образом, чтобы в нем всегда было около 10–15 элементов, подготовленных к разработке. Такой лишний запас гарантирует, что поток не иссякнет, а команда имеет возможность выбирать по мере надобности элементы из задела не по порядку, а из соображений производительности или прочих ограничений, накладываемых на спринт (более подробно этот вопрос рассматривается в главе 19).

Требующиеся заделы продуктов и их количество

Когда требуется решить, какие именно заделы продуктов и сколько их требуется сформировать, я обычно руководствуюсь следующим правилом: один продукт, один задел. Это означает, что у каждого продукта должен быть свой отдельный задел, который допускает описание и расстановку выполняемых работ по приоритетам в пределах данного продукта.

Но данное правило следует применять аккуратно, чтобы получить в конечном итоге практичную, работоспособную структуру задела продукта. Например, в одних случаях не всегда ясны составляющие продукта, в других — продукт оказывается очень крупным, в третьих — над продуктом поочередно работают разные команды, а иногда — одна команда над несколькими продуктами. Рассмотрим каждый из этих случаев по отдельности, чтобы выяснить их влияние на упомянутое выше правило единственного задела на каждый продукт.

Что такое продукт

Трудность соблюдения правила единственного задела на каждый продукт состоит в том, что не всегда ясно, что именно составляет сам продукт. Например, является ли текстовый редактор Microsoft Word продуктом или только составляющей более крупного продукта под названием Microsoft Office? Если предполагается продавать только набор продуктов, то следует ли иметь задел на весь этот набор или на каждое составляющее его приложение (рис. 6.13)?

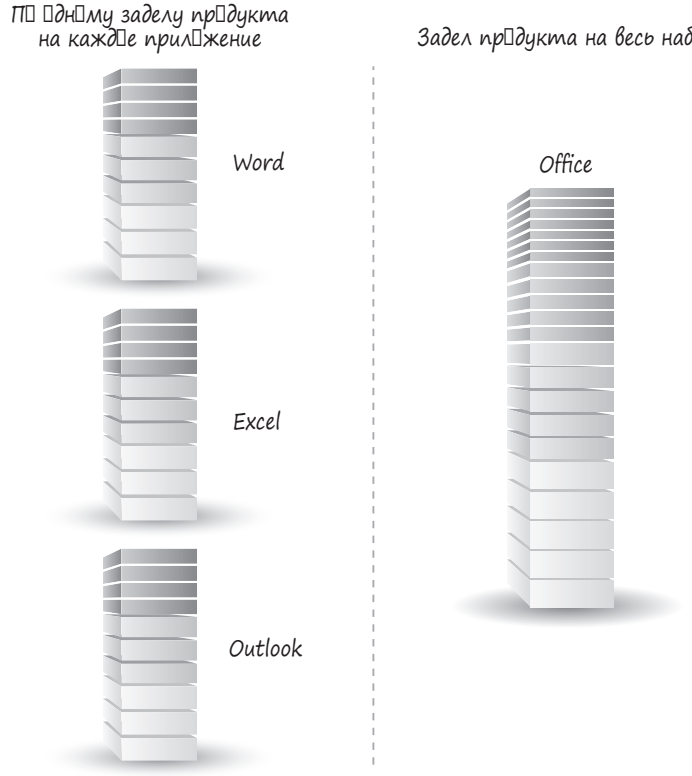


Рис. 6.13. Задел тесно связан с продуктом

Когда я работал в компании IBM, то на вопрос: “Что такое продукт с точки зрения заказчика?” был такой ответ: “Все, что имеет свой однозначный идентификационный номер продукта”. Прелесть такого ответа заключается в его простоте. Компания IBM продавала свою продукцию по каталогу, и достаточно было указать в нем идентификационный номер продукта, чтобы агенты по сбыту включили его в заказ, а следовательно, это и был “продукт”. Несмотря на кажущуюся чрезмерную простоту такого ответа, им можно воспользоваться в качестве отправной точки для определения сущности продукта. Продукт — это нечто такое ценное для заказчика, за что он готов платить, или нечто такое, что изготовитель готов упаковать и продать.

Применение рассматриваемого здесь правила усложняется, если команды разработчиков формируются по отдельным составляющим продукта и предназначены для создания *одного компонента* крупного продукта, который заказчик намерен приобрести, и поэтому они называются *командами для компонентов* (более подробно команды для компонентов обсуждаются в главе 12). Например, когда я приобретал свое портативное устройство GPS, то покупал не алгоритм маршрутизации, а портативное устройство, которое даст мне точное географическое местоположение и подробно озвучит маршрут моего продвижения, указывая направление на каждом повороте.

Если производитель устройства GPS создал отдельную команду для разработки компонента маршрутизации, то требуется ли отдельный задел для этого компонента, а если имеется лишь один задел продукта на все устройство GPS, то включены ли функциональные средства маршрутизации в этот задел? А еще любопытнее, если один и тот же компонент маршрутизации предполагается внедрять в разные продукты GPS, у каждого из которых имеется свой идентификационный номер, то следует ли в таком случае создавать отдельный задел продукта для данного компонента, если его предполагается сделать общим для продуктов различных устройств?

Как видите, задавая подобные вопросы, мы можем зайти слишком далеко. Во избежание этого мы не должны забывать, что наша цель — свести к минимуму количество команд для компонентов, а следовательно, и потребность в заделах продукта для отдельных компонентов. Создаваемый продукт следует рассматривать как упаковываемый, доставляемый и приносящий пользу конечному потребителю. В таком случае задел продукта следует привести в соответствие с данным предложением.

Иерархические заделы для крупных продуктов

При всякой возможности я предпочитаю один задел продукта, даже если это и крупный продукт, подобный комплекту Microsoft Office. Но, применяя данное

правило, нужно все-таки быть практичным. Если организуются проектные работы по созданию такого крупного продукта, как сотовый телефон, то придется объединить усилия десятков, а то и сотен команд, чтобы выпустить конкурентоспособное устройство. Поэтому пытаться свести элементы задела продукта всех этих команд в единый поддающийся управлению задел продукта было бы непрактично и вряд ли нужно.

Прежде всего, не все эти команды работают на не связанных вместе участках. Например, семь команд могут работать над аудиовизуальным проигрывателем для сотового телефона, а еще восемь команд — над веб-браузером для него. На каждом из этих участков доставляются разные ценности, ощутимые для потребителя, и поэтому работа на каждом из них может быть организована и расставлена по приоритетам на подробном уровне независимо от остальных участков. Исходя из этих характеристик, в большинстве организаций проблема разработки крупных продуктов разрешается созданием иерархических заделов (рис. 6.14).

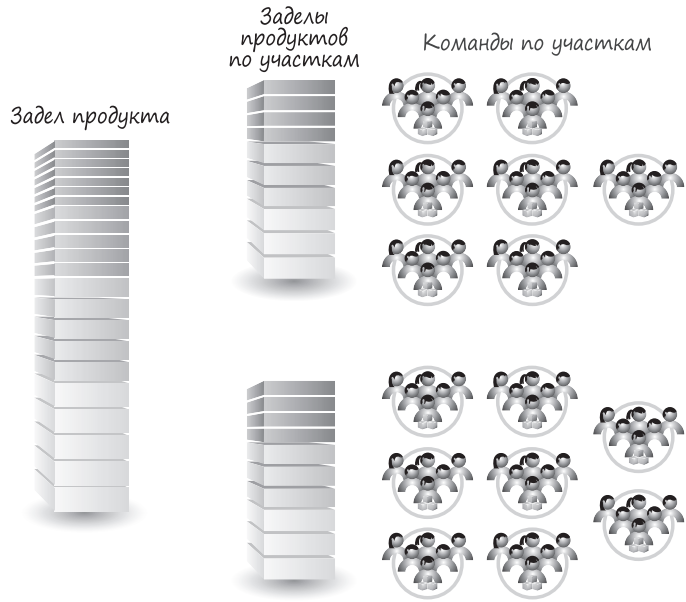


Рис. 6.14. Иерархические заделы продуктов

На вершине такой иерархии по-прежнему находится один задел продукта, описывающий и расставляющий по приоритетам крупномасштабные функциональные средства (возможно, эпические истории) разрабатываемого продукта. У такого задела должен быть также один главный владелец продукта, как поясняется в главе 9. А на каждом участке, связанном с отдельным функциональным средством, имеется свой задел продукта. Таким образом, на участке

аудиовизуального проигрывателя имеется задел продукта, содержащий элементы для семи команд, работающих на данном участке. Элементы задела продукта на уровне отдельных функциональных средств, скорее всего, будут иметь более мелкий масштаб (величиной с функциональное средство или историю), чем соответствующие элементы из главного задела продукта. В главе 12 обсуждается понятие *выпускного поезда*, основанное на трехуровневой модели заделов предприятия: заделе портфеля заказов, состоящем из эпических историй; заделе программы, состоящем из функциональных средств, а также на заделах команд, состоящих из поддающихся спринту историй.

Один задел продукта на несколько команд

Правило единственного задела на каждый продукт позволяет всем командам, работающим над продуктом, коллективно пользоваться единым заделом. Согласование работы всех команд по единому заделу продукта позволяет оптимизировать экономические показатели на уровне целого продукта. Такое преимущество получается потому, что все функциональные средства размещаются в одном заделе и выпускаются по приоритетам относительно остальных средств. И благодаря этому высокоприоритетные функциональные средства выявляются и разрабатываются с точки зрения целого продукта в первую очередь.

Если все команды взаимозаменяемы, т.е. любая команда может работать над любым элементом из одного задела продукта, то преимущество, которое дает единый задел продукта, становится вполне очевидным. Но что если команды не взаимозаменяемы? Например, команде, работающей над механизмом компоновки текста в Microsoft Word, вряд ли стоит поручать работу над механизмом расчета электронных таблиц в Microsoft Excel. И хотя это далеко от идеала, тем не менее не каждая команда может иногда работать над всяким элементом из задела продукта.

В подобных реальных условиях нужно знать, над какими именно элементами из задела продукта может работать каждая команда. Теоретически для каждой команды требуются отдельные заделы, а на практике заделы продуктов не создаются на уровне отдельных команд. Вместо этого имеются представления отдельных команд о коллективно используемом заделе. На рис. 6.15 показан единственный задел продукта, но он структурирован таким образом, чтобы отдельные команды выявляли и выбирали из него только те функциональные средства, которые отвечают уровню их квалификации и компетенции.

Кроме того, на рис. 6.15 показано, что элемент с наивысшим приоритетом на уровне задела для команды C оказывается производным от элемента, обладающего не очень высоким приоритетом на уровне задела всего продукта в целом.

Если бы команды были взаимозаменяемыми, то задел для команды C соответствовал бы большинству высокоприоритетных элементов из задела всего продукта в целом. Именно этот недостаток гибкости вынуждает многие организации стремиться владеть общим кодом на высоком уровне и делать команды более взаимозаменяемыми, чтобы извлечь выгоду из наличия команд, способных работать над продуктом на многих участках.

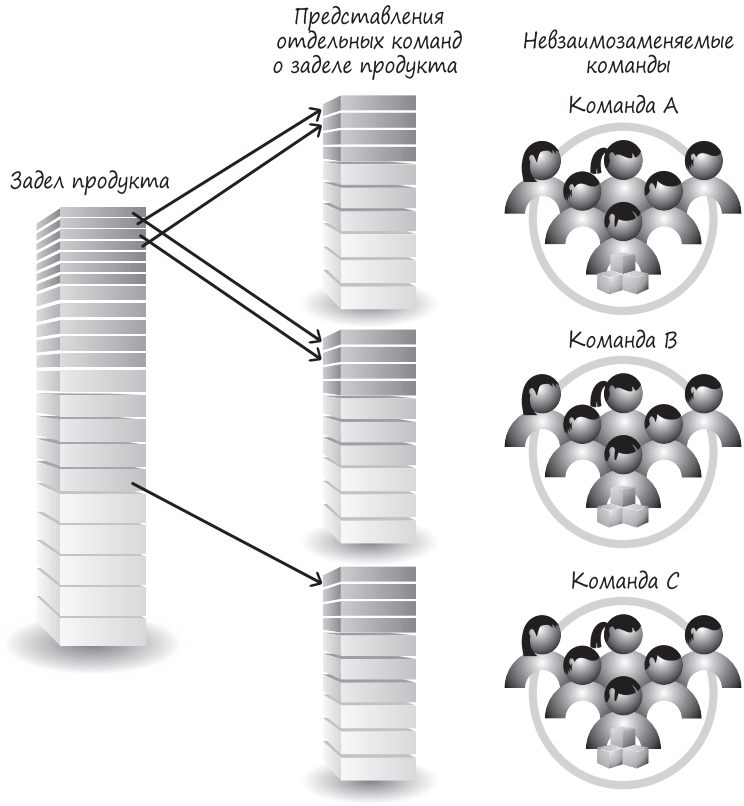


Рис. 6.15. Представления отдельных команд о коллективно используемом заделе

Несколько продуктов на одну команду

Если организация разрабатывает несколько продуктов, то у каждого из них имеется свой задел. Обращаться с несколькими заделами продуктов лучше всего, назначая каждый из них по отдельности для одной или более команды, как показано на рис. 6.16, *слева*.

Но иногда одной команде приходится работать над продуктами из нескольких заделов (см. рис. 6.16, *справа*). Как поясняется в главе 11, главная цель — свести к минимуму объем многопроектных работ, выполняемых командами

или их членами. Первое (и зачастую наилучшее) решение — распределить работу таким образом, чтобы команда работала над продуктам по очереди. И в каждом спринте команда должна работать только над элементами из одного задела продукта.

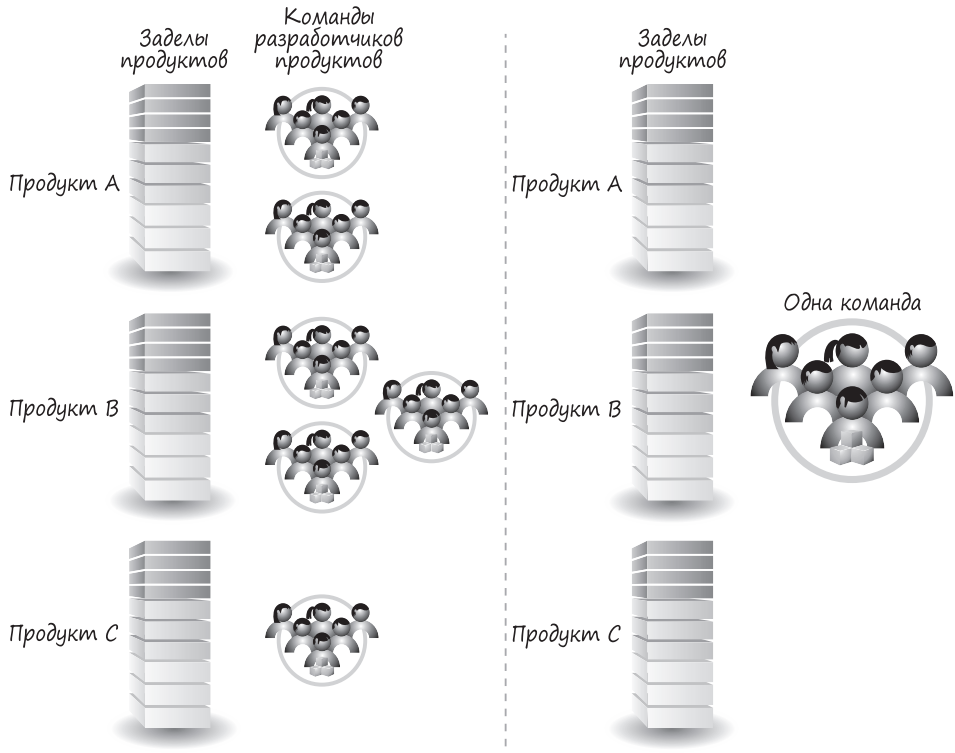


Рис. 6.16. Разные способы обращения с несколькими заделами продуктов

Но если препятствия организационного порядка вынуждают поручить одной команде работу над несколькими продуктами одновременно, то имеет смысл объединить элементы из заделов всех этих продуктов в один задел. Для этого владельцы всех этих продуктов должны собраться вместе и определить приоритетность работы над ними.

Даже если выбрать ведение трех отдельных заделов продуктов, в каждом спринте кому-то (скорее всего, владельцу продукта) все равно придется собрать расставленные по приоритетам элементы из трех заделов (вероятно, выделив предварительно команде время на каждый продукт в течение спринта) и представить их команде на рассмотрение и взятие соответствующих обязательств на выполнение работ.

Заключение

В этой главе обсуждалась важная роль задела продукта в достижении быстрого и гибкого потока доставки ценности в присутствии неопределенности. В ней обсуждался целый ряд вопросов, связанных со структурой задела продукта и процессом его ведения, включая типы элементов задела продукта, а также упорядочение задела продукта для достижения желательных его характеристик. И в заключение пояснялось, какие именно заделы продукта требуются и сколько их должно быть. А в следующей главе обсуждается порядок оценки элементов задела продукта и измерения скорости разработки на основании этих оценок.

Глава 7
Оценивание и скорость работы

В этой главе описываются понятия оценивания и скорости работы. Сначала в ней дается краткий обзор важных ролей, которые оценивание и скорость играют в планировании гибкой разработки. Затем обсуждаются различные элементы, которые требуется оценивать, а также моменты и способы их оценки. Бо́льшая часть этой главы посвящена оцениванию элементов задела продукта, включая выбор единицы измерения и применение покера планирования. Далее в главе рассматривается понятие скорости работы и важность скоростного диапазона для планирования. По ходу дела обсуждается, каким образом новые команды могут прогнозировать скорость работы в отсутствие хронологических данных. И в заключение поясняется, как можно оказывать влияние на скорость работы и как не следует злоупотреблять этим диагностическим показателем.

Краткий обзор

При планировании и управлении разработкой продукции приходится отвечать на следующие важные вопросы: “Когда продукт будет готов и во что это обойдется?” Для ответа на эти вопросы по методике Scrum придется оценить объем работ и изменить скорость их выполнения. Имея в своем распоряжении эту информацию, можно определить вероятную продолжительность разработки продукции (и соответствующие затраты), разделив оцененный объем работ над рядом функциональных средств на скорость работы команды (рис.7.1).

Если оценивать задел продукта, приведенный на рис. 7.1, то сколько времени потребуется для создания функциональных средств в первом выпуске? Чтобы ответить на этот вопрос, нужно сначала определить объем работ в первом выпуске, т.е. его размер. Для этого достаточно сложить оценки размеров отдельных элементов задела спринта, предназначенных для первого выпуска. (В данном примере сумма всех подобных оценок составляет 200 очков.)

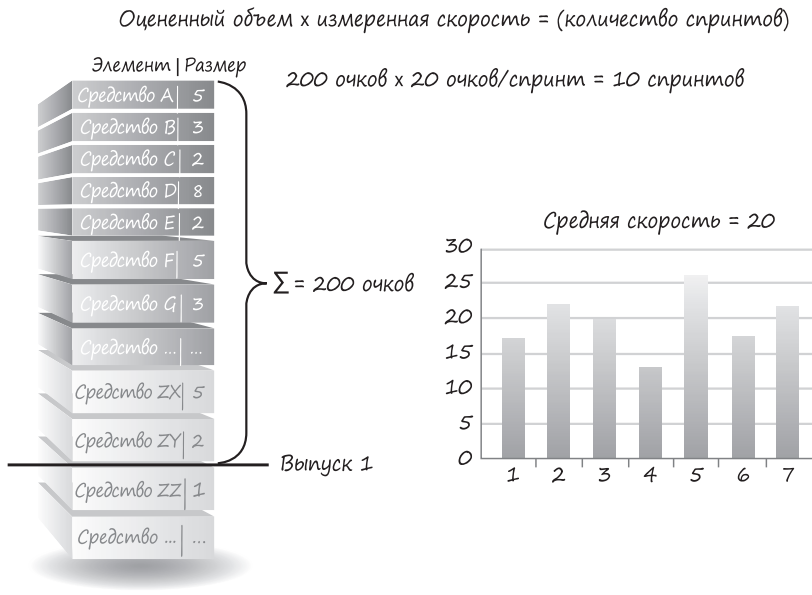


Рис. 7.1. Соотношение объема, скорости и продолжительности работ

Зная приблизительный размер выпуска, можно обратить внимание на скорость работы команды, т.е. объем работ, который команда обычно выполняет в каждом спринте. Измерить скорость совсем не трудно. Для этого достаточно сложить оценки размеров отдельных элементов, завершенных в течение спринта. Если же элемент не завершен, он не учитывается при измерении скорости работы. Сумма размеров всех элементов задела продукта, завершенных в спринте, и даст скорость работы команды. Как следует из графика на рис. 7.1, *справа*, средняя скорость работы команды в течение семи спринтов составляет **20**.

Имея в своем распоряжении оцененный объем работ и измеренную скорость их выполнения, можно рассчитать их продолжительность. Для этого достаточно разделить объем работ на их скорость. Так, если размер первого выпуска (т.е. объем его работ) составляет 200 очков, а команда способна в среднем выполнить объем работы на 20 очков в течение каждого спринта, то для завершения первого выпуска ей потребуется 10 очков (более подробное описание планирования выпуска приведено в главе 18). Далее в этой главе поясняются причины, по которым для подобных расчетов лучше пользоваться скоростным диапазоном, поскольку он дает более точные результаты, чем средняя скорость работы. Но в целях иллюстрации здесь употребляется средняя скорость.

Несмотря на то что основное соотношение объема, скорости и продолжительности работ остается неизменным, в каких-то мелочах оно может отличаться в зависимости от стадии проектных работ, измеряемого показателя

и намерений использовать полученные данные. Рассмотрим оценку и скорость работы более подробно, чтобы выяснить, каким образом эти показатели изменяются в зависимости от характера и момента выполнения работ.

Что и когда оценивается

На рис. 7.1 были использованы очки за историю, чтобы выразить оценки элементов задела продукта для расчета продолжительности выпуска. Но на протяжении всего срока разработки продукции подобные оценки приходится делать с разной степенью точности, а следовательно, употреблять для этой цели разные единицы измерения (рис. 7.2).

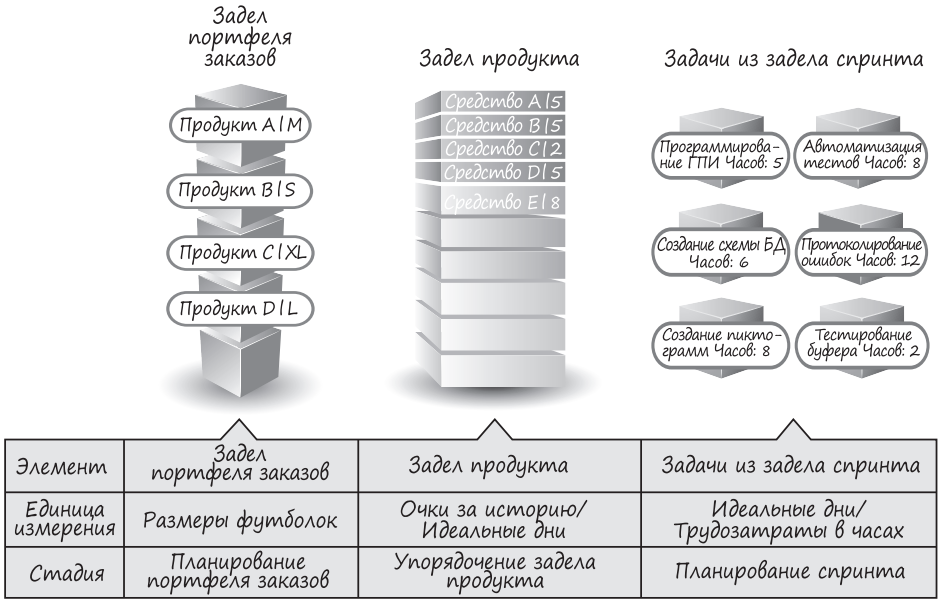


Рис. 7.2. Что и когда оценивается

Большинство организаций делают оценки для целей планирования на трех разных уровнях детализации. Эти оценки проявляются в заделе портфеля заказов, продукта и спринта. Рассмотрим каждую из этих оценок в отдельности.

Оценивание элементов из задела портфеля заказов

Несмотря на то что задел портфеля заказов формально не относится к Scrum, он ведется во многих организациях и состоит из списка очередности разработки всех продуктов (или проектов). Чтобы правильно назначить приоритет для каждого элемента в заделе портфеля заказов, нужно знать приблизительные затраты

на реализацию этого элемента. Как пояснялось в главе 5, на момент, когда запрашиваются цифры этих затрат, еще не составлен полный ряд требований, и поэтому нельзя воспользоваться стандартной методикой для оценивания каждого требования в отдельности и последующего суммирования отдельных оценок с целью получить совокупную оценку общих затрат.

Вместо этого для оценки элементов из задела портфеля заказов во многих организациях предпочитают пользоваться грубыми оценками относительных размеров вроде малого, среднего, большого, очень большого и прочих размеров футболок. Более подробно употребление размеров футболок для планирования портфеля заказов рассматривается в главе 16.

Оценивание элементов из задела продукта

Как только продукт или проект будет утвержден, можно приступать к конкретизации элементов его задела, но оценивать их следует по-разному. Если приоритет элементов из задела продукта повышается и в результате упорядочения они содержат больше подробностей, то большинство команд предпочитают оценивать их размеры численно, употребляя для этого очки за историю или идеальные дни. Более подробно оба подхода рассматриваются далее в этой главе.

Оценивание элементов из задела продукта является частью вида деятельности по упорядочению задела. Момент, когда такое упорядочение происходит, приведен на рис. 6.9. Как правило, оценивание элементов из задела продукта проводится на так называемых "оценочных" совещаниях, первое из которых зачастую совпадает с первоначальным планированием выпуска. Владелец продукта может также созвать дополнительные оценочные совещания в течение спринта, если потребуется оценить любые новые элементы в заделе продукта.

Не все практикующие Scrum считают, что оценивание размеров элементов из задела продукта необходимо. Как показывает их опыт, когда команды приобретают достаточный опыт работы по методике Scrum, они в состоянии создавать элементы задела продукта достаточно мелкими и приблизительно одинакового размера. Такие практикующие считают оценивание мелких элементов сходного размера напрасной тратой времени, а вместо этого просто подсчитывают количество элементов в заделе продукта. Они все-таки пользуются понятием скорости работы, но измеряют этот показатель в виде количества элементов задела продукта, завершенных в течение спринта, а не суммы их размеров.

Мне, конечно, понятны аргументы тех, кто ратует за отказ от оценивания размеров элементов из задела продукта. Тем не менее я лично предпочитаю оценивать их по следующим причинам.

- Как упоминалось в главе 5, не все элементы задела продукта одновременно одинаковы, и поэтому некоторые из них оказываются в заделе крупнее

остальных, несмотря на то, что ближе к вершине располагается ряд более мелких элементов сходного размера.

- Командам требуется некоторое время на приобретение навыков разбиения задела продукта на элементы, приблизительно одинаковые по размеру.
- Командам, возможно, придется разбивать истории в неудобные моменты времени, чтобы добиться одинаковости их размеров.
- И наконец, самое главное: одно из главных преимуществ оценивания заключается в обучении, которое происходит во время оценочных совещаний. Ничто так не способствует здоровому обсуждению, как просьба к его участникам оценить что-нибудь. Благодаря этому сразу же раскрываются любые разногласия и выявляются допущения. Если же отказаться от оценивания, то вместо него придется предложить равноценный способ поощрения здоровых обсуждений.

Оценивание задач

На самом детализированном уровне находятся задачи из задела спринта. Большинство команд предпочитают определять размеры задач при планировании спринта, и поэтому они могут быть уверены, что рассматриваемые ими обязательства вполне обоснованы (подробнее об этом — в главе 19).

Размеры задач оцениваются в *идеальных часах*, называемых также *трудозатратами в часах* или *человеко-часах*. Так, на рис. 7.2 показано, что команда оценила выполнение задачи программирования графического пользовательского интерфейса (ГПИ) за 5 человеко-часов. Но это не означает, что на выполнение задачи потребуется 5 часов времени. Одному человеку для программирования ГПИ может потребоваться два дня, а двум работающим вместе людям — меньше одного дня. Данная оценка просто устанавливает трудозатраты, которые потребуются команде для выполнения рассматриваемой здесь задачи. Более подробно оценивание задач рассматривается в главе 19 при описании подробностей планирования спринта.

Принципы оценивания элементов из задела продукта

Несмотря на важность всех трех уровней детализации, в остальной части этой главы основное внимание уделяется оцениванию на уровне задела продукта. Оценивая элементы задела продукта, Scrum-команды руководствуются несколькими важными принципами (рис. 7.3). Рассмотрим каждый из этих принципов в отдельности.

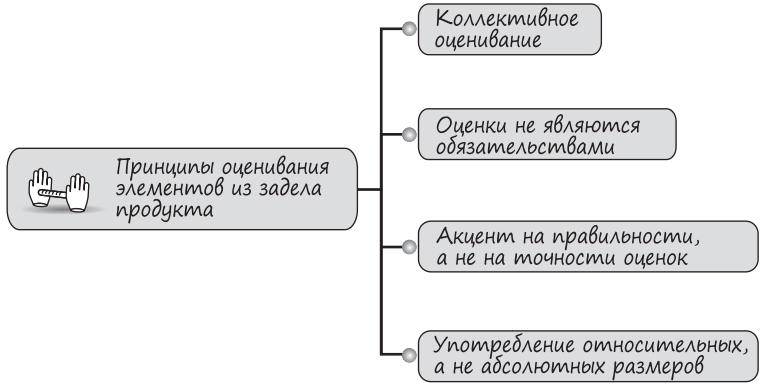


Рис. 7.3. Принципы оценивания элементов из задела продукта

Коллективное оценивание

Во многих традиционных организациях первоначальное оценивание объемов работ может выполнять руководитель проекта, главный конструктор, архитектор или ведущий разработчик. А остальным членам команды может быть предоставлена возможность для последующего анализа и комментирования полученных оценок. Применяя методику Scrum, мы придерживаемся следующего простого правила: те, кто выполняют работу, коллективно оценивают ее.

Точнее говоря, под теми, кто выполняет работу, подразумевается команда разработчиков, которая делает всю работу по проектированию, построению и тестированию реализуемых элементов задела спринта. А владелец продукта и Scrum-мастер не делают никаких оценок. И хотя они присутствуют при оценивании элементов задела продукта, но сами этого не делают (рис. 7.4).

Роль владельца продукта состоит в том, чтобы описывать элементы задела продукта и отвечать на уточняющие вопросы, которые могут возникнуть у членов команды. Владелец продукта не должен направлять или "привязывать" команду к желательной оценке. А роль Scrum-мастера заключается в том, чтобы наставлениями и координацией содействовать оценочной деятельности.

Главная цель команды разработчиков при оценивании — определить коллективными усилиями размер каждого элемента в заделе продукта. А поскольку каждый член команды смотрит на оцениваемую историю по-своему, исходя из собственного опыта, то очень важно, чтобы все члены команды разработчиков принимали участие в оценивании.

Оценки не являются обязательствами

В связи с тем что оценки не являются обязательствами, очень важно не трактовать их как обязательства. Как правило, этот принцип касается руководителей,

перед которыми встают следующие вопросы: “Что имеется в виду, если команду просят не брать на себя никаких обязательств в отношении ее оценок, и как получить точные оценки в отсутствие всяких обязательств?”

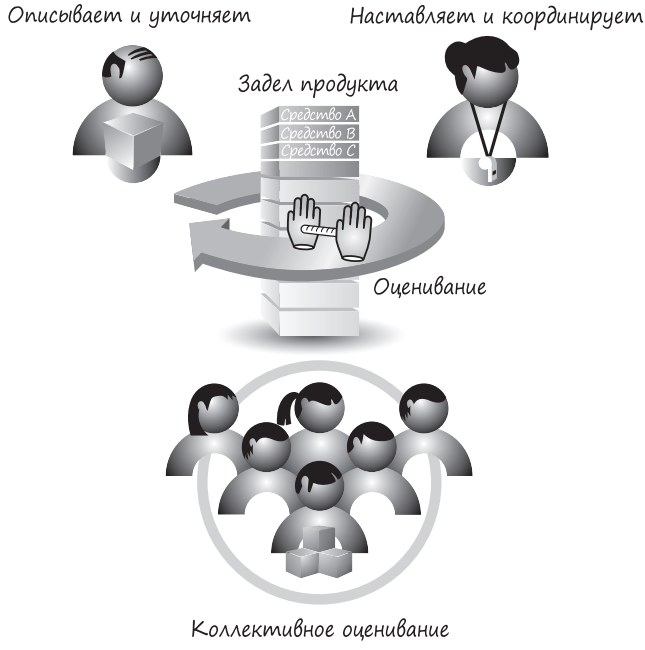


Рис. 7.4. В оценивании элементов задела продукта принимает участие вся команда

Когда на своих курсах я дохожу до этого вопроса, то демонстрирую его наглядно. С этой целью я беру наклейку для заметок и говорю: “Представьте, что я прошу вас оценить размер этой истории, а вы мне отвечаете, что она крупная”. И затем я демонстрирую ладонями своих рук размер истории, как показано на рис. 7.5, *слева*.

Рис. 7.5. Влияние обязательств на оценивание

Далее я говорю нечто вроде: “Ах да, я забыл упомянуть, что ваша премия за следующий год будет полностью зависеть от правильности ваших оценок.

Поэтому я даю вам возможность еще раз оценить данную историю". В этот момент я начинаю разводить в сторону ладони своих рук, чтобы показать, как оценки постепенно становятся все крупнее (см. рис. 7.5, *справа*). И тогда я обычно говорю следующее: "Скажите, когда мне остановиться. Мои руки слишком далеко разведены в стороны. Я ведь не баскетболист!"

Суть данного наглядного примера ясна. Если попросить людей просто оценить размер истории, то, скорее всего, можно получить правдоподобную оценку. А если сообщить им после этого, что их премии будут зависеть от правильности оценок, все они, в том числе и я сам, дадут намного более высокую оценку, чем первоначальная оценка, которая считалась правильной.

Оценки должны служить правдоподобной мерой определяемой величины. Они не должны искусственно раздуваться под внешним влиянием. Ведь это в конечном итоге приводит к раздутым графикам работ, постоянным переоценкам раздутых размеров членами команды и снижению качества управления проектом. Когда все сказано и сделано, то уже потеряно ясное понимание цифр, поскольку ими столько раз манипулировали разные люди.

Акцент на правильности, а не точности оценок

Оценки должны быть правильными, но совсем не обязательно слишком точными. Многим из нас приходилось принимать участие в разработке продуктов, где оценки были до нелепого точными. Например, одна оценка трудозатрат на выполнение проекта — 10275 человеко-часов, а другая оценка его стоимости — 132865,87 доллара.

Формирование подобных чрезмерно точных оценок является расточительством. Во-первых, напрасно тратятся усилия на получение оценки, которая может быть пересмотрена. И во-вторых, расточительство проявляется в том, что мы себя обманываем, считая, что хорошо понимаем оцениваемый предмет, хотя это совсем не так, а затем на основании этого обмана мы принимаем ответственные, но неверные дорогостоящие деловые решения. Поэтому мы должны прикладывать больше усилий для получения достаточно хороших, приближенных, но правильных оценок (рис. 7.6).

Употребление относительных размеров

Элементы задела продукта следует оценивать, употребляя относительные, а не абсолютные размеры. При этом размеры одних элементов определяются относительно размеров других элементов. Как показано на рис. 7.7, размеры одного бокала очень просто обсуждать относительно размеров другого, но не так просто определить объем жидкости, вмещающейся в каждом бокале.

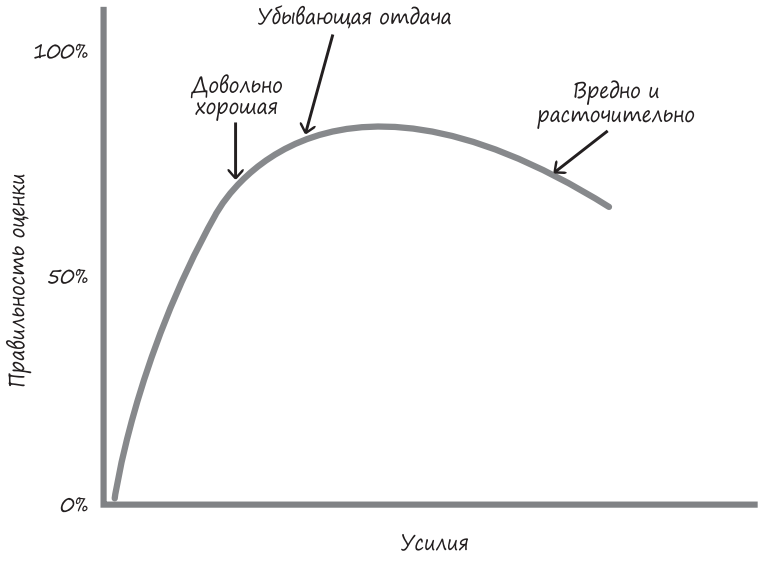


Рис. 7.6. Зависимость правильности оценок от затраченных усилий

Рис. 7.7. Оценка относительных размеров

Мои личные наблюдения убедили меня в том, что люди намного лучше оценивают относительные размеры, чем абсолютные. Это положение наглядно показывает пример, взятый из моих курсов и приведенный на рис. 7.8.

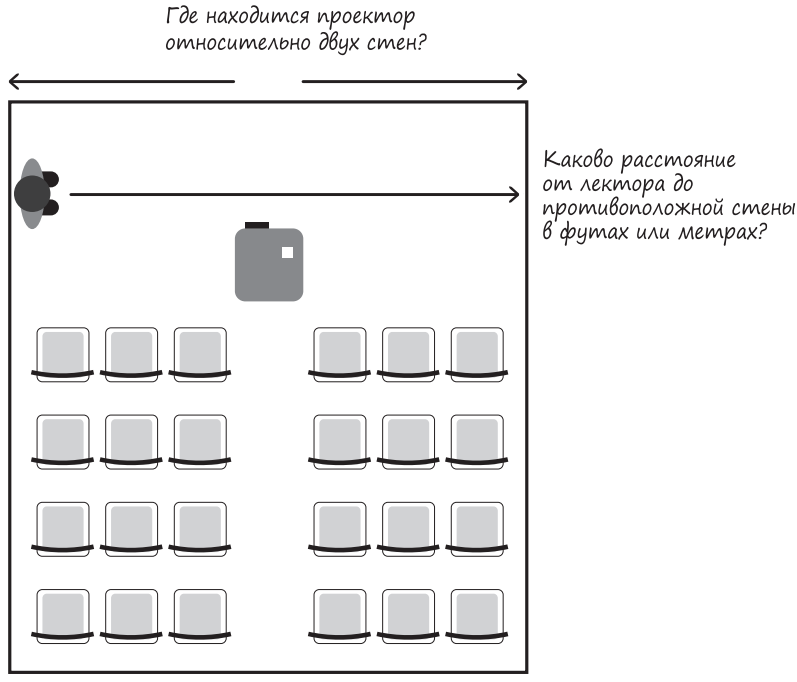


Рис. 7.8. Оценка относительных размеров в сравнении с абсолютными

Сначала я подхожу к одной боковой стене классной комнаты, обращаясь лицом к противоположной стене. Затем прошу кого-нибудь из присутствующих в комнате приблизительно оценить расстояние от меня до противоположной стены в абсолютных единицах измерения, например, в футах или метрах. (При этом я прошу не ловчить тех, кто пытается подсчитать количество плиток на потолке между двумя стенами!)

В моей классной комнате обычно имеется ЖК-проектор, установленный под потолком почти посредине между боковыми стенами. Поэтому я прошу далее кого-нибудь из присутствующих приблизительно оценить положение проектора относительно меня и противоположной стены комнаты.

Почти всегда я получаю одни и те же результаты оценок. Из 30 присутствующих в классе я получаю 27 разных ответов на вопрос о расстоянии от меня до противоположной стены, тогда как на второй вопрос 29 из 30 присутствующих в классе отвечают, что проектор находится приблизительно посредине между стенами, а 30-й просто приводит меня в замешательство ответом “на 5/11 расстояния”!

Безусловно, это не совсем точный научный эксперимент, но большинство людей быстро соглашаются с тем, что оценивают относительные размеры лучше, чем абсолютные. Для сравнения проектор иногда располагается на треть расстояния

от одной стены и на две трети расстояния от меня, но и в этих случаях результаты получаются почти такими же: большинство людей дают одну и ту же оценку относительного расстояния. Из этого можно сделать следующий вывод: если попросить людей оценить что-нибудь, то нужно выбрать такую методику, которой они хорошо владеют (оценивание относительных размеров), а не ту методику, которой они владеют плохо (оценивание абсолютных размеров).

Единицы оценивания элементов в заделе продукта

Несмотря на то что для оценивания элементов задела продукта не существует стандартных единиц, чаще для этой цели применяются очки на историю и идеальные дни. Решение выбрать одну из этих двух единиц нельзя считать ни верным, ни неверным. Должен сказать, что в 70% всех организаций, с которыми мне приходилось сотрудничать, употреблялись очки за историю, а в 30% — идеальные дни. Рассмотрим каждую из этих единиц оценивания в отдельности.

Очки за историю

В очках за историю оценивается крупность или величина элемента задела продукта. На оценивание в очках за историю оказывают влияние несколько факторов, в том числе сложность и физический размер. Крупным может считаться не только то, что имеет большой физический размер. История может представлять разработку сложного коммерческого алгоритма. Конечный результат может оказаться не очень крупным, но для его получения, возможно, придется приложить немало усилий. С другой стороны, история может быть довольно крупной физически, но не сложной технически. Допустим, что требуется обновить каждую из 60 тысяч ячеек электронной таблицы. Обновить отдельную ячейку нетрудно, но требуется автоматизировать процесс обновления. Сколько же труда потребует такая работа в спринте? Эта история будет крупной, хотя и не сложной.

Очки за историю объединяют такие факторы, как сложность и физический размер, в одну меру относительного размера. При этом преследуется цель сравнить истории и сделать, например, следующий вывод: если история создания заявки оценивается в 2 очка, то история поиска заявок — в 8 очков. В данном случае подразумевается, что история поиска превышает по размеру историю создания в четыре раза.

В примере, приведенном в начале этой главы, рассматривался подход, который состоял в том, чтобы сначала оценить размеры элементов задела продукта, а затем рассчитать продолжительность работы, разделив сумму размеров на среднюю скорость работы. Такие единицы оценивания, как очки за историю,

в конечном итоге употребляются для расчета времени (или продолжительности) работ, и поэтому очки за историю должны отражать затраты труда на реализацию истории с точки зрения команды разработчиков.

Идеальные дни

В качестве альтернативной единицы оценивания элементов задела продукта можно употреблять идеальные дни, которые обозначают трудозатраты в человеко-днях или человеко-часах, требующиеся для завершения истории. Идеальное время отличается от фактически затраченного времени. В идеальном случае игра в футбол состоит из двух таймов по 45 минут и продолжается 90 минут, но на практике одна длится на несколько минут больше из-за остановок в игре и добавленного судьей времени.

Как упоминалось ранее, не существует верного или неверного ответа на вопрос выбора между очками за историю и идеальными днями. Но главным аргументом против идеальных дней служит риск неверного их истолкования.

Допустим, что мы с вами присутствуем после полудня во вторник на оценочном совещании. Я показываю вам элемент из задела продукта и спрашиваю: "Какой размер этого элемента?" Вы отвечаете: "Два дня". Я говорю: "Итак, вы завершите его к концу дня в четверг". А вы говорите: "Нет, я завершу двухдневную работу после полудня сегодня и завтра [в среду]. Мне требуется целый день, чтобы войти в суть дела, и поэтому я собираюсь приступить к этому элементу задела продукта в четверг. Но поскольку я не могу посвятить этому элементу задела продукта целый день, то думаю, что мне удастся завершить его как-нибудь в понедельник". Тогда я говорю: "Не понимаю. Вы сказали, что этот элемент задела продукта займет два дня, следовательно, вы должны завершить его в четверг". Но вы говорите: "Я имел в виду два идеальных, но не календарных дня. Пожалуйста, не вписывайте мои идеальные дни в календарь. Так не годится".

В 30% организаций, с которыми мне приходилось сотрудничать и где успешно употреблялось идеальное время, описанный выше пример прокомментировали бы следующим образом: "Похоже на правду, но у нас такой проблемы неверного истолкования не существует. Когда мы говорим людям о двух днях, они знают, что это не два календарных дня". Если существует небольшой риск неверного истолкования идеального времени в вашей организации, эта единица оценивания, скорее всего, вам подойдет. А если вы считаете, что люди будут неверно трактовать идеальное время, то лучше пользуйтесь очками за историю.

Имеются и другие отличия очков за историю от идеального времени, но главное из них заключается в неверном истолковании. Учащаяся на одном из моих курсов так подытожила свое предпочтение одной из этих двух единиц оценивания, обратившись к коллегам со словами: "Послушайте, мы пользовались идеальным

временем в течение 15 последних лет, как я здесь тружусь, и оно никогда не работало. Честно говоря, мне бы хотелось попробовать что-нибудь другое".

Покер планирования

Покер планирования — это методика определения размеров элементов из задела продукта, впервые описанная Джеймсом Греннингом [Grenning, James. 2002], а затем распространенная Майком Коном [Cohn, Mike. 2006]. В основу покера планирования положено несколько важных принципов (рис. 7.9).

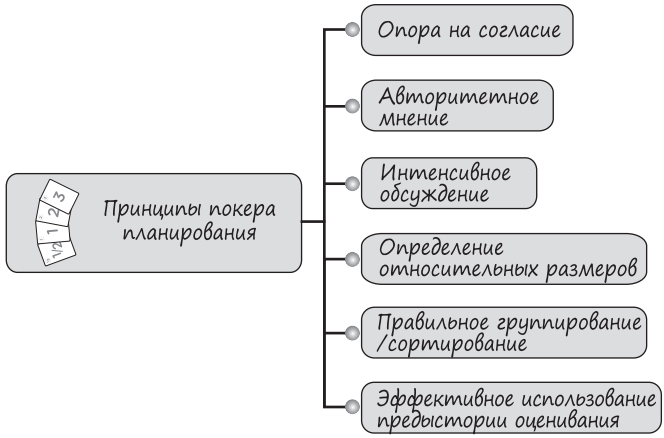


Рис. 7.9. Принципы, положенные в основу покера планирования

Покер планирования опирается на согласие при оценке затрат труда. Знающие люди (эксперты), назначенные для работы над элементами задела продукта, привлекаются к обсуждению для высказывания предположений, достигают общего понимания и определяют размеры элементов задела продукта. Путем группирования или сортировки элементов одинакового размера в ходе покера планирования вырабатываются оценки относительных размеров. Команда эффективно пользуется исторически установившейся в ней практикой оценивания элементов из задела продукта с целью упростить оценку очередного ряда этих элементов.

Оценочная шкала

Чтобы провести покер планирования, команда должна решить, какую именно шкалу или последовательный ряд чисел использовать для присваивания оценок. Ведь самое главное — правильность, а не точность оценивания. Поэтому совсем не обязательно пользоваться всеми числами оценочной шкалы, а напротив, бо́льшим количеством чисел на нижнем, более плотном конце шкалы и меньшим их количеством на верхнем, более разреженном конце шкалы.

Чаще всего употребляется оценочная шкала, предложенная Майком Коном и основанная на следующем видоизмененном ряде чисел Фибоначчи: **1**, **2**, **3**, **5**, **8**, **13**, **20**, **40** и **100**. Альтернативная шкала, которой пользуются команды, основывается на ряде чисел в степени **2**: **1**, **2**, **4**, **8**, **16**, **32 . . . .**.

Применяя подобного рода шкалу, мы группируем или сортируем сходные по размеру элементы задела продукта и присваиваем им одно и то же число на оценочной шкале. Чтобы проиллюстрировать этот принцип на конкретном примере, допустим, что мы работаем на почте и нам требуется сложить посылки сходного размера в одном ящике (рис. 7.10).

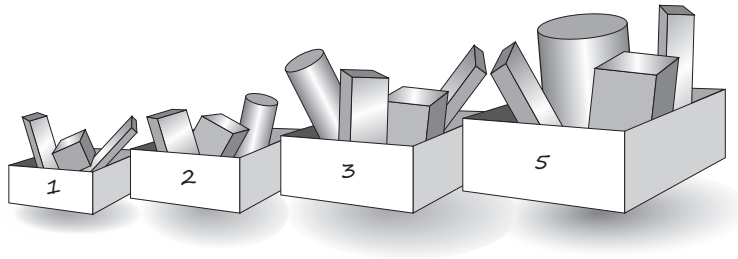


Рис. 7.10. В покере планирования применяется принцип сортировки по ящикам

Получив посылку, мы должны решить, в какой ящик ее уложить. А поскольку не все посылки находятся в одном ящике в силу их неодинакового размера, формы или веса, мы должны осмотреть посылки, уже находящиеся в ящиках, чтобы найти наиболее подходящий ящик для оцениваемой посылки. Найдя такой ящик, мы укладываем в него посылку и переходим к следующей посылке. Очевидно, что чем больше посылок мы уложим в ящиках, тем легче будет определить размеры и рассортировать по ящикам очередные посылки, поскольку у нас будет больше мест для сравнения.

Чтобы избежать чрезмерной точности оценок, мы исключаем из сортировки ящик “4”, если пользоваться оценочной шкалой в виде ряда чисел Фибоначчи. Так, если мы получаем посылку, которая оказывается крупнее по размеру, чем 2, но мельче, чем 8, то должны уложить ее в ящик “3” или “5”.

Порядок проведения покера планирования

В покере планирования принимает участие вся Scrum-команда. В сеансе этой игры владелец продукта представляет, описывает и разъясняет элементы задела продукта, а Scrum-мастер наставляет команду на успешное проведение покера планирования. Кроме того, Scrum-мастер внимательно следит за теми участниками, которые языком своих жестов или молчанием выражают несогласие, и помогает им войти в игру. А команда разработчиков коллективно вырабатывает оценки.

Каждому члену команды разработчиков раздается колода карт для покера планирования (рис. 7.11). Типичная интерпретация этих карт приведена в табл. 7.1.

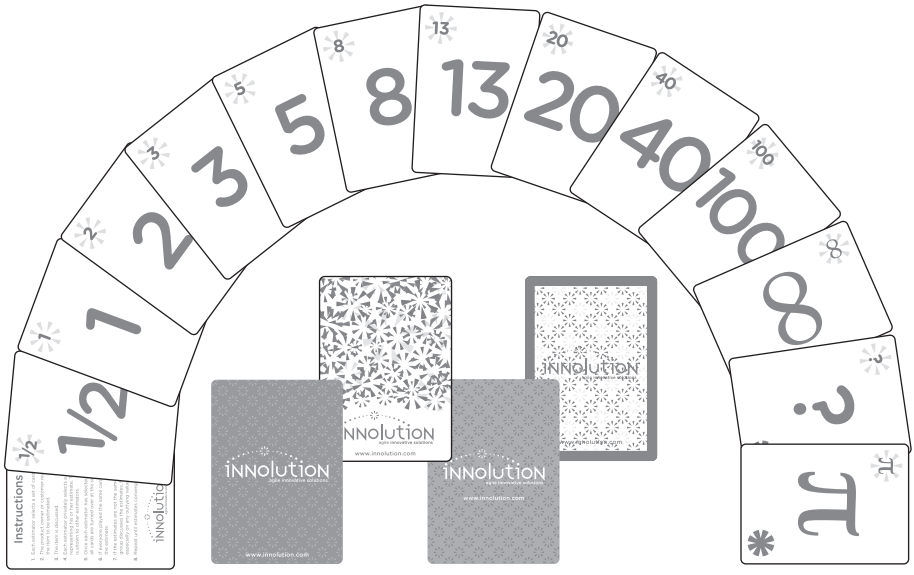


Рис. 7.11. Карты для покера планирования, употребляемые в компании Innolution

Таблица 7.1. Типичная интерпретация карт для покера планирования

Карта	Описание
`0`	Эта карта не показана на рис. 7.11, но входит в некоторые колоды и обозначает элемент, который уже завершен или настолько мал, что ему не имеет смысла присваивать номер размера
`1/2`	Служит для обозначения размеров крошечных элементов
`1`, `2`, `3`	Служат для обозначения размеров мелких элементов
`5`, `8`, `13`	Служат для обозначения размеров средних элементов. Для многих команд элемент размером `13` оказывается самым крупным из тех, что планируются в спринте. Поэтому они разбивают элемент крупнее `13` на ряд более мелких элементов
`20`, `40`	Служат для обозначения размеров крупных элементов (например, историй на уровне функционального средства или темы)
`100`	Служит для обозначения размера очень крупного функционального средства или эпической истории
∞ (бесконечность)	Служит для обозначения элемента настолько крупного размера, что ему даже не имеет смысла присваивать число

Окончание табл. 7.1

Карта	Описание
? (вопросительный знак)	Служит для обозначения того, что член команды не понимает элемент и просит у владельца продукта дополнительных разъяснений. Некоторые члены команды пользуются этой картой для отказа от оценивания текущего элемента, как правило, потому, что они настолько отстранены от него, что даже не представляют, как его оценивать. Если отказываться от оценивания еще допускается, то отказываться от участия в оценивании нельзя! Следовательно, если кто-то не считает для себя возможным дать оценку, это не дает ему право уклоняться от обсуждения или ответственности за помощь команде прийти к согласию в оценках
π (число пи)	В данном контексте знак π обозначает число **3,1415926**! А карта π служит для обозначения необходимости, по мнению члена команды, сделать перерыв на чашку чая или кофе (возможно, с бутербродом). В некоторых колодах карт вместо числа π на данной карте изображена чашка. Так или иначе, эта карта подчеркивает важный момент: члены команды могут быть вовлечены в интенсивное обсуждение в течение только ограниченного периода времени (от одного до двух часов). К этому моменту они обычно устают и должны сделать перерыв, иначе воодушевленное обсуждение перейдет в желание побыстрее покончить с оцениванием, не обращая внимания на его правильность или возможность извлечь полезный урок. Если кто-нибудь из команды выкладывает карту π, нужно сделать перерыв

Ниже приведены правила игры в покер планирования.

1. Владелец продукта выбирает элемент для оценивания из задела продукта и читает его описание команде.
2. Члены команды разработчиков обсуждают элемент и задают уточняющие вопросы владельцу продукта, который отвечает на них.
3. Каждый оценивающий тайно выбирает карту, соответствующую его оценке.
4. Как только каждый оценивающий выберет тайно нужную карту, все оценивающие одновременно показывают свои карты, обозначающие их личные оценки.
5. Если все оценивающие выберут одинаковую карту, значит, согласие достигнуто, и согласованное число становится оценкой данного элемента задела продукта.
6. Если же оценки расходятся, члены команды приступают к обсуждению, чтобы выявить любые недоразумения и высказать свои предположения.

Как правило, обсуждение начинается с просьбы к тем оценивающим, которые поставили высокие и низкие оценки, пояснить или обосновать свои оценки.

7. После обсуждения происходит возврат к п.3, и процедура повторяется до тех пор, пока не будет достигнуто общее согласие.

В покере планирования не допускается ни усреднение оценок, ни употребление любых чисел, отсутствующих на выбранной оценочной шкале или в колоде карт. Самое главное — достичь не компромисса, а согласия в оценивании командой разработчиков общего размера истории (т.е. затрат труда на ее реализацию). Как правило, это согласие может быть достигнуто за два или три раунда голосования, в течение которых члены команды сосредоточиваются на обсуждении, помогающем достичь общего понимания истории.

Преимущества покера планирования

Покер планирования объединяет разных людей, которым предстоит работать вместе, позволяя им достичь согласия относительно правильности оценок. И зачастую такое коллективное оценивание достигает лучших результатов, чем индивидуальное.

Как упоминалось ранее, в сообществе практикующих рассматриваемую здесь методику гибкой разработки бытует мнение, что оценивание элементов задела продукта не стоит затраченного труда и времени. Тем не менее интенсивное обсуждение элементов задела продукта, к которому побуждает его владелец, оказывается невероятно ценным. Как показывает мой опыт, люди получают сильную мотивацию к обдумыванию подробностей элементов задела продукта и высказыванию любых предположений, когда их просят поставить оценку размера обсуждаемого элемента.

Бо́льшая часть ценности, связанной с покером планирования, кроется в обсуждении элементов задела продукта членами команды для достижения лучшего понимания всеми участниками обсуждения. Можно надеяться, что в конечном итоге они правильно оценят элементы задела продукта, но еще важнее, что они узнают больше об этих элементах. В таком случае они получат хорошую отдачу от времени и усилий, затраченных командой на оценивание элементов задела продукта.

Что такое скорость работы

Скорость работы — это объем работ, завершенных в каждом спринте. Этот показатель измеряется путем сложения размеров отдельных элементов из задела продукта, завершенных к концу спринта. Элемент задела продукта может быть

завершен (готов) или не завершен (не готов). Владелец продукта не извлекает никакой пользы из незавершенных элементов, и поэтому в скорость работы не входят размеры частично готовых элементов задела продукта.

Скорость работы позволяет измерить итог (т.е. объем того, что выпускается), а не конечный результат (т.е. доставляемую ценность). При этом предполагается, что если владелец продукта согласился с тем, что команда должна работать над конкретным элементом задела продукта, то команда должна доставить ему некоторую ценность. Но завершение элемента задела продукта размером 8 совсем не означает доставку большей коммерческой ценности, чем завершение элемента задела продукта размером 3. Вполне вероятно, что элемент задела продукта размером 3 может иметь бо́льшую ценность и меньшую стоимость затрат, а следовательно, работа над ним начнется раньше, чем над элементом задела продукта размером 8, имеющим меньшую ценность и бо́льшую стоимость затрат.

Скорость работы употребляется по двум важным причинам. Во-первых, это очень важный показатель для планирования по методике Scrum. Как показано на рис. 7.1, при планировании на уровне выпуска его размер делится на среднюю скорость работы команды, чтобы определить количество спринтов, требующихся для завершения выпуска. И во-вторых, при планировании спринта скорость работы команды используется в качестве входных данных, чтобы выяснить способность команды взять на себя обязательство выполнить работу в течение спринта (подробнее об этом — в главе 19).

Кроме того, скорость работы служит для команды диагностическим показателем, чтобы оценивать свою работу и совершенствоваться в применении методики Scrum с целью доставлять потребительскую ценность. Наблюдая во времени за скоростью своей работы, команда может составить представление, как изменения в конкретном процессе оказывают влияние на доставку ощутимой потребительской ценности.

Расчет скоростного диапазона

Скорость работы оказывается наиболее полезной для целей планирования, если она выражается в виде диапазона, аналогичного следующему: как правило, команда способна выполнять в каждом спринте объем работ в пределах от 25 до 30 очков. Скоростной диапазон позволяет дать правильную, но не чрезмерно точную оценку возможностей команды.

С помощью скоростного диапазона можно найти правильные ответы на следующие вопросы: “Когда работа будет завершена?”, “Сколько элементов можно завершить?” или “Во что все это обойдется?” Большинство подобных вопросов встают на ранней стадии проектных работ, когда имеется минимум информации о разрабатываемой продукции, и поэтому найти точные ответы на эти вопросы

невозможно. А с помощью скоростного диапазона можно вполне справиться с возникающей при этом неопределенностью (рис. 7.12).

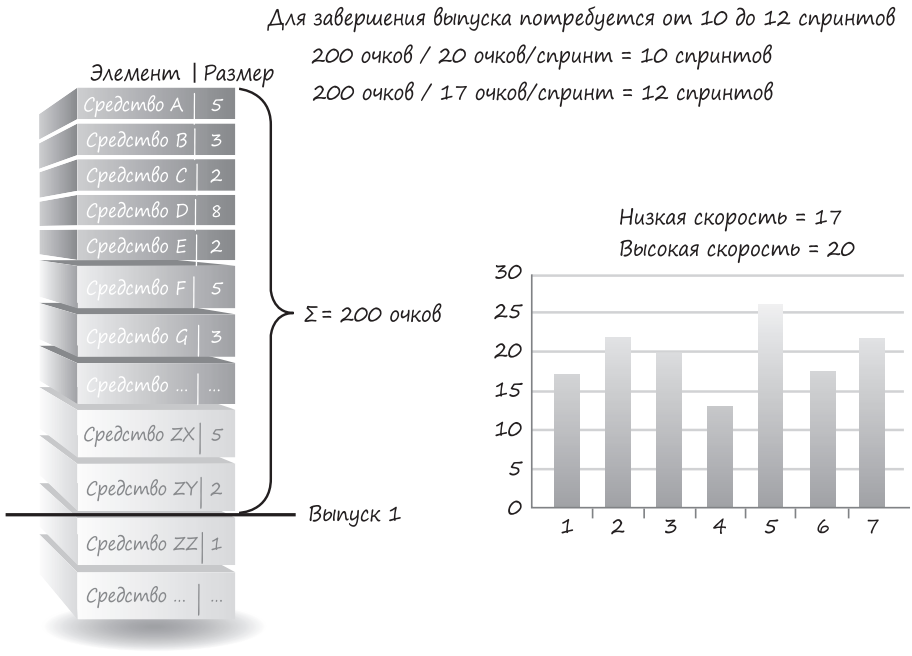


Рис. 7.12. Расчет и применение скоростного диапазона

В данном примере (а по существу, в пересмотренном примере, приведенном на рис. 7.1) вместо объявления точного спринта, в течение которого должны быть завершены все элементы из выпуска, о чем, скорее всего, можно только догадываться, в качестве ответа на поставленный вопрос предоставляется скоростной диапазон. Чтобы рассчитать этот диапазон, потребуются две скорости работы команды. Так, если разделить размер выпуска на более высокую скорость работы команды, то получится наименьшее количество требующихся спринтов. А если разделить размер выпуска на более низкую скорость работы команды, то получится наибольшее количество требующихся спринтов.

Произведя простые математические расчеты вроде верхнего и нижнего среднего, 90%-ных доверительных интервалов и т.д., можно легко получить два показателя скорости работы, исходя из хронологических данных об этом показателе (в данном примере — 17 и 20 спринтов). В главе 18 приводятся дополнительные подробности о выполнении подобных расчетов, чтобы ответить на вопросы, когда, как и сколько можно сделать.

Прогнозирование скорости работы

В приведенных выше примерах предполагалось, что у команды имеются хронологические данные о скорости ее работы, исходя из которых можно спрогнозировать скорость работы на будущее. Очевидно, что одно из преимуществ существования долго работающих команд заключается в том, что они могут без особого труда получить эти столь полезные хронологические данные (более подробно преимущества долго работающих команд рассматриваются в главе 11). Но как быть в том случае, если имеется новая команда, члены которой еще не работали вместе, а следовательно, у них отсутствуют хронологические данные? В таком случае придется прибегнуть к прогнозированию скорости работы.

Чтобы спрогнозировать скорость работы, можно, в частности, поручить команде спланировать спринт с целью выяснить, какие элементы задела продукта она может взять обязательство выпустить в течение одного спринта. Если эти обязательства окажутся вполне обоснованными, то останется только сложить вместе размеры тех элементов задела продукта, завершить которые команда обязалась, и пользоваться полученной в итоге суммой в качестве прогнозируемой скорости работы.

На самом деле требуется лишь скоростной диапазон, и поэтому команде можно поручить спланировать два спринта, чтобы воспользоваться одним показателем оцененной скорости работы в качестве верхнего предела, а другим — в качестве нижнего предела (обе оценки, скорее всего, будут разными). С другой стороны, можно произвести интуитивную коррекцию одной из оценок скорости работы, исходя из хронологических данных о работе других команд, преобразовав таким образом одну оценку в диапазон из двух оценок.

Как только команда выполнит первый спринт и появится фактический показатель скорости ее работы, прогнозируемую скорость работы можно отбросить и далее пользоваться фактической скоростью. А после того, как накопятся хронологические данные о работе команды, следует рассчитать средние показатели скорости ее работы или применить другие статистические методы обработки этих данных, чтобы получить надежный скоростной диапазон (другие тому примеры см. в [Cohn, Mike. 2009]).

Влияние на скорость работы

Любопытно, должна ли скорость работы команды постоянно расти со временем? Один из руководителей как-то сказал мне следующее: “В прошлом году скорость работы моей команды составила в среднем 30 очков за спринт. А в этом году я ожидаю, что она достигнет 35 очков за спринт”. Этот руководитель считает, что скорость работы команды должна соответствовать тенденции

1 на рис. 7.13. Он, вероятно, полагает, что если команда постоянно выполняет обследование и адаптацию, т.е. непрерывно совершенствуется, то скорость ее работы должна все время расти.

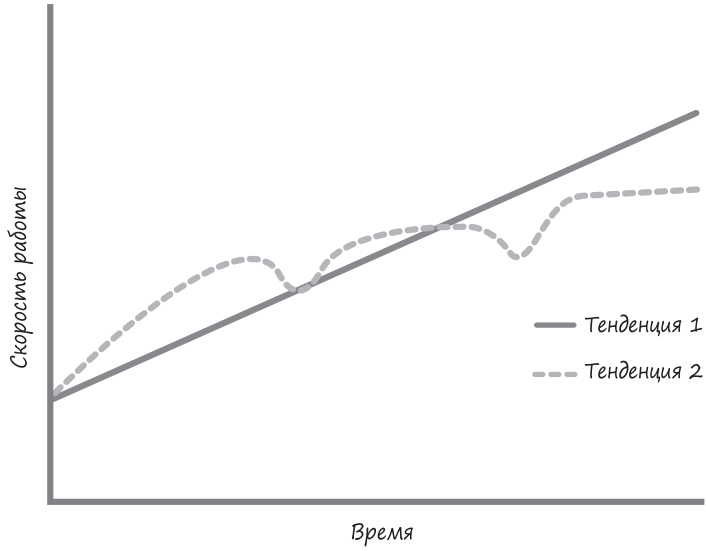


Рис. 7.13. Изменение скорости работы команды со временем

Вполне возможно, что если команда резко попытается усовершенствоваться, сосредоточившись на выпуске функциональных средств в соответствии с надежным критерием готовности и малым техническим долгом (см. главу 8), то скорость ее работы заметно возрастет. По крайней мере, она будет расти до определенного момента, а затем ее рост, скорее всего, остановится аналогично тенденции 2 на рис. 7.13.

Но если скорость работы команды выровняется, то это совсем не означает, что у нее не останется больше потенциала для роста. Scrum-команда и ее руководители могут довести скорость работы до следующего установившегося уровня самыми разными способами. С одной стороны, внедрение новых инструментальных средств или повышение квалификации членов команды может положительно сказаться на скорости ее работы. А с другой стороны, руководители могут стратегически поменять состав команды в надежде на то, что эти перемены приведут в конечном итоге к увеличению скорости работы. Безусловно, руководители должны действовать осторожно, меняя состав команды, чтобы это не привело к снижению скорости ее работы.

Несмотря на то что внедрение новых инструментальных средств или смена состава команды разработчиков может оказать положительное влияние на скорость ее работы, эти меры обычно приводят к падению скорости работы

на первоначальном этапе, когда команда приспосабливается к переменам (см. тенденцию 2 на рис. 7.13). А после этого временного падения скорость работы, вероятно, начнет расти до следующего установившегося уровня, на котором она останется до тех пор, пока новые перемены не поспособствуют тому, что скорость работы достигнет очередного установившегося уровня.

Для увеличения скорости работы можно, конечно, прибегнуть к следующей очевидной мере: работать дольше. Если много работать сверхурочно, то первоначально скорость работы может возрасти (см. участок "Сверхурочные работы" кривой на рис. 7.14).

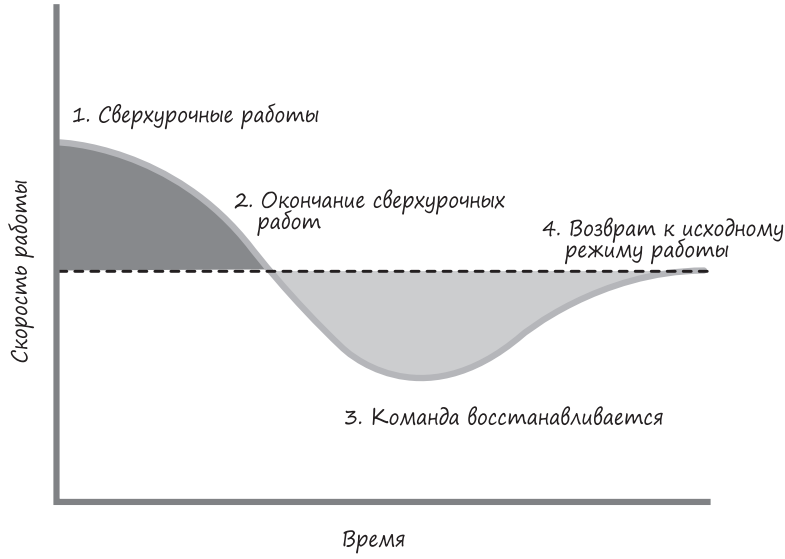


Рис. 7.14. Влияние сверхурочного труда на скорость работы (кривая построена на основании иллюстрации, взятой из [Cook, Daniel. 2008])

После такого увеличения скорости работы, скорее всего, наступит резкое ее падение наряду со снижением качества. Даже по окончании периода сверхурочных работ команде потребуется некоторое время, чтобы восстановиться, прежде чем вернуться к ее исходной скорости работы. Мне приходилось видеть примеры, когда впадина на кривой, приведенной на рис. 7.14, оказывалась в течение периода восстановления больше, чем гребень в течение сверхурочных работ. Из всего сказанного выше можно сделать следующий вывод: если много работать сверхурочно, это дает краткосрочное преимущество, но зачастую оно сводится на нет долгосрочными последствиями.

Злоупотребление скоростью работы

Скорость работы служит в качестве средства планирования и диагностического показателя возможностей команды. Она не применяется в качестве количественного показателя для оценки производительности труда в команде. Если злоупотреблять скоростью работы именно таким образом, то она может побуждать к расточительному и опасному поведению.

Допустим, что команде, имеющей высокую скорость работы, решено дать самую большую премию. На первый взгляд это решение кажется вполне обоснованным. Ведь команда с высокой скоростью работы должна завершать бо́льшую часть работы в каждом спринте, не так ли? Так почему бы не вознаградить ее за такое поведение?

Но если сравнивать команды, которые не определяют размеры элементов заделов своих продуктов на общем основании, что вполне вероятно, то сравнивать их количественные показатели не имеет смысла. Допустим, что команда А присваивает 5 очков элементу задела продукта, тогда как команда В — 50 очков тому же самому элементу. Вряд ли стоит сравнивать эти команды по скорости работы. Команда В работает в десять раз быстрее, чем команда А, несмотря на то, что обе команды на самом деле выполняют приблизительно одинаковый объем работы в каждом спринте.

Как только в команде А заметят этот недостаток, ее члены будут стремиться "обыграть" систему, чтобы добиться более высоких показателей в скорости их работы. Это проще всего сделать, изменив шкалу, которой команда пользуется для оценивания элементов задела продукта. В итоге команда А оценит в 500 очков тот же самый элемент, который она раньше оценивала в 5 очков. Такое поведение называется *очконадувательством* и не служит никакой другой цели, кроме ориентации команды на ложную систему оценок. Даже если команды и пользуются одинаковыми единицами для оценивания элементов задела продукта, но установлена система вознаграждения за большие числовые показатели, то в конечном итоге это приведет лишь к очконадувательству.

Еще хуже, чем очконадувательство, когда команды начинают идти по пути наименьшего сопротивления, чтобы сделать "побольше", стремясь достичь более высоких скоростей работы. Такое поведение обычно приводит к заметному увеличению технического долга.

В конце рабочего дня скорость работы следует оценивать по тому, как она способствует точному выполнению плана и помогает команде внутренне совершенствоваться. А любое другое употребление скорости работы способствует неверному поведению.

Заключение

В этой главе обсуждались способы оценивания объемов, измерения скорости и расчета продолжительности работ. Сначала в ней было показано, каким образом производится оценивание на уровне элементов задела портфеля заказов, задела продукта и отдельных задач. Затем обсуждались важные понятия, связанные с оцениванием элементов задела продукта, включая очки за историю и идеальные дни. Далее была описана методика покера планирования, широко применяемая для оценивания элементов задела продукта.

После оценивания в этой главе была рассмотрена скорость работы и показано, как ее следует употреблять. При этом подчеркивалось, что скорость работы приносит наибольшую пользу, если она выражается в виде диапазона, а не одного числа. Вкратце были рассмотрены способы прогнозирования скорости работы новой команды. И в заключение главы обсуждалось, как зачастую злоупотребляют скоростью работы. А следующая глава посвящена понятию технического долга и тому, как обращаться с ним, применяя методику Scrum.

Глава 8
Технический долг

В этой главе обсуждается понятие технического долга. Сначала в ней дается определение технического долга, которое охватывает наивный, неизбежный и стратегический долг. Затем исследуются типичные причины появления технического долга и последствия его накапливания в большом количестве. После этого описываются следующие три вида деятельности, связанные с технических долгом: управление накапливанием, выявление и обслуживание технического долга. Особое внимание уделяется организации этих видов деятельности по методике Scrum.

Краткий обзор

Впервые понятие *технического долга* было сформулировано Уэрдом Каннингхемом [Cunningham, Ward. 1992]. Он дал техническому долгу такое определение:

> *Поставить код в первый раз — все равно что влезть в долг. Небольшой долг ускоряет разработку, при условии, что он сразу же возмещается переписыванием кода... Опасность возникает, когда долг не возмещается. Каждая минута, потраченная на недостаточно верный код, начисляется как процент по этому долгу. Целые проектные организации могут остановиться под бременем долга не сведенной во едино реализации ...*

Каннингхем употребил метафору технического долга, чтобы пояснить своей рабочей команде причины, по которым программное обеспечение полезно разрабатывать быстро, чтобы получить нужную реакцию. Но при этом он обратил внимание на следующие ключевые моменты: команда в частности и организация вообще должны внимательно следить за возмещением долга по мере того, как они начинают лучше понимать область своей деятельности, а разработка и реализация системы должны развиваться в направлении лучшего восприятия этого понимания.

С момента внедрения термина *технический долг* в начале 1990-х годов в отрасли разработки программного обеспечения были допущены некоторые вольности в определении, которое дал Каннингхем. Ныне под техническим долгом подразумеваются намеренно избираемые кратчайшие пути к цели и многие

неверные действия, наносящие вред программным системам. К их числу относятся следующие.

- Негодное (плохое) проектирование, которое было оправдано раньше, но не теперь, когда произошли важные изменения в деловой и технологической сфере.
- Дефекты — известные недостатки в программном обеспечении, на устранение которых еще не выделено время.
- Недостаточное покрытие тестами тех участков, которые заведомо должны тестироваться тщательнее, но этого не делается.
- Чрезмерное тестирование вручную, когда тесты должны быть на самом деле автоматизированы.
- Плохое управление интеграцией и выпусками — выполнение этих видов деятельности, приводящее к напрасной трате времени и чреватое ошибками.
- Отсутствие опыта работы с целевыми платформами. Например, имеются приложения, написанные на COBOL для большой ЭВМ, но больше нет в достаточном количестве опытных специалистов, программирующих на COBOL.
- И многое другое, поскольку термин *технический долг* ныне употребляется в качестве заменителя многомерной задачи.

Каннингхем не имел намерения обозначить техническим долгом незрелость членов команды или деловых людей или же недостатки в технологическом процессе, приводящие к негодному проектированию, неправильным нормам инженерной практики и недостаточному тестированию. Такого рода долг можно ликвидировать должным обучением, правильным представлением о том, как следует применять нормы инженерной практики, а также здравым подходом к принятию деловых решений. В силу легкомысленного, а зачастую и случайного характера формирования такого рода долга он называется *наивным техническим долгом*. В литературе он называется также *опрометчивым долгом* [Fowler, Martin. 2009], *неумышленным долгом* [McConnell, Steve. 2007] и *беспорядком* [Martin, Robert C. 2008].

Кроме того, имеется *неизбежный технический долг*, который обычно непредсказуем и непредотвратим. Например, наше представление о качественном проектировании возникает в результате выполнения проектных работ и построения ценных для пользователей функциональных средств. Но мы не можем заранее предвидеть, каким образом продукт и его проектное решение должны развиваться со временем. Следовательно, принятые ранее решения о проектировании и реализации, возможно, придется изменить по мере завершения циклов обучения и приобретения знаний в результате утвержденного обучения. Требующиеся изменения на участках воздействия являются признаками неизбежного технического долга.

В качестве другого примера допустим, что у нас имеется лицензированный сторонний компонент, который предполагается использовать в продукте, а также интерфейсы для сопряжения с этим компонентом, которые развиваются со временем. Продукт, который раньше нормально функционировал с этим сторонним компонентом, накапливает технический долг не по нашей вине. И хотя такой долг вполне предсказуем, поскольку вполне благоразумно допустить, что поставщик стороннего компонента внесет со временем изменения в интерфейсы своего компонента, тем не менее, избежать этого долга не удастся, потому что невозможно заранее предвидеть, в каком направлении разработчики будут далее совершенствовать сторонний компонент.

И последней разновидностью технического долга является *стратегический технический долг*. Такого рода долг служит инструментальным средством, помогающим организациям лучше и эффективнее определять экономические показатели, когда принимаются важные и зачастую срочные решения. Например, в организации может быть намерено принято стратегическое решение избрать кратчайшие пути во время разработки продукции, чтобы достичь важной краткосрочной цели вроде оперативного выпуска продукции на рынок. Кроме того, организации с ограниченным капиталом может просто не хватить средств на завершение продукта, и поэтому выпуск продукта с техническим долгом при урезанных первоначальных затратах на разработку и получение дохода для самофинансирования дальнейшей разработки может оказаться для нее единственным выходом из положения, чтобы избежать краха перед развертыванием продукта.

Независимо от того, каким образом технический долг был накоплен, он служит сильной метафорой, поскольку дает возможность лучше осознать и уяснить важный вопрос. Эта метафора вполне согласуется с представлениями деловых людей, хорошо разбирающихся в том, что такое финансовый долг. Когда они слышат о техническом долге, они способны быстро оценить проницательность данной метафоры, проведя параллели с финансовым долгом и осознав, что технический долг потребует выплаты процентов в виде дополнительных затрат труда на разработку недостающих функциональных средств. С одной стороны, проценты можно выплачивать и дальше, не решая возникшие проблемы, а с другой — погасить основную часть долга, реорганизовав, например, код, чтобы сделать его более простым и удобным для модификации.

Последствия накапливания технического долга

По мере нарастания технического долга последствия становятся все более серьезными. Обсудим некоторые из наиболее примечательных последствий накапливания большого технического долга (рис. 8.1).

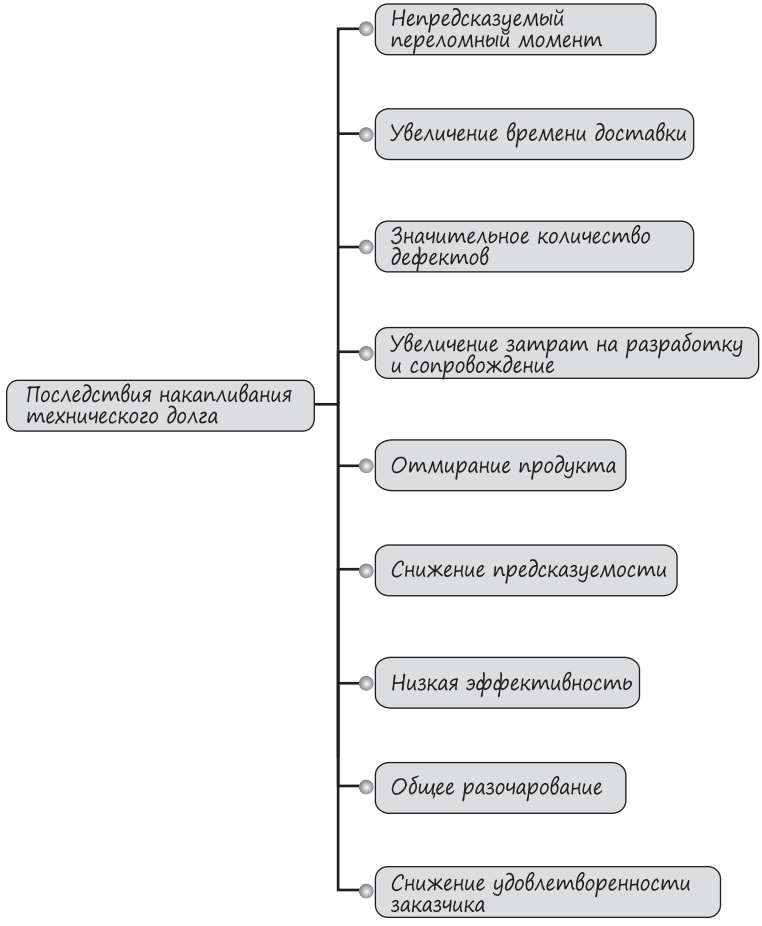


Рис. 8.1. Последствия накапливания технического долга

Непредсказуемый переломный момент

Важным свойством технического долга является его непредсказуемое, нелинейное нарастание. Каждая часть, добавляемая в общий фонд существующего технического долга, способна нанести намного более существенный вред, чем может первоначально подразумевать величина этой новой части. Технический долг достигает своего рода "критической массы", когда продукт доходит до переломного момента и становится неуправляемым или хаотичным. В такой переломный момент даже незначительные изменения в продукте становятся главными причинами неопределенности. Данное нелинейное свойство технического долга представляет значительный коммерческий риск, поскольку заранее неизвестно, когда грянет гром. Но когда это произойдет, все последствия только усугубятся.

Увеличение времени доставки

Принять на себя технический долг означает взять взаймы время, требующееся для последующей работы. Чем больше долг сегодня, тем медленнее скорость работы завтра. А когда замедляется скорость работы, требуется больше времени на доставку новых функциональных средств заказчикам и устранение дефектов в продукте. Таким образом, при наличии большого технического долга время между доставками готовых компонентов на самом деле увеличивается, а не сокращается. При постоянной конкуренции на рынке технический долг активно противодействует самым благим намерениям.

Значительное количество дефектов

Продукты со значительным техническим долгом становятся сложными, затрудняя их правильную разработку. Нарастание сложности дефектов может привести к серьезным отказам, возникающим в продукте с тревожной частотой. Такие отказы служат главным препятствием для нормального хода проектных работ, повышающих ценность продукции. Кроме того, издержки на устранение многих дефектов поглощают время, отводимое на создание дополнительных функциональных средств. В какой-то момент мы начинаем вязнуть в болоте устраняемых дефектов, но настолько заняты этим, что даже не видим, как выбраться из болота, в котором оказались.

Увеличение затрат на разработку и сопровождение

По мере увеличения технического долга растут затраты на разработку и сопровождение. То, что раньше можно было сделать просто и дешево, теперь сложно и дорого. При наличии постоянно накапливающегося технического долга даже небольшие изменения обходятся недешево (рис. 8.2). Когда кривая затрат при большом техническом долге начинает резко возрастать, как показано на рис. 8.2, достигается критическая масса технического долга и наступает переломный момент.

Кроме того, увеличение затрат может привести к изменению экономических показателей, по которым можно судить, следует ли продолжать разработку функционального средства или устранить дефект. Создание функционального средства или устранение дефекта, которое можно было бы осуществить с малыми затратами при наличии небольшого технического долга, может обойтись дорого при наличии большого технического долга. В результате увеличения затрат продукты становятся менее приспособленными к той развивающейся среде, в которой они должны существовать.

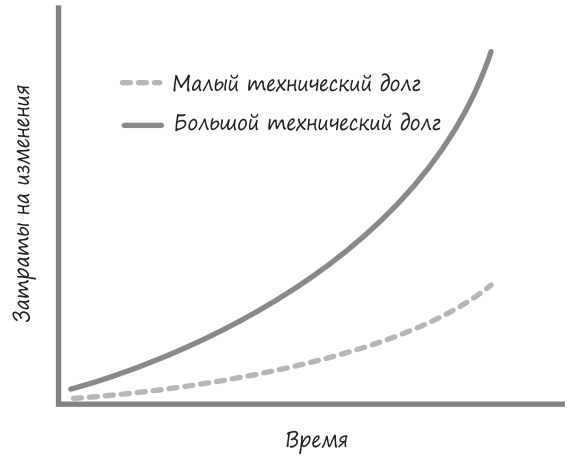


Рис. 8.2. Кривая затрат на изменения под воздействием технического долга

Отмирание продукта

По мере того, как прекращается ввод новых функциональных средств или устранение дефектов, способных омолодить стареющий продукт, последний становится все менее привлекательным для текущих и потенциальных потребителей. В итоге продукт начинает отмирать и просто перестает существовать как нечто ценное для большинства потребителей. А те из них, которые продолжают им пользоваться, как правило, остаются еще некоторое время привязанными к нему. Но как только появится первая же возможность перейти на другой появившийся продукт, они не преминут ею воспользоваться!

Снижение предсказуемости

Если у продукта накопился большой технический долг, то сделать какие-то прогнозы практически невозможно. Например, оценки оказываются неверными даже у опытных членов команды. А все дело в том, что, когда приходится иметь дело с отягощенным долгами продуктом, возникает слишком большая неопределенность при попытке оценить продолжительность каких-нибудь работ. Следовательно, способность команды брать на себя обязательства и делать обоснованные прогнозы относительно их выполнения серьезно ослабляется. Руководители предприятия перестают доверять словам разработчиков, а заказчики — словам руководителей!

Низкая эффективность

К сожалению, по мере роста технического долга приходится наблюдать постепенное снижение эффективности разработки, а следовательно, и ожиданий относительно того, что можно сделать. Разумеется, пониженные ожидания начинают распространяться по всей цепочке создания ценности, и в конечном итоге снижается эффективность деятельности организации в целом.

Общее разочарование

Человеческий фактор в неудачных последствиях накопления большого долга состоит в том, что все, кто задействован в цепочке создания ценности, оказываются разочарованными. Накапливание всех этих мелких, но неприятных упрощений ради быстрого достижения поставленной цели делает работу над продуктом мучительной. И в конечном итоге удовольствие от разработки исчезает, а на смену ему приходит рутинное разрешение вопросов, которые никто не хочет (или не должен) решать. Энтузиазм людей гаснет. Грамотные члены команды начинают искать более привлекательные возможности для приложения своих знаний и опыта. И если они находятся в лучшем положение, чтобы каким-то образом справиться с техническим долгом, то их уход еще больше осложняет положение тех, кто остается. Моральная обстановка резко ухудшается по мере повышения напряженности в коллективе.

Технический долг лишает удовольствия от труда не только инженерных работников, но и деловых людей. И в связи с этим у них возникают следующие вопросы: как долго мы будем продолжать брать на себя коммерческие обязательства, которые нельзя выполнить, и что делать с бедными заказчиками, которые пытаются применить наш отягощенный долгами продукт в своей деятельности? Они слишком быстро начинают уставать от повторяющихся сбоев в продукте и нашей неспособности выполнить любые обещания, которые мы даем. Таким образом, на смену доверию, некогда существовавшему во всей цепочке создания ценности, приходит разочарование и недовольство.

Снижение удовлетворенности заказчика

Удовлетворенность заказчика снижается по мере роста его разочарования. Следовательно, ущерб, наносимый техническим долгом, затрагивает не только команду разработчиков, но и всю проектную организацию в целом. Хуже того, последствия накопления технического долга могут существенно повлиять на заказчиков и их отношение к исполнителям.

Причины технического долга

Напомним, что технический долг проявляется в трех основных формах, каждая из которых имеет свои коренные причины. В частности, неизбежный долг накапливается независимо от принятых предупредительных мер. Наивный долг является результатом незрелости членов команды, руководства организации или технологического процесса. А стратегический долг берется в тех случаях, когда выгоды от его накапливания значительно превышают затраты на его погашение.

Нажим с целью уложиться в срок

Но стратегический и наивный технический долг нередко возникает вследствие нажима, который руководство организации оказывает на разработчиков с целью уложиться в приближающийся срок, как показано на рис. 8.3, основанном на материалах, взятых из [Mar, Kane. 2006]. По вертикальной оси на рис. 8.3 отложен объем работ, которые требуется выполнить к желательному сроку выпуска, отложенному по горизонтальной оси. Линия, проведенная между заданным объемом работ и желательным сроком выпуска, обозначает постоянную запланированную скорость, с которой проектные работы должны быть выполнены к желательной дате выпуска. Работа с запланированной скоростью преследует цель своевременно завершить разработку высококачественных функциональных средств, сведя к минимуму накапливание технического долга.

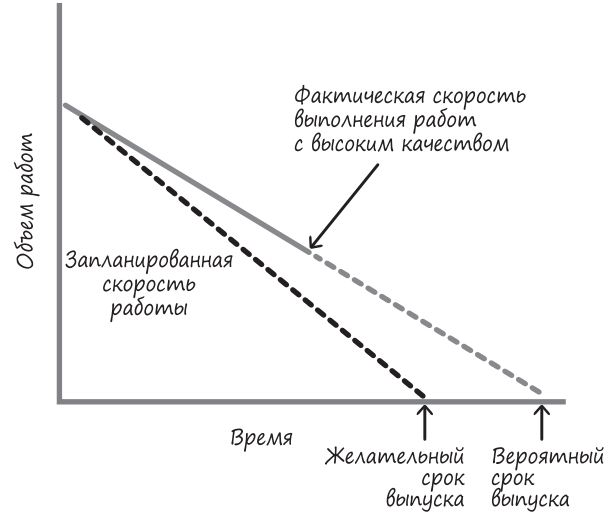


Рис. 8.3. Нажим с целью уложиться в срок

Но как только мы приступаем к работе, фактическая скорость работы, требующаяся для достижения высококачественных результатов, оказывается ниже запланированной скорости. Если мы продолжим получать результаты с фактической скоростью, то не уложимся в желательный срок и завершим работу к вероятной дате выпуска.

Попытка ошибочно увеличить скорость работы

На данной стадии нужно принять следующее коммерческое решение: требуется ли сократить объем работ, чтобы уложиться в желательный срок, или же выделить больше времени в графике выполнения работ, чтобы адаптировать доставку к вероятной дате выпуска? К сожалению, во многих случаях руководители организаций отвергают обе эти возможности и приказывают команде уложиться в желательный срок, завершив к нему все функциональные средства. В подобной ситуации команде, выполняющей намеченные работы, приказывают увеличить скорость работы, чтобы уложиться в желательный срок (рис. 8.4).

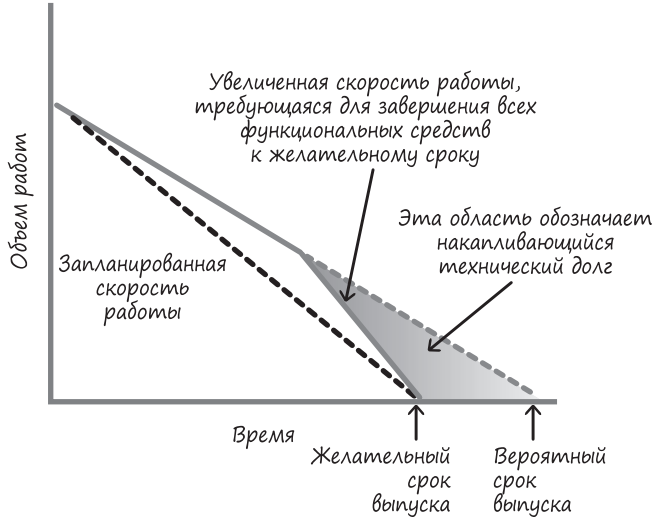


Рис. 8.4. Накапливание технического долга с целью уложиться в необоснованные объемы работ и сроки

Работая с такой увеличенной скоростью, команда будет вынуждена принять решение намеренно взять на себя технический долг, т.е. пойти по пути наименьшего сопротивления, чтобы работать быстрее и уложиться в желательный срок. Такое решение, скорее всего, приведет к тому, что проектное решение окажется не таким качественным, как должно быть, а некоторые виды тестирования (например, нагрузочное тестирование) будут отложены. В конечном итоге

накапливается технический долг, обозначенный треугольной областью на рис. 8.4. В этой области оказывается вся работа, которую нужно было выполнить, но для этого не хватило времени.

Миф об увеличении скорости работы за счет сокращения тестирования

Весьма распространен следующий миф: тестирование влечет за собой дополнительные издержки, и если сократить его, то можно увеличить скорость работы (рис. 8.5).

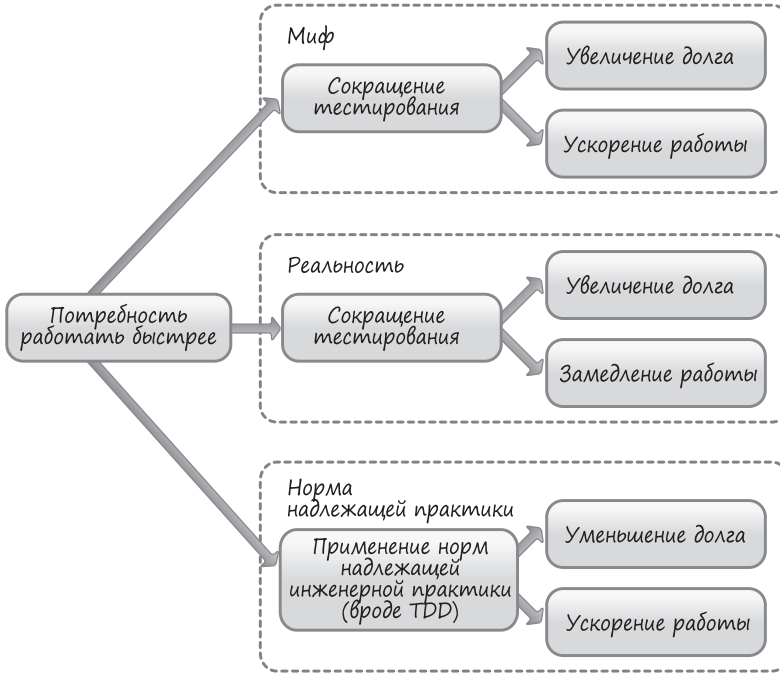


Рис. 8.5. Миф, реальность и норма надлежащей практики влияния тестирования на скорость работы

Реальность такова, что сокращение тестирования приводит не только к увеличению технического долга, но и вынуждает работать медленнее, поскольку недостатки проявляются лишь впоследствии, когда потребуется больше времени для их устранения. Опытные команды достигают качественных результатов быстрее и с меньшим техническим долгом, когда тестирование является неотъемлемой частью процесса разработки. С этой целью они применяют такие нормы надлежащей инженерной практики, как разработка посредством тестирования (TDD), когда разработчик пишет и автоматизирует небольшие блочные тесты,

прежде чем написать небольшой фрагмент кода, который должен обеспечивать прохождение теста [Crispin, Lisa, and Janet Gregory. 2009].

Образование одних долгов из других

Будущий технический долг быстро образуется на основании уже существующего технического долга. И как только образуется новый технический долг, начинают проявляться экономически пагубные последствия. Так, на рис. 8.6 показаны последствия выпуска 2 на основании технического долга из выпуска 1.

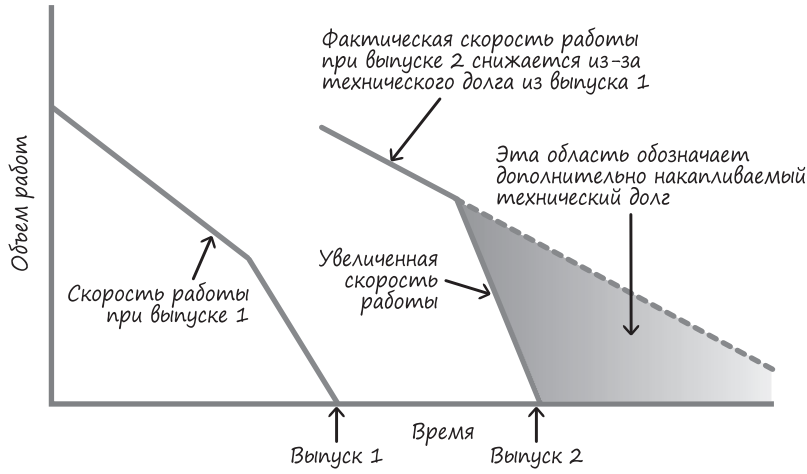


Рис. 8.6. По мере увеличения технического долга уменьшается скорость работы

Как показано на рис. 8.6, фактическая скорость работы при выпуске 2 оказывается ниже, чем при выпуске 1. Очевидно, что при такой скорости снова не удастся уложиться в намеченный желательный срок. И в этом случае руководство организации настаивает, чтобы команда успела завершить все функциональные средства к желательному сроку выпуска. А в конечном итоге технический долг накапливается еще больше.

Если и дальше следовать этому образцу, то в конечном итоге линия, обозначающая скорость работы, может стать горизонтальной. Это будет состояние, в котором технический долг в системе настолько велик, что скорость работы фактически равна нулю. В итоге будет выпущен такой продукт, вносить какие-либо изменения в который будет просто страшно, поскольку изменение на одном участке могут повлечь за собой нарушения нормальной работы 18 других совершенно не связанных участков продукта. Хуже того, подобные нарушения на 18 других участках невозможно предвидеть. И разумеется, для определения момента этих нарушений у разработчиков, как всегда, не окажется подходящей

среды тестирования. Но они могут быть уверены, что заказчики не преминут сообщить им о подобных нарушениях!

Если накопится большой технический долг, то все перечисленные ниже варианты выхода из сложившегося положения окажутся неудачными. Когда такие варианты выбора появятся на горизонте, очень важно организовать правильное управление техническим долгом, прежде чем он выйдет из-под контроля.

- Ничего не делать, и тогда положение только ухудшится.
- Вложить средства в сокращение технического долга, который может потреблять все больше и больше ценных ресурсов, выделяемых для разработки продукта.
- Объявить техническое банкротство, погасить технический долг и заменить обремененный долгами продукт новым продуктом со всеми затратами и рисками, связанными с его разработкой.

Технический долг должен поддаваться управлению

Как и финансовый, технический долг должен поддаваться управлению. Очень важно понять, что ни один продукт не свободен от долгов, и поэтому не стоит стремиться к состоянию продукта, свободного от долгов. Даже если бы это и было возможно, свобода от долгов могла бы оказаться экономически неоправданной. Тем не менее следует стремиться к тому, чтобы технический долг оставался настолько малым, чтобы не оказывать существенного влияния на последующую разработку продукции.

Для правильного управления техническим долгом от инженерных работников и деловых людей требуется решение, взвешенное технически и коммерчески. И это одна из причин, по которым в каждой Scrum-команде имеется владелец продукта. Наличие владельца продукта в Scrum-команде позволяет принимать взвешенное решение как с деловой, так и с технической точки зрения, чтобы достичь оптимального экономического компромисса. Как поясняется в главе 9, в связи с этим для участия в подобных обсуждениях очень важно выбрать владельца продукта с хорошей деловой хваткой.

Для управления техническим долгом имеются три основных вида деятельности (рис. 8.7). Каждый из них рассматривается по очереди в последующих разделах.

Контроль накапливания технического долга

Для правильного управления техническим долгом очень важно контролировать процесс его накапливания. Как обсуждалось ранее, взять на себя можно лишь такой технический долг, величина которого еще не достигла критической

массы. Непрерывное накапливание технического долга можно сравнить с постоянным одалживанием денег в кредит под дом. В какой-то момент нужно просто остановиться и сказать себе: “Довольно!”, потому что последствия окажутся очень серьезными.

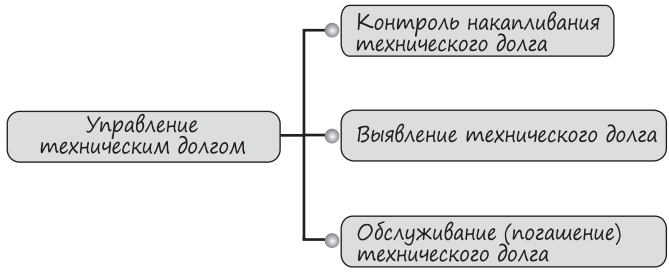


Рис. 8.7. Виды деятельности для управления техническим долгом

Прежде всего нужно перестать вводить наивный долг в разрабатываемые продукты, т.е. не проявлять больше опрометчивость и не создавать беспорядок. Необходимо также осознать, что стратегический или неизбежный долг может быть настолько большим, насколько мы можем накапливать его без погашения, прежде чем достигнем переломного момента. Далее будет показано, как добиться этого, но в то же время не обсуждается, как контролировать накапливание неизбежного долга, поскольку этот долг по своему характеру неизбежен, хотя и может быть выявлен и затем обслужен.

Применение норм надлежащей инженерной практики

Первый способ контроля над накапливанием технического долга состоит в том, чтобы перестать вводить наивный долг в разрабатываемые продукты. В качестве отличной отправной точки можно воспользоваться нормами надлежащей инженерной практики. И хотя *нормы инженерной практики* формально не определены в Scrum, всякая успешно работающая Scrum-команда применяет в своей деятельности такие нормы практики, как простая структура проекта, разработка посредством тестирования, непрерывная интеграция, автоматизированное тестирование, реорганизация кода и т.д. (подробнее об этом — в главе 20). Понимание и заблаговременное применение этих норм практики поможет командам остановить ввод в их продукты многих форм наивного долга.

Если накапливается технический долг, то для его погашения удобно воспользоваться таким важным инструментальным средством, как реорганизация кода. Это дисциплинирующая методика перестройки существующего тела кода, изменения его внутренней структуры, не затрагивающая внешнее поведение [Fowler et al. 1999]. Иными словами, мы исправляем внутренний механизм продукта,

но с точки зрения потребителя он работает, как и прежде. Реорганизуя код, мы стремимся упростить его и в то же время улучшить его сопровождаемость и расширяемость. В результате реорганизации кода выполнять текущую работу становится легче, что равнозначно сокращению процентных платежей.

Каннингхем поясняет преимущества реорганизации кода на следующем простом примере:

> *...заказчик желает заплатить за новое функциональное средство; если это средство не подходит, реорганизуйте код таким образом, чтобы оно подошло; после этого его нетрудно реализовать. Такую реорганизацию кода можно назвать оперативной. Ее суть я обычно поясняю руководству следующим образом: мы надеемся, что в нашем программном обеспечении найдется место для каждого нового запроса. Но иногда у нас нет места для нужного функционального средства, и тогда мы должны сначала расчистить место, а затем реализовать данное средство...*

Применение строгого критерия готовности

Работа, которую нужно было выполнить при построении функционального средства, но которая в конечном итоге была отложена на потом, служит одной из главных причин появления технического долга. Применяя методику Scrum, мы должны руководствоваться строгим критерием готовности (см. главу 4), чтобы помочь команде найти такое решение, которое обеспечит небольшой долг или полное его отсутствие в конце каждого спринта.

Чем более технически охватывающим будет составлен контрольный список по критерию готовности, тем меньше вероятность накапливания технического долга. И как пояснялось в главе 2, затраты на погашение технического долга, который проскальзывает мимо слабого критерия готовности, оказываются значительно большими, чем на его погашение в течение спринта. Работать без строгого критерия готовности — все равно, что выдавать лицензию на накапливание технического долга.

Правильное понимание экономических последствий технического долга

Чтобы пользоваться техническим долгом стратегически и с выгодой, нужно правильно понимать его влияние на экономические последствия принимаемых решений. К сожалению, в большинстве организаций недостаточно понимают, как правильно подсчитать экономические последствия взятия на себя технического долга. Это положение иллюстрирует пример, приведенный на рис. 8.8.

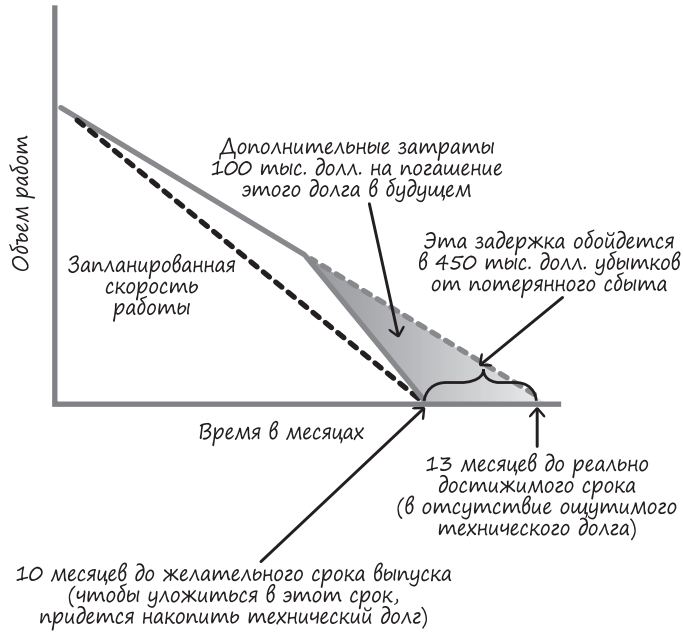


Рис. 8.8. Пример экономического анализа технического долга

В данном примере предполагается следующее.

- Ежемесячные затраты на разработку составляют 100 тысяч долларов.
- Уложиться в намеченный (через десять месяцев) срок выпуска со всеми требующимися, обязательными функциональными средствами нельзя.
- Пропустить выпуск каких-нибудь функциональных средств неприемлемо.

Рассмотрим два возможных варианта. Во-первых, срок выпуска продукта можно перенести на три месяца, чтобы благоразумно и профессионально завершить работу над обязательным рядом функциональных средств с минимальным техническим долгом через 13 месяцев. И тогда общие затраты на разработку составят 1,3 млн долларов. В ходе обсуждений со специалистами по сбыту и анализу рынка выяснилось, что задержка выпуска продукта на три месяца равноценна убыткам в 450 тыс. долларов от потерянного сбыта.

И во-вторых, можно ускорить разработку, пойдя по наименьшему пути сопротивления, чтобы уложиться в намеченный срок выпуска продукта через десять месяцев. Чтобы правильно подсчитать экономические последствия выбора такого варианта, нужно знать, во что обойдется взятие на себя технического долга. Именно здесь и возникают главные трудности. Представьте, что руководство задаст команде следующий вопрос: “Итак, если вам придется теперь пойти на некоторые проектные и реализационные компромиссы, чтобы получить обязательные

функциональные средства в готовом виде к первоначальному желательному сроку, то во что обойдется погашение долга после первого выпуска продукта?"

Допустим, что команда обсудит этот вопрос и придет к выводу, что ей потребуется четыре месяца на окончательную доводку системы. Это означает, что команде потребуется еще один дополнительный месяц, помимо тех трех месяцев, которые были "сэкономлены" благодаря выбору путей наименьшего сопротивления ради соблюдения намеченного срока выпуска продукта. Таким образом, команда понесет дополнительные затраты в 100 тыс. долларов на разработку, а в целом они составят 1,4 млн долларов вместо 1,3 млн, как в первом варианте. Следовательно, организации не придется дополнительно тратить 100 тыс. долларов, если уделить время правильному ведению проектных работ, не вводя, прежде всего, технический долг в разрабатываемый продукт.

На первый взгляд, верное экономическое решение очевидно. В самом деле, стоит ли брать на себя технический долг на 100 тыс. долларов, чтобы получить дополнительный доход в размере 450 тыс. долларов? Ясно, что никто от этого не откажется. И такой ответ может оказаться верным, если есть уверенность, что учтены все (или почти все) важные стоимостные факторы, связанные с техническим долгом.

Тем не менее ниже перечислены лишь два из многих факторов, которые не были учтены.

- Как насчет стоимости задержки погашения технического долга? Сумма 100 тыс. долларов покрывает затраты команды на сокращение технического долга в будущем. Но во что обойдется само время, затрачиваемое на сокращение долга? Время, потраченное на погашение основной части долга, означает задержку в работе над каким-нибудь другим продуктом или следующего выпуска того же самого продукта. Какова стоимость такой задержки? Так, если команде потребуется еще один месяц на погашение долга, то выпуск какого-нибудь другого продукта будет, скорее всего, задержан на тот же самый месяц. Такая стоимость упущенной прибыли имеет настолько реальные экономические последствия, что ее нельзя не учитывать.
- В большинстве организаций дело с погашением технического долга обстоит не очень хорошо. Когда дело начинает принимать серьезный оборот, руководство нередко предпочитает разрабатывать новые функциональные средства, а не переделывать уже существующие. Поэтому реальность такова, что в конечном итоге можно так и не погасить любые или все долги, а это означает, что, скорее всего, придется платить проценты по долгам за полезный срок службы системы. И этот фактор нужно также учитывать.

Цифры экономических последствий от взятия или исключения технического долга в рассматриваемом здесь примере приведены в табл. 8.1.

Таблица 8.1. Пример экономических последствий от взятия или исключения технического долга

	Исключение долга	**Взятие долга**
Ежемесячные затраты на разработку (в долларах США)	**100** тыс.	**100** тыс.
Общая продолжительность разработки (в месяцах)	**13**	**10**
Общие затраты на разработку (в долларах США)	**1,3** млн	**1,4** млн
Задержка выпуска продукта (в месяцах)	**3**	**0**
Стоимость задержки на месяц (в долларах США)	**150** тыс.	**150** тыс.
Общая стоимость задержки (в долларах США)	**450** тыс.	**0**
Обслуживание долга (в месяцах)	**0**	**4**
Затраты на обслуживание долга (в долларах США)	**0**	**400** тыс.
Общая сумма прибылей в течение срока службы продукта (в долларах США)	**1,75** млн	**1,4** млн
Стоимость задержки дополнительного времени на погашение долга (в долларах США)	**0**	**X**
Выплата процентов по техническому долгу в течение срока службы продукта (в долларах США)	**0**	**Y**
Прочие затраты, связанные с техническим долгом (в долларах США)	**0**	**Z**
Реальная сумма прибылей в течение срока службы продукта (в долларах США)	**1,75** млн	**1,4** млн + **X** + **Y Z**

Очевидно, что технический долг оказывает влияние на самые разные аспекты общих экономических расчетов. Если не принимать во внимание хотя бы самые важные факторы этого влияния, то вряд ли можно правильно подсчитать экономические последствия взятия технического долга.

Конечно, если экономические аргументы в пользу взятия технического долга перевешивают и весьма убедительны, — например, коммерческая деятельность прекратится, если не взять на себя долг и не выпустить продукт на рынок со всеми обязательными функциональными средствами, или ведущее место на рынке будет утрачено вместе с его львиной долей, — то вряд ли стоит тратить время на рассмотрение менее важных факторов, поскольку уже известно, что взять на себя долг экономически целесообразно.

Но чаще всего решение оказывается не таким ясным. Чтобы решить, брать или не брать на себя технический долг, обычно требуется провести подробный

анализ, позволяющий сделать правильный выбор. Принимая решение, следует склоняться к тому, чтобы не брать технический долг. Как показывает мой опыт, в большинстве организаций сильно недооценивают истинные затраты, которые влечет за собой взятие технического долга, и даже близко не пытаются их оценить, поскольку считают, что смогут погасить долг.

Выявление технического долга

Одно из главных преимуществ метафоры технического долга заключается в том, что она позволяет команде разработчиков и деловым людям обсуждать насущные вопросы в общем контексте. Для такого обсуждения и те и другие должны выявить в продукте технический долг таким образом, чтобы это стало ясно всем.

Выявление технического долга на деловом уровне

Выявление технического долга во многих организациях затрудняется тем, что если у команды разработчиков имеется хотя бы некоторое обоснованное представление о наличии у продукта технического долга, то у деловых людей такое представление, как правило, отсутствует. Если спросить любого инженерного работника, осведомленного о продукте, на каком его участке сосредоточен самый большой технический долг, он, скорее всего, сможет дать ответ. Если же спросить то же самое у делового человека, то зачастую оказывается, что у него нет ни малейшего представления о разновидности и величине существующего технического долга.

Ничего подобного не происходит с финансовым долгом. Если спросить делового человека о финансовом долге его организации, он сможет дать довольно точный ответ.

Следовательно, деловым людям очень важно показать наличие у продукта технического долга. Если бы технический долг можно было подсчитать количественно, а исследованию этого вопроса ныне уделяется немало внимания [SEI. 2011], то в правой части балансового отчета организации рядом с финансовым долгом появились бы строки с цифрами краткосрочного и долгосрочного технического долга (табл. 8.2).

Трудно найти такую организацию, в балансовых отчетах которой присутствовали бы строки с цифрами краткосрочного и долгосрочного технического долга, хотя это было бы весьма кстати. Данный пример служит лишь для иллюстрации того, что в каждой организации нужно найти способ сообщить величину технического долга у продукта таким образом, чтобы это было понятно деловым людям. В противном случае они не будут иметь ни малейшего представления об истинном состоянии продукта, чтобы принимать экономически обоснованные решения.

Таблица 8.2. Технический долг, показываемый в балансовом отчете

Актив (в долларах США)		Пассив (в долларах США)	
Наличные	600 тыс.	Текущая задолженность	
Дебиторская задолженность	450 тыс.	Векселя к оплате	100 тыс.
		Кредиторская задолженность	75 тыс.
		Краткосрочный технический долг	90 тыс.
Инструменты и оборудование	250 тыс.	Долгосрочная задолженность	
		Векселя к оплате	300 тыс.
		Долгосрочный технический долг	650 тыс.
...	...	...	...

Чтобы выявить коммерческие последствия появления технического долга, в некоторых организациях отслеживают скорость работы во времени. Как показано на рис. 8.6, увеличение технического долга приводит к снижению скорости работы. Такое снижение можно описать с точки зрения финансов. Допустим, что имеется Scrum-команда с фиксированными затратами 20 тыс. долларов на каждый спринт и исторически сложившейся скоростью работы в 20 очков за спринт. Используя эти цифры, можно рассчитать, что затраты команды на каждое очко составляют 1 тыс. долларов. Если накопление технического долга приводит к снижению скорости работы до 10 очков за спринт, то затраты на каждое очко возрастут до 2 тыс. долларов. Если же в общей сложности команда должна завершить работу, которая оценивается приблизительно в 200 очков, а скорость ее работы снижается наполовину, то первоначальные затраты в 200 тыс. долларов на эту работу возрастут до 400 тыс. долларов. Таким образом, используя показатель скорости работы, можно явно выявить финансовые затраты на выплату процентов по накопившемуся техническому долгу.

Выявление технического долга на инженерном уровне

Инженерные работы нередко имеют *неявное знание* об участке продукта, где находится самый вопиющий технический долг. Но поскольку это знание неявное, то оно не позволяет анализировать, обсуждать или воздействовать на технический долг. На рис. 8.9 показаны три способа выявления технического долга на инженерном уровне.

Во-первых, технический долг можно запротоколировать подобно дефектам в существующей системе обнаружения неисправностей (см. рис. 8.9, *слева*). Преимущество такого способа заключается в том, что долг можно разместить в привычном месте, используя известные инструментальные средства и методики. Если сведения о долге хранятся в том же месте, где и сведения о дефектах, то

долг следует непременно пометить таким образом, чтобы легко его обнаружить, поскольку команда может решить обслуживать долг иначе, чем дефекты, как поясняется далее.

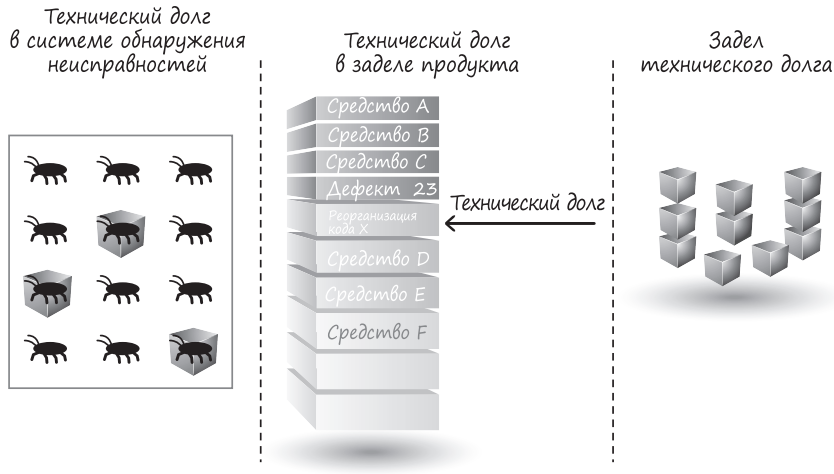


Рис. 8.9. Способы выявления технического долга на инженерном уровне

Во-вторых, можно создать элементы задела продукта, представляющие технический долг (см. рис. 8.9, *посредине*). Благодаря этому важный технический долг будет выявлен наравне с новыми функциональными средствами в заделе продукта. Как правило, команды применяют именно этот способ, когда затраты на обслуживание технического долга достаточно велики, а владельцу продукта приходится решать, как упорядочить работу относительно новых дополнительных средств в заделе продукта.

И в-третьих, для технического долга можно создать специальный задел, чтобы выявить в нем отдельные элементы технического долга (см. рис. 8.9, *справа*). Всякий раз, когда новый технический долг обнаруживается или внедряется в продукт, член команды разработчиков может создать новый элемент и ввести его в задел этого технического долга. Выявляя элементы технического долга, команда разработчиков может не только увидеть положение своего технического долга, но и заблаговременно определить момент, когда потребуется обслуживание каждой части этого технического долга.

Для совмещенных команд может подойти простой способ наглядного представления задела технического долга на специальной доске, вывешиваемой на стене, где отдельные элементы задела представлены в виде разноцветных наклеек для заметок или карточек. Как правило, доска технического долга располагается рядом с заделом спринта, чтобы при планировании спринта команда видела технический долг, который она собирается обслужить в предстоящем спринте (более подробно этот способ рассматривается в следующем разделе).

Большинство команд усматривают в заделе технического долга нецеремонный способ размещения на стене карточек технического долга. Но в некоторых командах может быть принято решение упорядочивать задел технического долга по карточкам или дать хотя бы приближенное представление о том объеме работ, который потребуется для погашения долга, описываемого на карточке.

Обслуживание технического долга

И последним видом деятельности по управлению техническим долгом является обслуживание или погашение накопившегося долга. Для обсуждения вопросов обслуживания технического долга полезно разделить его состояние на следующие категории.

- **Случайно обнаруженный технический долг** — это такой долг, о существовании которого команда и не подозревала до тех пор, пока он не проявился в нормальном ходе работы над продуктом. Такой долг возникает, например, в том случае, когда команда вводит новое функциональное средство в продукт и обнаруживает обходной прием, внедренный некогда в код каким-то уже давно уволившимся разработчиком.
- **Известный технический долг** — это такой долг, который известен команде разработчиков и выявлен одним из рассмотренных выше способов.
- **Намеченный технический долг** — это такой долг, который известен команде разработчиков и намечен ею для обслуживания.

Основываясь на этих категориях, к обслуживанию технического долга можно применить следующий алгоритм.

1. Выяснить, следует ли обслуживать известный технический долг (как поясняется далее, не весь долг должен быть обслужен). Если долг следует обслужить, перейти к п.2.
2. Если в работе над кодом выявится случайно обнаруженный технический долг, его нужно погасить. Если же величина случайно обнаруженного технического долга превышает некоторый благоразумный порог, этот долг следует гасить до тех пор, пока не будет достигнут упомянутый порог. Затем необслуживаемый, случайно обнаруженный технический долг следует отнести к категории известного технического долга, создав, например, элементы в его заделе.
3. Рассматривать в каждом спринте какую-то часть известного технического долга как намеченный технический долг, который должен быть обслужен в течение спринта. Предпочтение следует отдавать известному техническому долгу с высокой процентной ставкой, которая согласуется с ценной для заказчика работой.

Методы, приведенные на рис. 8.10, расширяют данный алгоритм для обслуживания технического долга. Каждый из этих методов и их применение по методике Scrum описывается в последующих разделах.

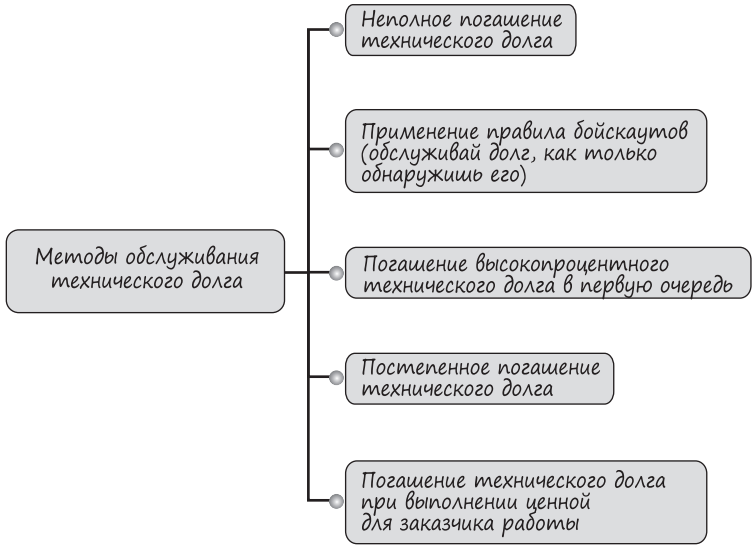


Рис. 8.10. Методы обслуживания технического долга

Неполное погашение технического долга

Иногда технический долг не следует гасить. Это одна из тех областей, на которую простирается аналогия с финансовым долгом. Как правило, ожидается, что весь финансовый долг в конечном итоге будет погашен, хотя известно, что это не всегда возможно!

Имеется целый ряд случаев, когда технический долг не должен быть погашен. Ниже рассматриваются три из них: приближение конца срока службы продукта, разработка одноразового прототипа и создание продукта, рассчитанного на краткий срок службы.

Приближение конца срока службы продукта

Если у продукта накопился значительный технический долг и приближается конец срока его службы, то вкладывать средства в сколько-нибудь значительное погашение долга финансово безответственно. Так, если продукт имеет малую ценность, то его лучше изъять из эксплуатации, а вместе с ним и ликвидировать долг, направив имеющиеся ресурсы на более ценные продукты. А если продукт имеет большую ценность и немалый технический долг, то лучше взять на себя большой риск и немалые затраты на разработку нового продукта, чем гасить технический долг у прежнего продукта.

Разработка одноразового прототипа

Иногда взять на себя намеренно технический долг, совершенно не собираясь его возмещать, оказывается наиболее экономически целесообразным решением. Характерным тому примером служит разработка одноразового прототипа, создаваемого только для целей приобретения знаний [Goldberg, Adele, and Kenneth S. Rubin. 1995]. Ценность прототипа состоит не в самом коде, а в получаемом утвержденном обучении [Ries, Eric. 2011]. Прототип обычно разрабатывается не для существования на рынке, и поэтому у него, скорее всего, имеется некоторый или даже большой технический долг. Но поскольку это одноразовый прототип, то и нет никаких оснований возмещать его долг. Безусловно, если создать одноразовый прототип и затем решить, что он еще пригодится как развивающийся далее в конкретный продукт, то можно почти с полной уверенностью сказать, что он основательно погряз в значительном техническом долге.

Построение продукта, рассчитанного на краткий срок службы

Если продукт строится с расчетом на краткий срок службы, то возмещать его технический долг экономически нецелесообразно. Этот случай я поясню на одном интересном примере из моей практики в конце 1980-х годов. В то время я работал в компании ParcPlace Systems, ставшей одним из первых лидеров на рынке объектно-ориентированных сред разработки. Тогда я помогал нескольким ведущим американским банкам внедрять Smalltalk в качестве платформы для разработки программного обеспечения. Однажды меня попросили позаниматься с командой разработчиков, чтобы помочь ее членам лучше усвоить объектно-ориентированную технологию и эффективнее пользоваться Smalltalk в качестве платформы для разработки. Эта команда только что выпустила одну из первых в своем роде систем для торговли производными финансовыми инструментами. Когда я прибыл на место, то сразу же попросил у вице-президента финансовой команды разрешение провести анализ проектирования и реализации продукта, только что построенного командой. Этот продукт еще не начал функционировать, но это планировалось вскоре сделать.

Потратив день на анализ архитектуры и кода, я встретился с вице-президентом и сказал ему, что их система может оказаться самой скверной реализацией на платформе Smalltalk из тех, что мне приходилось когда-либо видеть. Я указал на то, что реализация имеет слишком много недостатков, чтобы устранить их немедленно, а иначе их система (и деятельность) принесла бы немало разочарования.

И тогда вице-президент сказал мне буквально следующее: “Сынок, если ты потратишь хотя бы 5 центов на доводку этой системы, я лично выведу тебя вон и прибью”. Эти слова, мягко говоря, меня ошарашили. И я ответил: “Вы должны мне довериться. Эта система неудачно спроектирована и ужасно реализована, и поэтому у вас еще долго будут с ней проблемы”. Он резко возразил: “Ты не понимаешь мой

бизнес. Если я выхожу на свой рынок с новым финансовым инструментом, то получаю львиную долю прибыли в течение первых трех месяцев. Именно столько времени потребуется нашим конкурентам, чтобы поспеть с аналогичными продуктами. И в данный момент мне лучше выйти на рынок и приступить к разработке нового продукта. Мне нужно лишь, чтобы новая система просуществовала три месяца. И меня не волнует, будет ли она слеплена жевательной резинкой и упаковочной проволокой. Нужно только не задерживать получение дохода и дать моим конкурентам возможность победить меня на рынке. И поэтому мы ее запускаем".

Именно так они и поступили. За первый же час работы системы пользовавшиеся ею торговцы получили прибыль в размере 14 млн долларов. Я думал, что они сильно рисковали, запустив систему в столь хрупком состоянии, но с точки зрения полученной прибыли я оказался не прав.

Как правило, организации не создают продукты с расчетом на три месяца эксплуатации. И зачастую их больше интересует разработка продукта для длительного существования на рынке.

Применение правила бойскаутов

У бойскаутов имеется правило, которое гласит: "Всегда оставляй стоянку под лагерь чище, чем она была до твоего прихода". Если бойскауты обнаружат беспорядок на стоянке, они приберут ее, кто бы этот беспорядок ни оставил. Таким образом, они намеренно готовят стоянку для следующей команды туристов. Боб Мартин (Bob Martin) подобно поясняет в своей книге, почему данное правило так подходит для разработки продукции и обслуживания технического долга [Martin, Robert C. 2008].

Следуя этому правилу, мы стараемся сделать проектирование и реализацию продукта чуть лучше, а не хуже всякий раз, когда касаемся его. Так, если член команды, работающий на определенном участке продукта, обнаружит в нем недостаток (т.е. выявит случайно обнаруженный технический долг), он устранит его. И сделает он это потому, что так будет лучше не только для него лично, но и для всей команды и организации в целом.

Как следует из представленного ранее алгоритма, случайно обнаруженный технический долг обслуживается до некоторого благоразумного порога. Было бы опрометчиво сказать, что команда должна обслужить весь случайно обнаруженный технический долг, как только его выявит. Ведь обслуживание такого долга может потребовать немалых усилий, а команда находится посредине спринта, в котором ей нужно еще завершить другую работу. Если же команда попытается обслужить весь долг, она может и не достичь намеченной цели спринта.

В качестве выхода из данного положения команда может выделить некоторую долю времени для обслуживания случайно обнаруженного долга. С одной

стороны, для этого можно увеличить оцениваемые размеры отдельных элементов задела продукта, чтобы выделить дополнительное время на регулярное обслуживание долга. А с другой стороны, долю времени для обслуживания случайно обнаруженного долга можно выделить при планировании спринта. Как показывает мой прошлый опыт, для этой цели обычно выделяется от 5 до 33% времени, отводимого на спринт. Если выбрать именно такой метод, то доля времени, выделяемого на обслуживание долга, будет зависеть от конкретных обстоятельств. Что же касается любого случайно обнаруженного долга, который не обслуживается, как только он обнаруживается, то его следует отнести к категории известного долга и выявить, применяя любой метод, выбранный командой для наглядного представления технического долга.

Постепенное погашение технического долга

У некоторых продуктов может накапливаться довольно большой технический долг. Команды, работающие над такими продуктами, нередко делают крупные единовременные платежи в погашение обслуживаемого долга. Но им лучше было бы сделать много своевременных постепенных платежей в погашение известного технического долга вместо крупного платежа в последний момент. Более мелкие, но частые платежи сродни ежемесячным выплатам ипотеки. Такой метод обслуживания долга каждый месяц позволяет избежать крупного единовременного платежа в конце срока займа.

Меня сразу настораживает, когда я слышу, как команды обсуждают свои "спринты технического долга" или "спринты реорганизации кода". Единственная цель таких спринтов — выполнить работу по сокращению технического долга. Мне это кажется похожим на крупный единовременный платеж. На самом деле такие спринты создают впечатление, что величина долга позволяла не уделять должного внимания его сокращению. А когда долг стал мешать, то, вместо того, чтобы разрабатывать ценные для заказчика функциональные средства в следующем спринте, команда собирается полностью посвятить его обслуживанию долга, который она могла бы постепенно возмещать в каждом предыдущем спринте. Если накапливается большой технический долг, которому не уделяется должное внимание, то иногда оказывается даже полезно посвятить отдельный спринт погашению этого долга, сосредоточив на нем согласованные усилия всей команды. Но, как правило, таких спринтов следует избегать при всякой возможности. Возмещение технического долга должно происходить постепенно.

Применяя такой подход, мы берем на себя некоторую часть известного технического долга и обозначаем ее как намеченный технический долг, который должен быть обслужен в течение следующего спринта. Решение относительно доли технического долга, намечаемой для погашения в каждом спринте, может быть принято Scrum-командой во время планирования спринта.

Погашение высокопроцентного технического долга в первую очередь

Все виды упрощений и упущений было бы, конечно, удобно свести под одной маркой технического долга, но очень важно понимать, что не все формы технического долга имеют одинаковое значение. Характерным примером важной формы долга служит часто модифицируемый модуль, от которого зависит немало другого кода и который действительно требует реорганизации, поскольку вносить в него изменения с каждым разом становится все труднее. Мы все время платим проценты по этому долгу, а величина этих процентов продолжает нарастать по мере того, как мы вносим все больше и больше изменений.

С другой стороны, мы могли бы допустить технический долг (т.е. известные вопросы проектирования и реализации) как часть продукта, который редко используется и почти никогда не модифицируется. Проценты по такому долгу не выплачиваются регулярно или, по крайней мере, они невелики. И это не та форма долга, которая требует много внимания, если только отсутствует значительный риск, что данная часть продукта может отказать и что такой отказ будет иметь серьезные последствия.

Следовательно, обслуживая технический долг, мы должны в первую очередь наметить и обслужить высокопроцентный технический долг. Любой благоразумный деловой человек поступил бы точно так же с финансовым долгом. Например, если нет на то веских оснований, то, как правило, финансовый долг с процентной ставкой 18% гасится прежде, чем долг с процентной ставкой 6%.

В некоторых организациях технический долг накапливается настолько, что это отчасти парализует их, поскольку они не знают, с чего начать. Для них высокопроцентный долг может быть очевидным, но его величина — пугающей. Чтобы как-то стимулировать сокращение долга, они могут решиться на погашение небольшой его части, чтобы постепенно привыкнуть к процессу возмещения долга. Я лично сторонник любых культурных мер, которые требуются, чтобы побудить организации приступить к управлению своим долгом. И как поясняется далее, если мы возмещаем технический долг, выполняя ценную для заказчика работу, то можем постепенно сосредоточиться на той небольшой части этого долга, которую стоит погасить.

Погашение технического долга при выполнении ценной для заказчика работы

Отличный способ постепенно возмещать известный технический долг, уделяя основное внимание высокопроцентному техническому долгу и согласуя обслуживание технического долга с методикой Scrum, ориентированной на доставку

ценности, состоит в том, чтобы выплачивать долг по ходу выполнения ценной для заказчика работы. Это означает, что при всякой возможности следует избегать планирования целого спринта для работ по сокращению долга или определения элементов задела продукта специально для сокращения долга. Вместо этого известный технический долг следует обслуживать одновременно с разработкой ценных для заказчика функциональных средств из задела продукта.

Допустим, что мы делаем еще кое-что для каждого элемента задела продукта, который представляет ценность для заказчика и над которым мы работаем. Во-первых, мы обязуемся выполнять работу качественно, чтобы не вводить новый наивный технический долг, когда создаем функциональное средство для заказчика. Во-вторых, мы применяем правило бойскаутов, ликвидируя любой случайно обнаруженный технический долг, который можно вполне погасить, выполняя работу над функциональным средством. И в-третьих, что самое главное в данном методе, возмещаем намеченный технический долг на том конкретном участке, где мы собираемся работать.

Применение такого метода дает ряд следующих преимуществ.

- Позволяет согласовать работу по сокращению долга с ценной для заказчика работой, для которой владелец продукта назначил соответствующий приоритет.
- Дает всем членам команды разработчиков ясно понять, что ответственность за сокращение технического долга лежит на всем коллективе, и ее нельзя переложить на кого-то другого или вообще на другую команду.
- Упрочивает навыки предотвращения и устранения технического долга, поскольку все члены команды разработчиков постоянно практикуются в них.
- Помогает выявлять высокопроцентные части технического долга, на которых следует сосредоточить усилия по обслуживанию долга. По крайней мере, нам известно, что код (или другой артефакт разработки), которого мы касаемся, по-прежнему важен, поскольку пользуемся им для создания нового функционального средства.
- Позволяет избежать убытков от возмещения технического долга на тех участках, где этого не следует делать.

Ранее упоминался способ, применение которого мне приходилось наблюдать в некоторых Scrum-командах. С его помощью они старались согласовать работы по сокращению известного технического долга с реализацией элементов из задела продукта, как показано на рис. 8.11. Этим способом элементы известного технического долга вводятся в его задел, размещаемый в виде доски на стене рядом с заделом спринта (или в инструментальном средстве, специально предназначенном для той же самой цели) во время планирования спринта.

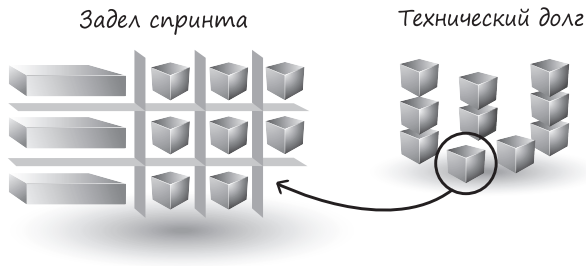


Рис. 8.11. Способ управления техническим долгом по методике Scrum

Во время планирования спринта члены команды выбирают вместе с владельцем продукта ценные для заказчика элементы из задела продукта, чтобы работать над ними в следующем спринте. С этой целью они обращаются к карточкам на доске технического долга, чтобы видеть, пересекается ли работа, которую они планируют выполнить над новым элементом задела продукта, естественным образом с участком продукта, связанным с карточкой технического долга. Если такое пересечение обнаруживается, то кто-нибудь из присутствующих снимает карточку с доски технического долга и переносит ее в задел спринта как работу, намечаемую для данного спринта. А когда выполняется работа, необходимая для завершения элемента задела продукта, члены команды решают также задачи возмещения технического долга, перенесенные в спринт. Это очень простой и изящный способ согласования обслуживания технического долга с созданием пользовательской ценности.

Заключение

В этой главе обсуждалось понятие технического долга, который накапливается, когда упрощения делаются сегодня за счет завтра. Сначала в ней были рассмотрены следующие формы технического долга: наивный, неизбежный и стратегический. Затем в этой главе пояснялись последствия неудачного управления техническим долгом на разных уровнях его накапливания. Далее обсуждались три вида деятельности по управлению техническим долгом: контроль накапливания, выявление и обслуживание технического долга.

Этой главой завершается часть I данной книги. А следующая глава открывает часть II, посвященную обсуждению различных ролей в проектных работах по методике Scrum, начиная с роли владельца продукта.

Часть II

Роли

Глава 9

Владелец продукта

В этой главе подробно описывается роль владельца продукта. Сначала в ней поясняется назначение этой роли по отношению к остальным ролям в Scrum. Затем подробно описываются основные обязанности и характеристики владельца продукта. Далее в главе представлен типичный график работы владельца продукта, раскрывающий характер его деятельности в течение многих недель. После этого поясняется, кто должен быть владельцем продукта для разных видов разработки продукции. И в завершение описывается сочетание роли владельца продукта с другими ролями и ее расширение до масштабов команды владельцев продукта.

Краткий обзор

Владелец продукта наделен полномочиями лидера разработки продукта. Ему принадлежит одна из трех партнерских ролей, составляющих каждую Scrum-команду, тогда как остальные роли принадлежат Scrum-мастеру и команде разработчиков. Владелец продукта должен одновременно смотреть, по крайней мере, в двух направлениях (рис. 9.1).

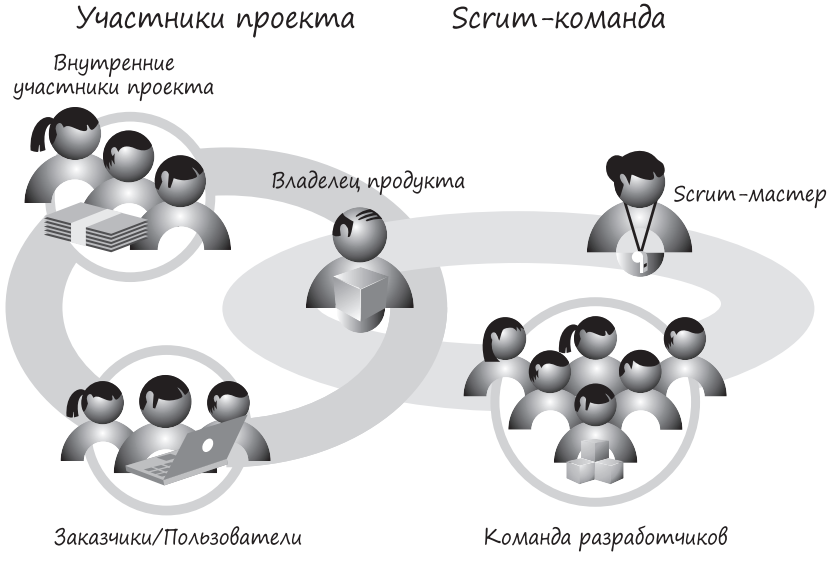


Рис. 9.1. Владельцу продукта приходится одновременно смотреть в двух направлениях

С одной стороны, владелец продукта должен ясно понимать потребности и приоритеты участников проекта из его организации, заказчиков и пользователей, чтобы служить их голосом. В этом отношении владелец продукта выполняет роль ответственного за выпуск продукции, обеспечивая выработку правильных решений.

С другой стороны, владелец продукта должен сообщать команде разработчиков, что и в каком порядке она должна разрабатывать. Кроме того, владелец продукта должен обеспечить определение критериев приемки разрабатываемых функциональных средств и выполнение тестов для проверки этих критериев, чтобы выяснить, завершены ли функциональные средства. Владелец продукта не пишет подробные тесты, но обеспечивает их написание на высоком уровне, чтобы команда могла определить, когда владелец продукта посчитает функциональное средство завершенным. В этом отношении владелец продукта отчасти выполняет роли специалиста по анализу экономической деятельности и тестировщика компонентов.

Основные обязанности

На рис. 9.2 показаны основные обязанности владельца продукта.

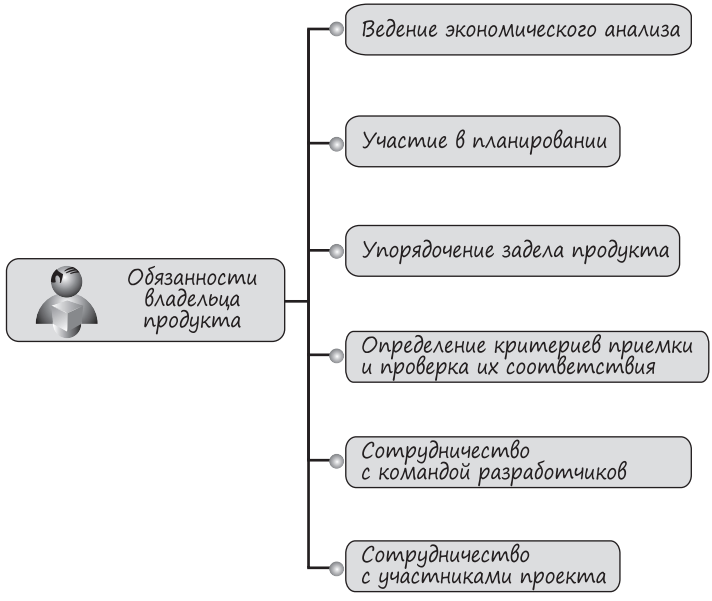


Рис. 9.2. Основные обязанности владельца продукта

Ведение экономического анализа

Владелец продукта отвечает за постоянное принятие обоснованных экономических решений на уровнях выпуска, спринта и задела продукта (рис. 9.3).

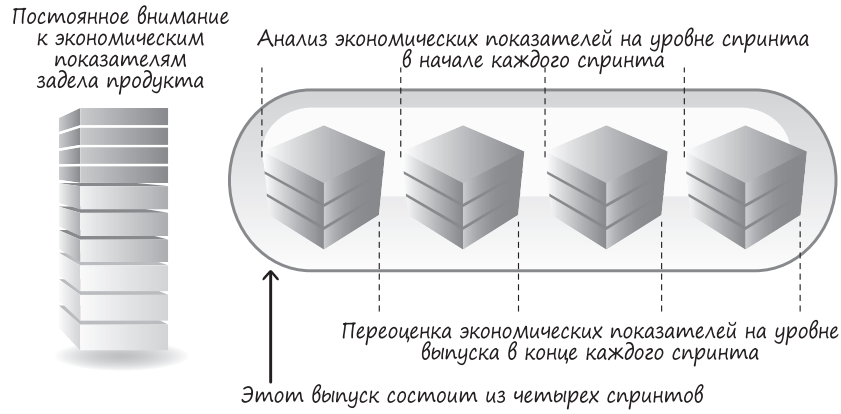


Рис. 9.3. Владелец продукта ведет экономический анализ

Экономический анализ на уровне выпуска

На уровне выпуска владелец постоянно принимает компромиссные решения в отношении объема работ, сроков, бюджета и качества по мере поступления экономически важной информации во время разработки продукции. Компромиссные решения, принимаемые в начале выпуска, могут оказаться неприемлемыми при поступлении новой информации во время выпуска.

Например, что если через несколько недель после начала шестимесячных проектных работ станет ясно, что прибыль можно увеличить на 50%, если выделить еще одну неделю (с 4%-ным отставанием от графика) для внедрения вновь выявленного функционального средства в выпуск? Следует ли расходовать одну неделю рабочего времени и нести дополнительные затраты в обмен на бо́льшую прибыль? За принятием такого решения следит владелец продукта. В одних случаях он может принять решение в одностороннем порядке, а в других — рекомендовать его, но сотрудничать с остальными участниками проекта над принятием решения, чтобы обеспечить их вклад, а иногда и утверждение, в исполнение решения.

Кроме того, в конце каждого спринта владелец продукта следит за принятием решения, следует ли финансировать следующий спринт. Если достигнут заметный прогресс в продвижении к цели выпуска или следующий спринт экономически оправдан, то он финансируется. А если прогресс незаметный или экономические показатели не способствуют дополнительным затратам, то дальнейшая работа может быть отменена.

Удовлетворенный владелец продукта может также проследить в конце спринта за принятием решения прекратить финансирование дальнейшей разработки, если продукт уже готов к поставке и дополнительные расходы на него просто не оправданы. Допустим, что планируется выпуск в течение десяти спринтов.

После седьмого спринта владелец продукта анализирует элементы, оставшиеся в заделе продукта, и приходит к выводу, что затраты на их реализацию окажутся выше, чем ценность, которую они доставят. И тогда владелец продукта может решить, что ранняя поставка продукта будет более целесообразной, чем продолжение его разработки по первоначальному плану в течение десяти спринтов. Подобная гибкость раннего выпуска допустима, если наиболее ценные элементы, расположенные у вершины задела продукта, реализованы в первую очередь, а команда завершает работу в каждом спринте в соответствии со строгим критерием готовности.

И, разумеется, владелец продукта может прийти в конце спринта к выводу, что финансирование следует прекратить, поскольку базовые экономические показатели изменились. Например, что если создается продукт для конкретной страны, а регуляторный орган этой страны пересматривает свои законы, из-за чего продукт становится неприбыльным, а возможно, даже запрещенным к продаже в данной стране? В подобных случаях владелец продукта может проследить за отменой проектных работ, даже если дела в остальном идут неплохо.

Экономический анализ на уровне спринта

Владелец продукта ведет экономический анализ не только на уровне выпуска, но и на уровне спринта, обеспечивая хорошую окупаемость капиталовложений в каждом спринте. Грамотный владелец продукта умеет распоряжаться средствами своей организации как своими. Как правило, владелец продукта знает затраты на следующий спринт, поскольку ему известны продолжительность спринта и состав команды. Зная это, владелец продукта должен при планировании спринта задать себе следующий вопрос: “Сниму ли я деньги со счета в моем банке, равные затратам на этот спринт, чтобы получить функциональные средства, которые мы планируем в нем создать?” Если ответ на этот вопрос окажется отрицательным, грамотный владелец продукта не потратит впустую средства своей организации.

Экономический анализ на уровне задела продукта

Как обсуждалось в главе 6, владелец продукта отвечает за расстановку элементов в заделе продукта по приоритетам. Когда изменяются экономические условия, изменятся, скорее всего, и приоритеты в заделе продукта.

Допустим, что в начале выпуска владелец продукта считает отдельное функциональное средство ценным для большой доли целевых пользователей, а команда считает, что для его создания потребуются самые скромные усилия. Но спустя несколько спринтов команда обнаруживает, что для завершения данного средства потребуются немалые усилия и что оно представляет ценность лишь небольшой части целевых пользователей. Вследствие значительного снижения

отношения затрат и прибыли для этого функционального средства владелец продукта должен переставить элементы в заделе продукта по приоритетам, чтобы отразить это новое обстоятельство, вероятнее всего, переместив вниз элементы, связанные с данным средством.

Участие в планировании

Владелец продукта является главным участником мероприятий по планированию портфеля заказов, продукта, выпуска и спринта. Во время планирования портфеля заказов (см. главу 16) владелец продукта сотрудничает с внутренними участниками проекта (возможно, с комитетом по утверждениям или с коллегиальным органом управления), чтобы найти продукту подходящее место в заделе портфеля заказов и определить даты начала и завершения разработки продукта. Во время планирования продукта (см. главу 17) владелец продукта сотрудничает с участниками проекта, чтобы выработать общий замысел продукта. Во время планирования выпуска (см. главу 18) владелец продукта сотрудничает с участниками проекта и командой разработчиков, чтобы определить содержимое следующего выпуска. А во время планирования спринта (см. главу 19) владелец продукта сотрудничает с командой разработчиков, чтобы поставить цель спринта. Кроме того, он предоставляет ценные входные данные, позволяющие команде разработчиков выбрать из задела продукта ряд элементов, которые команда может действительно выпустить к концу спринта.

Упорядочение задела продукта

Владелец продукта следит за упорядочением задела продукта, которое включает в себя создание и уточнение, оценивание и расстановку по приоритетам элементов в заделе продукта (см. главу 6). Сам владелец продукта выполняет не всю работу по упорядочению задела продукта. Например, он может не составлять элементы задела продукта, привлекая к этой работе других участников проекта. Кроме того, владелец продукта не оценивает элементы (это обязанность команды разработчиков), но готов ответить на уточняющие вопросы и дать необходимые разъяснения во время оценивания. Тем не менее владелец продукта несет окончательную ответственность за то, чтобы деятельность по упорядочению задела продукта способствовала плавному потоку доставки ценности.

Определение критериев приемки и проверка их соответствия

Владелец продукта отвечает за определение критериев приемки для каждого элемента задела продукта. Эти критерии представляют собой условия, при

которых владелец продукта может быть удовлетворен выполнением функциональных и нефункциональных требований. Владелец продукта может также писать приемочные тесты, отвечающие критериям приемки, или обратиться за помощью к экспертам в предметной области или членам команды разработчиков. Так или иначе, владелец продукта должен обеспечить составление подобных критериев приемки (а зачастую — конкретных приемочных тестов) перед тем, как элемент будет рассматриваться на совещании по планированию спринта. Без этих критериев команда не сможет до конца уяснить назначение элемента и не будет готова включить его в спринт. Именно поэтому многие Scrum-команды включают ясно определенные критерии приемки в качестве отдельных элементов в свои контрольные списки по критерию подготовленности (см. главу 6).

Владелец продукта несет окончательную ответственность за соответствие поверяемого элемента критериям приемки. И с этой целью он может сам выполнить приемочные тесты или обратиться за помощью к опытным пользователям, чтобы убедиться в том, что элемент задела продукта отвечает условиям удовлетворения. Команда может помочь ему создать среду тестирования, чтобы он сам или привлеченный им эксперт в предметной области мог эффективно выполнить эти тесты. Но окончательное суждение о том, оправдывает ли элемент возлагаемые на него надежды, выносит сам владелец продукта.

Владельцу продукта очень важно проверить соответствие элементов критериям приемки в течение спринта, а не ждать подведения его итогов. Тестируя функциональные средства по мере их готовности, владелец продукта может выявить ошибки и недоразумения, которые команда способна устранить еще до подведения итогов спринта. Кроме того, команде разрешается демонстрировать только завершенные функциональные средства при подведении итогов спринта, и поэтому владелец продукта должен еще до подведения итогов спринта убедиться, что приемочные тесты проходят, чтобы команда знала, какие именно функциональные средства действительно удовлетворяют критерию готовности.

Сотрудничество с командой разработчиков

Владелец продукта должен тесно и часто сотрудничать с командой разработчиков. Его роль — быть постоянно вовлеченным в процесс разработки. Во многих организациях, только начинающих внедрять методику Scrum, не стремятся поощрять надлежащее сотрудничество владельца продукта с командой разработчиков, что замедляет ответную реакцию и значительно сокращает ценность этой реакции, когда она происходит.

Подобное отсутствие тесного сотрудничества может объясняться также тем, что владелец продукта еще не освоился со своей ролью и считает, что его участие в процессе разработки по методике Scrum должно быть похожим

на участие в процессе разработки по традиционной последовательной методике. На рис. 9.4 сравнивается типичный уровень участия заказчиков и руководства в процессе разработки по традиционной последовательной методике с уровнем участия владельца продукта в процессе разработки по методике Scrum.

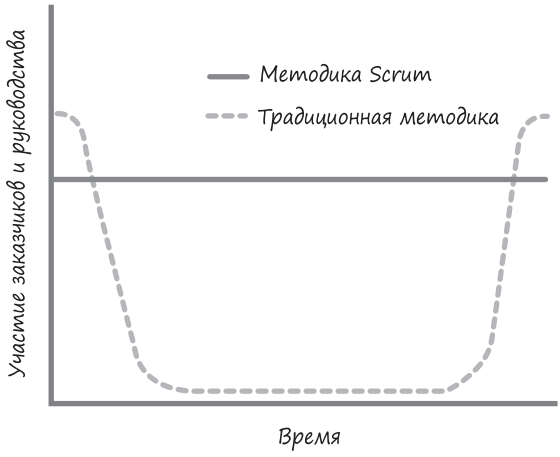


Рис. 9.4. Сравнение уровней участия заказчиков и руководства во времени

При разработке по традиционной последовательной методике кривая участия приобретает U-образную форму ванной. Первоначально заказчики принимают активное участие в процессе разработки, помогая определить полный ряд требований. Но как только проектные работы переходят в более техническую стадию (проектирования, программирования и некоторых видов тестирования), заказчики больше не нужны. А раз так, то уровень их участия становится довольно низким или вообще нулевым в течение большей части проектных работ. В действительности при разработке по традиционной последовательной методике заказчики не вмешиваются в этот процесс практически до самого его завершения, когда им как пользователям требуется выполнить приемочные испытания разработанного продукта. На этой стадии заказчики обычно обнаруживают, что этот продукт оказывается совсем не тем, что им требовалось. Хуже того, это происходит слишком поздно, а внесение изменений — по крайней мере, в данный выпуск — может обойтись им очень дорого. Приходя в ожидании получить удовольствие от продукта, заказчики уходят пораженными, обескураженными и разочарованными. И тогда начинается поиск виновных. В частности, заказчик заявляет: “Если бы вы внимательнее читали мое техническое задание, то разработали бы именно то, что мне действительно нужно”, на что команда разработчиков отвечает: “А если бы вы составили технические требования яснее, мы бы разработали нечто совсем другое. Мы разработали именно то, что вы просили!”

Применяя методику Scrum, мы разрабатываем функциональные средства по очереди, а не по стадиям. Это означает, что мы выполняем все виды работ для создания конкретного функционального средства (проектирование, программирование, интеграция, тестирование) в течение спринта. В этой связи очень важен постоянный уровень участия владельца продукта в процессе разработки. При таком тесном сотрудничестве в течение краткосрочных рабочих циклов или итераций вероятность потери связи владельца продукта с командой разработчиков значительно уменьшается. Это дает еще одно дополнительное преимущество: если разработка по методике Scrum выполняется правильно, то поиск виновных исключается!

Сотрудничество с участниками проекта

Владелец продукта служит в качестве единого голоса для всех участников проекта: как внутренних, так и внешних. К числу *внутренних участников проекта* относятся владельцы коммерческих систем, руководящий состав организаций, руководители программ, специалисты по анализу рынка и сбыту, а к числу *внешних участников проекта* — заказчики, пользователи, партнеры, регуляторные органы и прочие заинтересованные лица. Владелец продукта должен тесно сотрудничать со всеми участниками проекта, собирая входные данные и формируя согласованное представление, чтобы руководить разработкой продукции.

Если владелец продукта пребывает в растерянности и слишком разбрасывается, ему будет трудно сотрудничать на должном уровне как с командой разработчиков, так и с участниками проекта. Иногда рабочая нагрузка может оказаться ему не под силу, и тогда он вправе обратиться за посторонней помощью, чтобы справиться с обязанностями, возложенными на его роль. Мы еще вернемся к этому вопросу, когда будем рассматривать понятие команды владельцев продукта.

Характеристики и навыки

На рис. 9.5 приведены наиболее важные характеристики роли владельца продукта. Несмотря на многочисленность характеристик грамотного владельца продукта, их можно разделить на следующие четыре категории: навыки в предметной области, навыки работы с людьми, принятие решений и ответственность.

Навыки в предметной области

Владелец продукта должен быть дальновидным, чтобы формировать правильное представление о продукте и вести команду к достижению этого представления. Но наличие самого представления еще не означает, что ясно видны подробности и пути его достижения. Грамотный владелец продукта знает, что не все можно предвидеть заранее, и поэтому готов приспосабливаться к изменениям, если они оправданы.

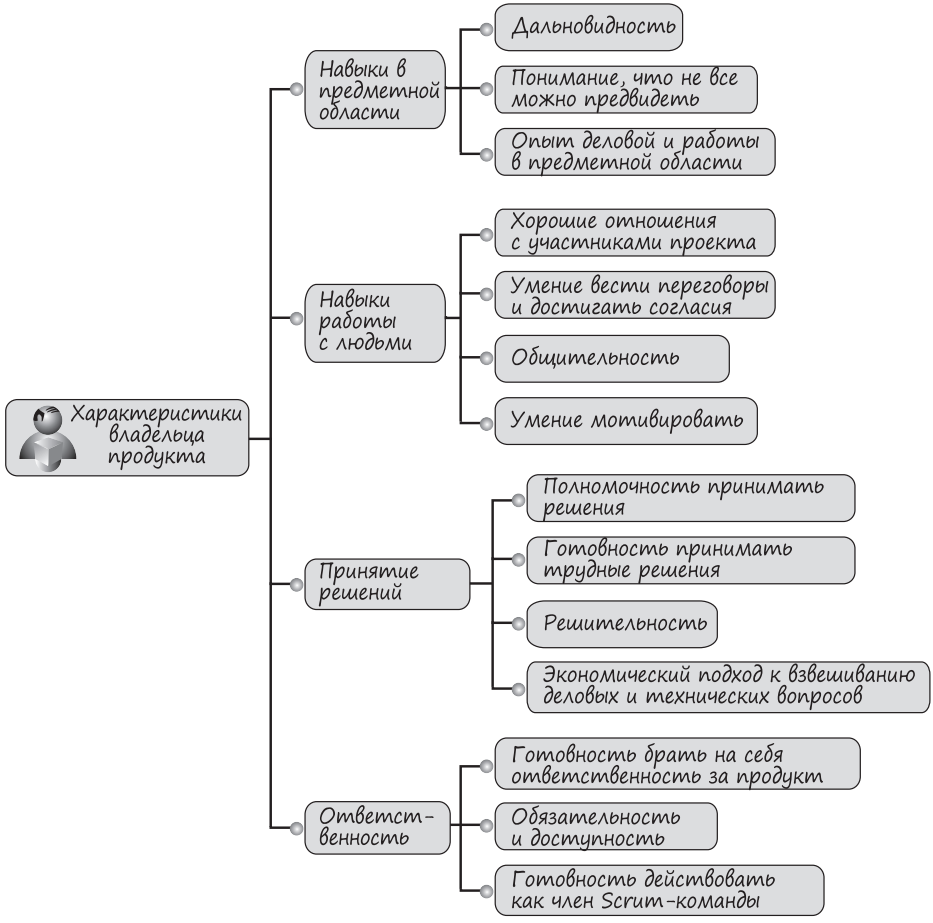


Рис. 9.5. Характеристики владельца продукта

Чтобы эффективно формировать свое представление о продукте и достигать его, владелец продукта должен обладать соответствующими знаниями коммерческой деятельности и предметной области. Очень трудно выполнять эффективно роль владельца продукта тому, кто слабо знаком с предметной областью разрабатываемого продукта. Как можно расставить конкурирующие функциональные средства по приоритетам, не зная предметной области?

Навыки работы с людьми

Владелец продукта должен также служить “гласом заказчика”, для чего требуются хорошие отношения с участниками проекта. Зачастую участников проекта оказывается немало, а их потребности нередко вступают в противоречие, и поэтому владелец продукта должен также уметь вести переговоры и достигать согласия.

Владелец продукта служит связующим звеном между всеми участниками проекта и Scrum-командой. И в этом отношении он должен обладать хорошими навыками общения, чтобы плодотворно сотрудничать с обеими сторонами и снабжать их необходимой информацией на понятном им языке. В умении общаться проявляются также следующие качества: готовность разговаривать, даже если этому не способствует существующее положение дел; уверенность в своей правоте; знание предмета; умение объясняться просто, кратко, понятно и внушать доверие.

Кроме того, владелец продукта должен прекрасно уметь мотивировать. Когда дело идет туго, владелец продукта может напомнить людям, почему они должны приложить усилия, и помочь им не терять оптимизм, укрепляя веру в выгодность данного коммерческого предложения.

Принятие решений

Владелец продукта должен быть наделен полномочиями принимать решения. Нередко помехой организациям, внедряющим методику Scrum, служит то обстоятельство, что человек, выбранный на роль владельца продукта, не наделен полномочиями принимать важные решения. В таком случае его нельзя считать владельцем продукта.

Владелец продукта должен быть также готов принимать трудные решения, зачастую находя компромисс между объемом работ, сроками и бюджетом. Такие решения должны приниматься своевременно и не должны отменяться без веских на то оснований. Иными словами, владелец продукта должен проявлять решительность в принятии решений.

Принимая подобные решения, владелец продукта должен сохранять нужное равновесие между коммерческими интересами и техническими возможностями. Несмотря на то что вся Scrum-команда несет ответственность за накапливание неприемлемых величин технического долга в разрабатываемых системах, наивные решения владельца продукта, которые не учитывают последствия на системном уровне, нередко оказывают существенное влияние на конечный результат.

Ответственность

Владелец продукта отвечает за достижение хороших коммерческих результатов. Впрочем, такая подотчетность не освобождает остальных членов Scrum-команды от ответственности за участие в достижении хорошей окупаемости капиталовложений. Но владелец продукта связан обязательствами обеспечить экономически обоснованное использование имеющихся ресурсов и должен нести ответственность, если этого не происходит. Ведь в конечном счете у владельца продукта имеется немало возможностей внести по ходу дела коррективы в задел продукта, переставить приоритеты и даже проследить за полным сворачиванием проектных работ.

Владелец продукта должен быть преданным своей роли и доступным как для участников проекта, так и для остальных членов Scrum-команды. Его роль подразумевает полную занятость. Если он будет выполнять ее не все рабочее время, он не справится со своими обязанностями.

И наконец, владелец продукта является членом Scrum-команды, а следовательно, он осознает, что достичь хороших экономических результатов невозможно без коллективных усилий всей Scrum-команды. Поэтому владелец продукта относится с уважением к команде разработчиков и Scrum-мастеру и доверяет им как партнерам в достижении желаемых результатов. Совместно все члены Scrum-команды должны действовать под девизом мушкетеров "Один за всех и все за одного" (подробнее о этом — в главе 11). Враждебные отношения между владельцем продукта и остальными членами Scrum-команды недопустимы. Владелец продукта, Scrum-мастер и команда разработчиков должны быть единым целым, работая вместе над достижением общей цели.

Типичный график работы владельца продукта

Чтобы стала понятнее широта охвата обязанностей владельца продукта, рассмотрим его типичный график работы в процессе разработки продукции (рис. 9.6).

В течение первой и второй недели владелец продукта принимает участие в планировании портфеля заказов (см. главу 16) и планировании продукта (см. главу 17). В ходе планирования портфеля заказов владелец продукта должен сотрудничать с ответственным за портфель заказов или коллегиальным органом управления, обсуждая ожидания от портфеля заказов, которые могут оказать влияние на планирование нового продукта. Результаты подобных обсуждений служат основанием для *планирования продукта*, когда владелец продукта вырабатывает вместе с соответствующими участниками проекта общий замысел продукта.

По завершении планирования продукта предлагаемый продукт предоставляется для планирования портфеля заказов, где он проходит через *экономический фильтр* на уровне организации, чтобы определить, следует ли финансировать его разработку, и когда можно приступить к работе. Как показано на рис. 9.6, это происходит сразу же после планирования продукта. Во многих организациях возможны задержки между завершением выработки общего замысла и моментом, когда комитет по утверждениям или коллегиальный орган управления рассмотрит и утвердит финансирование, чтобы приступить к работе.

На третьей неделе владелец продукта принимает участие в планировании первоначального выпуска (см. главу 18). Как правило, такое планирование включает в себя совещание по написанию элементов задела продукта (т.е. историй; подробнее об этом см. в главе 5), где внутренние участники проекта, команда разработчиков, а возможно, и внешние участники проекта составляют

на высоком уровне задел продукта, которым можно было бы пользоваться при планировании выпуска. Члены команды разработчиков должны найти время, чтобы принять участие в этом совещании, поскольку финансирование уже утверждено. Если же команда разработчиков еще не сформирована, то вместо нее в совещании может принять участие замещающая ее команда.

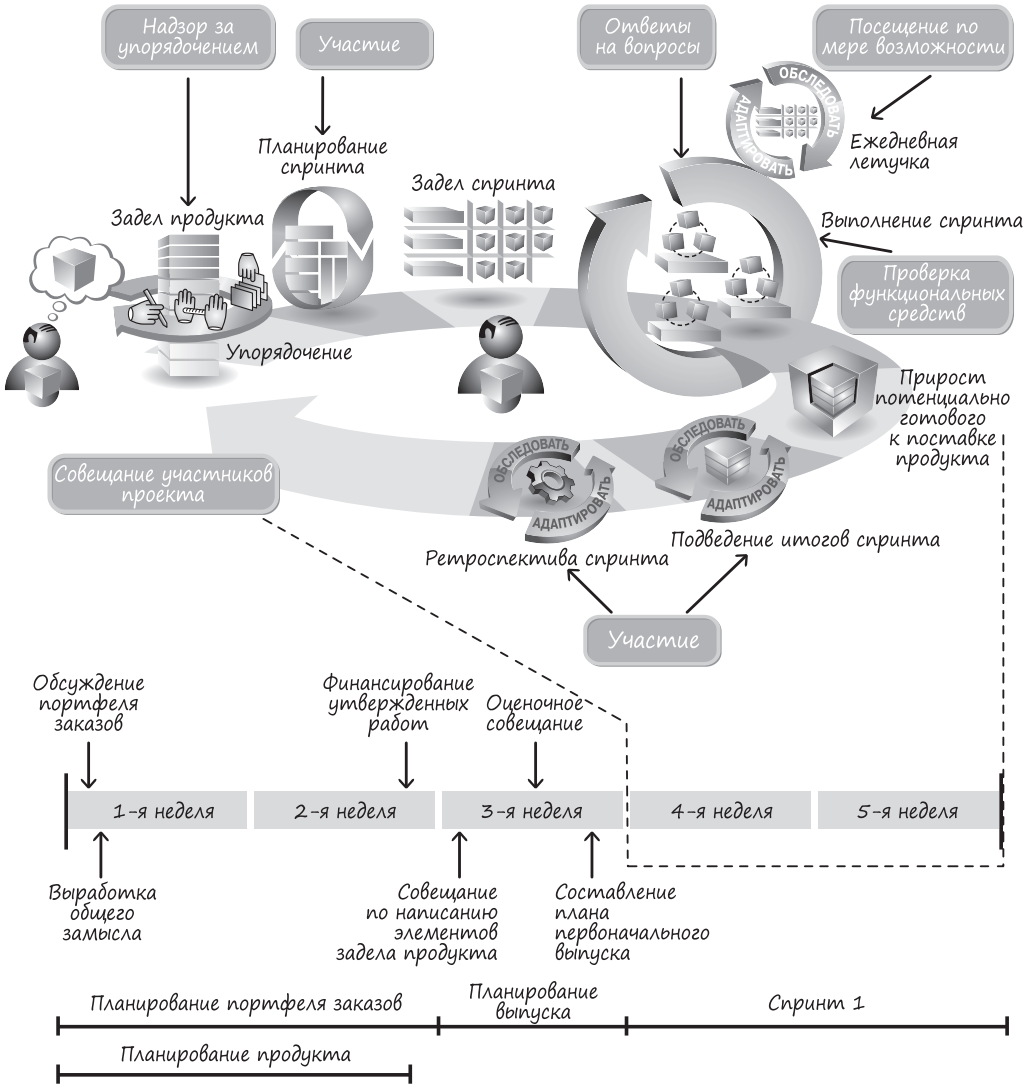


Рис. 9.6. Типичный график работы владельца продукта

После совещания по написанию элементов задела продукта владелец этого продукта принимает участие в оценочном совещании, а возможно, и в нескольких совещаниях в течение одного или двух дней. На этом совещании члены

команды разработчиков (или замещающей ее команды, если она еще не сформирована) оценивают размеры наиболее ценных элементов из задела продукта.

Далее владелец продукта координирует мероприятие по планированию первоначального выпуска, т.е. долгосрочному планированию. А поскольку некоторое количество элементов из задела продукта уже оценено, то данное мероприятие нацелено на расстановку приоритетов в заделе продукта и уравновешивание ограничений на объем работ, сроки и бюджет (см. главу 18). Основными действующими лицами этого мероприятия являются участники проекта, хотя некоторые или даже все члены команды разработчиков могут быть в какой-то момент привлечены к нему с целью выявить технические зависимости, которые могли бы повлиять на порядок расстановки элементов в заделе продукта.

Главная цель планирования первоначального выпуска — добиться приемлемой степени ясности всего выпуска и найти первоначальные ответы на коммерческие вопросы относительно того, что и когда должно быть выпущено. Для большинства продуктов данное мероприятие должно длиться не больше одного-двух дней. Как обсуждается в главе 18, планирование выпуска происходит непрерывно, и поэтому на первоначальном этапе не стоит стремиться к особой точности. Ведь по мере поступления уточняющей информации план выпуска все равно будет обновляться.

По завершении планирования выпуска Scrum-команда выполняет первый спринт (на рис. 9.6 показано, что этот спринт длится в течение двух (4-й и 5-й) недель). В начале спринта владелец продукта следит за деятельностью по планированию спринта (см. главу 19), а в течение периода выполнения спринта (см. главу 20) посещает ежедневные летучки. И хотя это не всегда возможно, но все же рекомендуется делать регулярно. На ежедневной летучке владелец продукта выслушивает мнения членов команды разработчиков, чтобы лучше понять, как продвигается дело в текущем спринте, и найти возможность чем-то помочь команде разработчиков. Так, член команды разработчиков может сказать, что ему не до конца ясны особенности отдельного элемента задела продукта, и поэтому для завершения текущей задачи ему требуются дополнительные разъяснения. Если такое разъяснение можно дать кратко, владелец продукта может сделать это прямо на летучке. А если поставленный вопрос требует более основательного, а не краткого ответа, то владелец продукта должен сказать, что он готов обсудить этот вопрос после ежедневной летучки.

Кроме того, владелец продукта должен быть готов (обычно каждый рабочий день) отвечать на вопросы и проверять функциональные средства по мере их готовности к проверке. Если же владелец продукта знает, что не сможет выполнять эти обязанности каждый рабочий день, он должен поручить их подходящему лицу, чтобы работа команды разработчиков не стопорилась. Более подробно данный вопрос рассматривается далее в главе.

В ходе выполнения спринта владелец продукта встречается как с внутренними, так и с внешними участниками проекта, чтобы правильно расставить приоритеты на предстоящий спринт и получить от пользователей ценные входные данные, которые способны повлиять на выбор функциональных средств для последующих спринтов. Кроме того, владелец продукта регулярно организует упорядочение задела продукта, включая написание новых и уточнение уже имеющихся элементов в заделе продукта. Он также работает вместе с командой и участниками проекта над оцениванием элементов задела продукта, чтобы правильно расставить их по приоритетам.

В конце спринта владелец продукта принимает участие в следующих двух мероприятиях по обследованию и адаптации: подведении итогов спринта (см. главу 21) и ретроспективе спринта (см. главу 22). По завершении этих мероприятий цикл спринтов повторяется, и владелец продукта принимает участие в планировании следующего спринта.

Кто должен быть владельцем продукта

В большинстве организаций, не практикующих гибкую разработку по методике Scrum, скорее всего, отсутствует должностное лицо, отвечающее роли владельца продукта. Так кто же в организации должен исполнять эту важную роль?

Как упоминалось ранее в этой главе, владельцу продукта нужно одновременно смотреть в двух направлениях: в сторону внешних и внутренних участников проекта, а также в сторону команды разработчиков. Следовательно, роль владельца продукта — сочетать руководство и обязанности, которые традиционно относились к нескольким ролям. В своем наиболее объемлющем выражении роль владельца продукта включает в себя элементы ролей ответственного за выпуск продукции, руководителя проекта (как поясняется далее в главе 13), специалиста по анализу экономической деятельности и специалиста по приемочному тестированию.

Выбор конкретного лица на роль владельца продукта зависит от вида проектных работ и характера деятельности организации. В табл. 9.1 перечислены кандидаты, подходящие на роль владельца продукта для разных видов разработки продукции.

Внутренняя разработка

Для проведения внутренних проектных работ полномочиями владельца продукта наделяется человек из команды лиц, заинтересованной в извлечения выгоды из подобной разработки. Так, если внутренняя команда из отдела информационных технологий разрабатывает систему для отдела маркетинга, то один из работников этого отдела наделяется полномочиями владельца продукта (рис. 9.7).

Таблица 9.1. Владельцы продукта для разных видов разработки продукции

Вид разработки	Кандидат на роль владельца продукта
Внутренняя разработка	Заказчик или его представитель из деловой сферы, заинтересованный в разработке
Коммерческая разработка	Внутренний заместитель конкретных заказчиков и пользователей (как правило, ответственный за выпуск продукции; ответственный за продвижение продукции на рынок или руководитель проекта)
Субподрядная разработка	Заказчик или его представитель из организации, оплачивающей разработку и извлекающей из нее свои выгоды
Архитектурная разработка силами команды для компонентов	Как правило, инженерный работник, способный наилучшим образом расставить по приоритетам технические элементы в заделе продукта

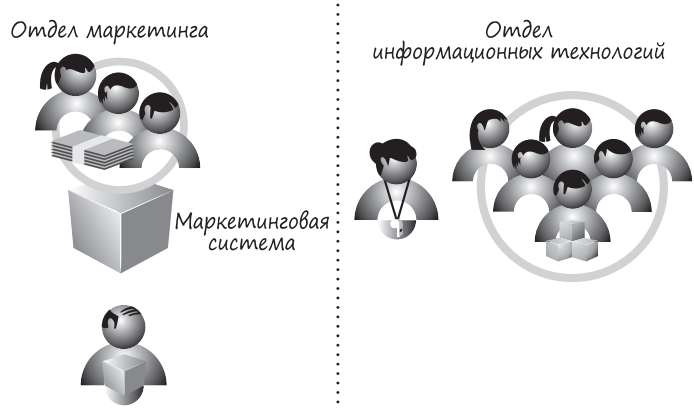


Рис. 9.7. Пример роли владельца продукта при внутренней разработке

В некоторых организациях (как правило, там, где еще не осознали, насколько важно иметь делового человека, постоянно выполняющего роль владельца продукта) для выполнения постоянных обязанностей владельца продукта может быть назначен представитель отдела информационных технологий. При обсуждении понятия команды владельцев продукта далее в этой главе поясняется, насколько оправданным оказывается такое назначение.

Коммерческая разработка

Для проведения коммерческих проектных работ, например, для разработки продукта, предназначенного для продажи внешним заказчикам, на роль владельца продукта следует назначить работника организации, который должен служить в качестве представителя конкретных заказчиков. И зачастую на эту роль назначается работник, отвечающий за выпуск или маркетинг продукции (рис. 9.8).

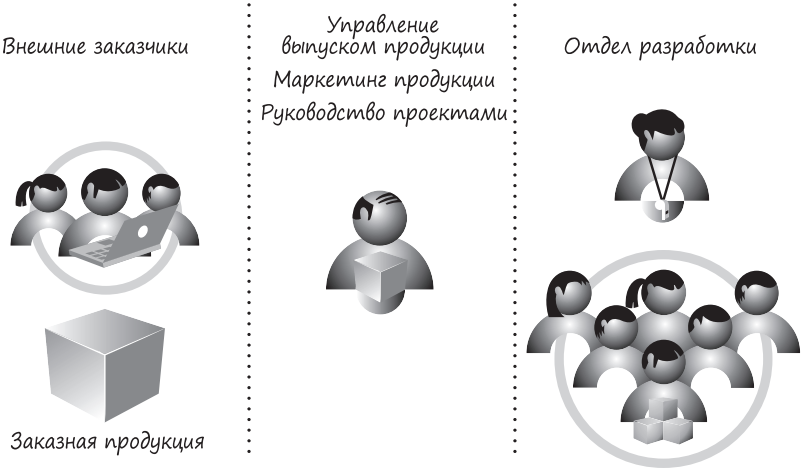


Рис. 9.8. Пример роли владельца продукта при коммерческой разработке

Среди практикующих Scrum разгорелась горячая полемика по поводу того, действительно ли роль владельца продукта служит лишь другим наименованием роли *руководителя* проекта в методике Scrum в частности и гибкой разработке вообще. Одни считают, что эти роли одинаковы, а другие — что роль владельца продукта шире, чем руководителя проекта. И, разумеется, нашлись третьи, считающие, что, наоборот, шире роль руководителя проекта. Ниже я поясняю свой взгляд на роль владельца продукта.

Области управления выпуском и маркетинга продукции весьма обширны. Известная и уважаемая в обеих этих областях компания Pragmatic Marketing, Inc. создала весьма почитаемую инфраструктуру, определяющую роли и обязанности для команд управления выпуском и маркетинга технологической продукции (рис. 9.9).

Как считают в компании Pragmatic Marketing, для охвата всех этих видов деятельности требуется несколько ролей, включая ответственного за стратегию разработки продукции, ответственных за выпуск технической продукции и ответственных за маркетинг продукции. Многие согласятся, что если все эти виды деятельности требуется осуществить в организации для выпуска крупного продукта, то для этого, скорее всего, потребуется целая команда людей.

Следует ли ожидать, что владелец продукта сумеет осуществить все эти виды деятельности? Те, кто считают, что роль владельца продукта является подмножеством традиционной роли руководителя проекта, приводят доводы в пользу того, что владелец продукта выполняет лишь роль "ответственного за выпуск технической продукции", и поэтому он должен сосредоточить свое внимание на небольшом ряде видов деятельности, отмеченных пунктиром на рис. 9.9.

Они считают именно так потому, что владелец продукта должен находиться в ежедневной связи с командой, а следовательно, у него не остается времени на остальные виды деятельности.

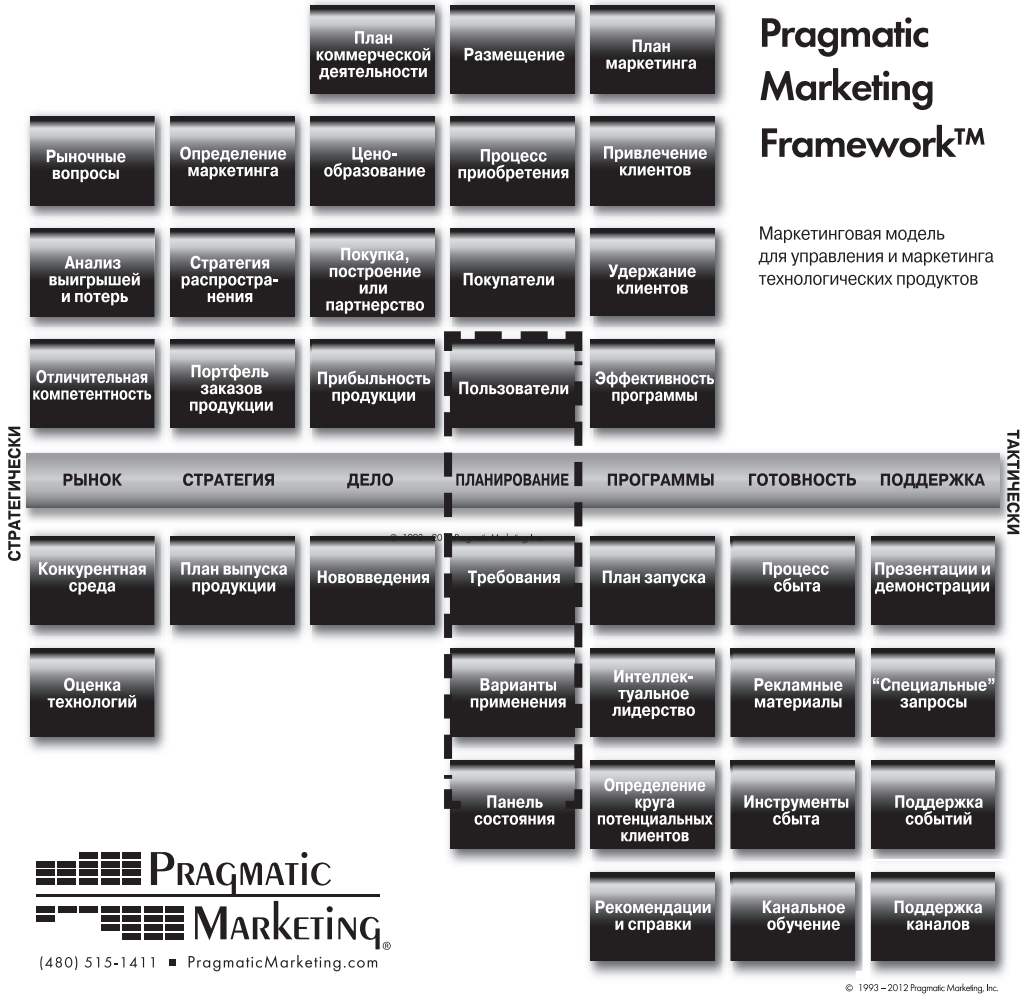


Рис. 9.9. Инфраструктура от компании Pragmatic Marketing, Inc.

Безусловно, владелец продукта отвечает за осуществление видов деятельности, отмеченных пунктиром на рис. 9.9, но я лично считаю, что его роль состоит также в том, чтобы отвечать и за остальные виды деятельности. На мой взгляд, владелец продукта должен отвечать за столько видов деятельности, перечисленных на рис. 9.9, сколько требуется и практически осуществимо для владельца

продукта. Расширение этой ответственности зависит от конкретной организации, самого продукта и профессиональных навыков человека, выбранного на роль владельца продукта. Например, организации, разрабатывающей простое приложение, выполняющее преобразование единиц измерения и предназначенное для сбыта в интернет-магазине приложений мобильных устройств, не потребуется столько видов деятельности, сколько их может потребоваться организации, работающей над следующей новой версией продукта для анализа деловой информации. Таким образом, нет никакого практического смысла глобально определять расширение обязанностей владельца продукта относительно инфраструктуры от компании Pragmatic Marketing.

Как поясняется далее, иногда широта охвата ролью владельца продукта различных видов деятельности может оказаться чрезмерной, чтобы один человек мог справиться с этой ролью. В подобных случаях, возможно, имеет смысл организовать команду владельцев продукта, включив в нее людей, которые должны заниматься исключительно стратегией и маркетингом. Тем не менее в роли владельца продукта для Scrum-команды должен выступать только один человек.

Субподрядная разработка

Для проведения субподрядных проектных работ, например, в компании A, заключающей контракт с компанией B на построение программного решения, владельцем продукта должен быть представитель компании A. Компания B может назначить своего человека для поддержания тесной связи с владельцем продукта, но последний должен быть непременно из организации, оплачивающей программное решение и извлекающей из него свои выгоды (рис. 9.10).

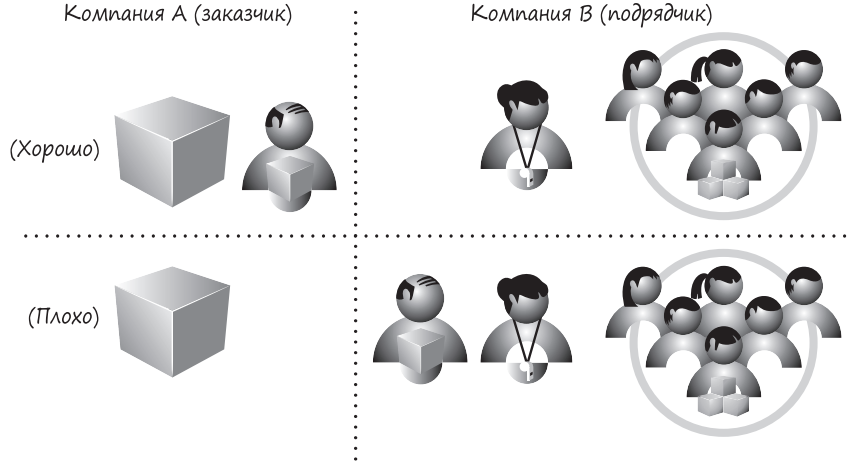


Рис. 9.10. Пример роли владельца продукта при субподрядной разработке

Роль владельца продукта усложняется, если компании A и B заключают традиционный контракт с фиксированной стоимостью разработки. В этом случае в компании B, скорее всего, буду считать, что ей придется выполнять бо́льшую часть обязанностей владельца продукта, поскольку она взяла на себя риск заключить контракт с фиксированной стоимостью разработки. Но в действительности роль владельца продукта должна выполнять компания A, фактически выступающая в роли заказчика. В более подходящем контракте может быть предусмотрено, что компания A нанимает высокопроизводительную команду разработчиков и Scrum-мастера в компании B и предоставляет своего владельца продукта.

Разработка компонентов

И наконец, организации могут создавать команды для компонентов (см. главу 12) ради построения отдельных частей, а не целых программных решений для заказчиков. Такие команды обычно разрабатывают компоненты или иные ресурсы, которые затем неоднократно используются другими командами для сборки программных решений, представляющих ценность для заказчиков. А поскольку работа таких команд сосредоточена только на уровне технических компонентов, то их владельцы продуктов, как правило, являются людьми, ориентированными на решение технических задач, способными определять и расставлять по приоритетам технические средства в своих заделах (рис. 9.11).

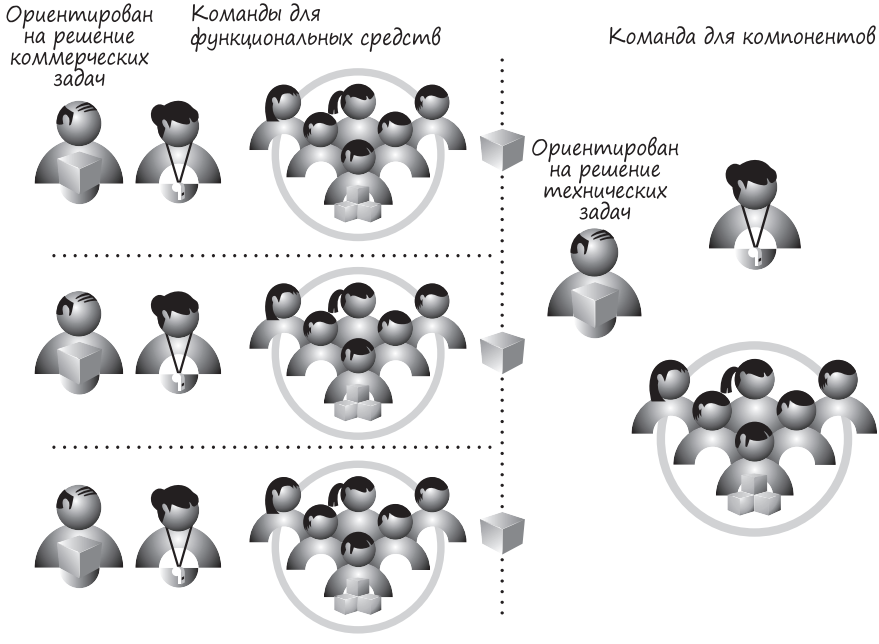


Рис. 9.11. Пример роли владельца продукта при разработке компонентов

В примере, приведенном на рис. 9.11, имеются три команды, ориентированные на решение коммерческих задач и создающие функциональные средства, ценные для конечных пользователей. У каждой такой команды имеется свой владелец продукта, уделяющий основное внимание функциональным средствам, создаваемым в этой команде. Кроме того, каждая команда для функциональных средств опирается на работу команды для компонентов, которая предоставляет ей ресурс, требующийся для завершения создаваемых ею функциональных средств. А команде для компонентов требуется свой владелец продукта, способный расставить компоненты по приоритетам и следить за запросами этих компонентов, поступающих от команд для функциональных средств. Владелец продукта, руководящий командой для компонентов, в большей степени ориентирован на решение технических задач, чем владельцы команд для функциональных средств.

Сочетание роли владельца продукта с другими ролями

Если позволяют способности, один и тот же человек может выполнять роль владельца продукта для нескольких команд (рис. 9.12). Как правило, такому человеку проще быть владельцем продукта нескольких команд в ходе одних и тех же проектных работ, поскольку работа этих команд, скорее всего, очень тесно связана.

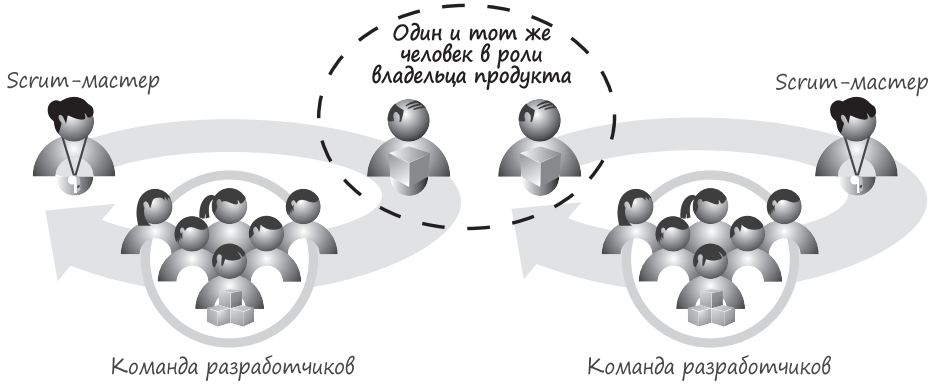


Рис. 9.12. Один и тот же человек в роли владельца продукта нескольких команд

И хотя одному и тому же человеку иногда приходится быть владельцем продукта и членом команды, выполнять одновременно роли владельца продукта и Scrum-мастера все же не рекомендуется. Обе эти роли уравновешивают друг

друга, и если один и тот же человек выполняет обе эти роли, то возникает конфликт интересов, избежать которого очень трудно.

Команда владельцев продукта

В каждой Scrum-команде должен быть лишь один человек, назначенный на роль владельца продукта. И только один человек может быть наделен полномочиями и нести ответственность за выполнение обязанностей владельца продукта в данной Scrum-команде.

Стоит ли поручать роль владельца продукта другим людям? Если под командой подразумевается круг людей, способных коллективно принимать решения и нести ответственность, то определенно не стоит. Для правильного применения методики Scrum требуется *один* человек, выполняющий роль владельца продукта, принимающий решения и представляющий интересы остальных участников проекта в Scrum-команде.

Тем не менее в одних организациях может быть образована так называемая "команда владельцев продукта", поскольку в особых обстоятельствах этих организаций владелец продукта просто не в состоянии справиться со своими обязанностями, не отобрав команду людей для оказания содействия и руководства. А в других организациях нагрузка на владельца продукта может быть больше, чем способен выдержать любой работающий целый день человек. В подобных случаях владелец продукта поручает часть своих обязанностей другим людям. Так или иначе, сформировать команду владельцев продукта вполне допустимо, при условии, что окончательное решение в этой команде принимает один человек, а сама команда владельцев продукта не превращается в комитет по разработке, где каждое решение должно приниматься коллегиально.

Но формируя команду владельцев продукта, следует проявлять особую осторожность. Ведь владельцам продуктов, не умеющим единолично руководить процессом разработки продукции, требуется не совещательный орган, а совсем другая роль. Аналогично владельцам продуктов, слишком занятым, чтобы выполнять свои обязанности, команда может и не понадобиться. Настоящая проблема, вероятно, кроется в том, что организация взялась одновременно выполнять слишком много проектных работ, а владельцев продуктов не хватает, чтобы охватить вниманием нужные продукты.

С другой стороны, разрабатываемый продукт может просто оказаться слишком крупным, и поэтому он должен быть разбит на более мелкие и чаще выпускаемые части. Благодаря такому разбиению на мелкие части одному человеку будет легче справиться с ролью владельца продукта. Если же плохо структурированы команды (см. главу 12) или заделы продуктов (см. главу 6), то

одному владельцу продукта будет трудно справиться со своими обязанностями. Поэтому следует непременно убедиться, что команды владельцев продукта действительно нужны, а не пытаться закрыть ими основную проблему. В противном случае ситуация усложнится и поставит под угрозу общий исход дела.

Заместитель владельца продукта

Как упоминалось ранее, в некоторых организациях, занимающихся внутренней разработкой, на роль владельца продукта назначается сотрудник отдела информационных технологий (например, специалист по анализу коммерческой деятельности или руководитель разработки), поскольку сотрудники служебного подразделения слишком заняты другой работой. Но так как всем известно, что этот сотрудник отдела информационных технологий на самом деле не наделен полномочиями принимать важные окончательные решения, что является одной из главных обязанностей любого владельца продукта, то в таких организациях роль владельца продукта используется неэффективно и неверно. Вместо этого лучше освободить сотрудника служебного подразделения от его обязанностей настолько, чтобы он мог выполнять роль владельца продукта, а сотрудника отдела информационных технологий назначить его заместителем в некоторых вопросах взаимодействия с командой разработчиков.

Заместитель владельца продукта — это лицо, уполномоченное владельцем продукта действовать от его имени в особых случаях. Всем членам Scrum-команды известно, что он не является фактическим владельцем продукта, но им также известно, что владелец продукта наделил своего заместителя полномочиями принимать, по крайней мере, тактические решения от его имени. Подобная ситуация нередко возникает в том случае, если владелец продукта проводит много времени на совещаниях с заказчиками и пользователями, чтобы быть в курсе положения дел на рынке. В таком случае он объективно недоступен для команды разработчиков каждый рабочий день, и поэтому он может поручить своему заместителю постоянно взаимодействовать с командой разработчиков, чтобы решать вопросы, касающиеся элементов задела продукта.

Для того чтобы такой метод руководства разработкой оказался действенным, владелец продукта должен наделить своего заместителя полномочиями принимать решения и не отменять их без веских на то оснований, чтобы не подорвать доверие команды к его заместителю. Но не следует забывать, что владелец продукта не может переложить окончательную ответственность за выполнение работ на своих помощников, которых он в праве наделить соответствующими полномочиями. Ведь именно он отвечает за это.

Главный владелец продукта

Команда владельцев продукта нередко образуется и в том случае, если разрабатываются очень крупные продукты. Как отмечалось ранее, один человек может справиться с обязанностями владельца продукта лишь нескольких Scrum-команд, но как быть, если таких команд много? Мне, например, приходилось обучать и инструктировать сотрудников организации, где проектными работами было занято до 2500 человек. Они были разделены на 250 команд по 10 человек в каждой. Владельцем продукта для всех этих 250 команд не может быть один человек. На самом деле один человек способен выполнять ежедневные обязанности владельца продукта лишь в нескольких командах. Поэтому в подобных случаях роль владельца продукта должна быть расширена иерархически, как показано на рис. 9.13.

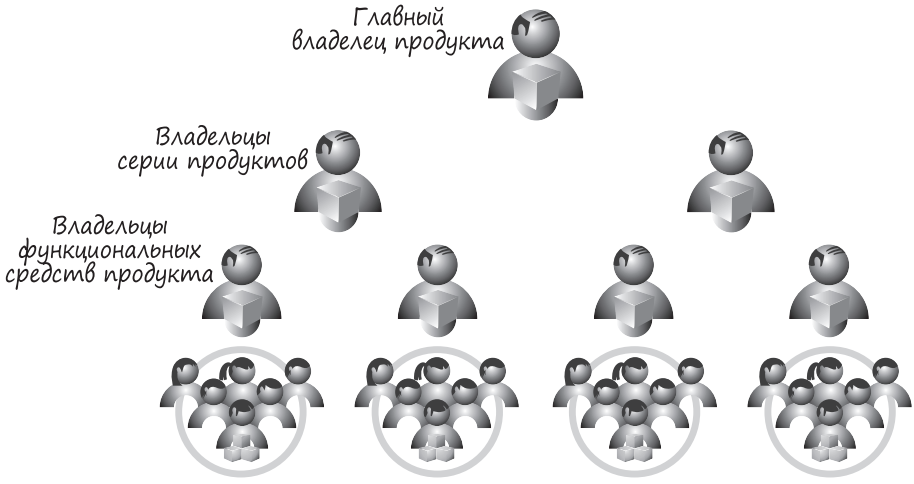


Рис. 9.13. Иерархическая роль владельца продукта

В конечном счете лицо, обозначенное на рис. 9.13 как *главный владелец продукта*, является владельцем всего продукта. Но у главного владельца продукта имеется своя команда владельцев продукта для правильного выполнения его роли на каждом из более низких уровней иерархии. Если выбрать именно такой метод руководства разработкой, то каждый член команды владельцев продукта должен быть наделен полномочиями принимать обширный ряд решений на своем уровне, а не передавать их вверх по иерархии для принятия на более высоких уровнях.

Заключение

В этой главе была подробно описана роль владельца продукта. Сначала в ней было подчеркнуто, что владелец продукта наделен полномочиями руководителя разработкой продукта, а также рассмотрены основные обязанности и характеристики его роли. Затем было пояснено, чем владелец продукта занимается в ходе различных мероприятий, предусматриваемых в Scrum-проекте. Далее в этой главе обсуждалось, кого именно следует назначать на роль владельца продукта для разных видов разработки. После этого пояснялось, как один человек может выполнять роль владельца продукта для нескольких Scrum-команд, а иногда — быть одновременно владельцем продукта и членом той же самой Scrum-команды. И в заключение обсуждалось понятие команды владельцев продукта с акцентом на заместителей и главных владельцев продукта. А в следующей главе будет рассмотрена роль Scrum-мастера.

Глава 10
Scrum-мастер

В этой главе описывается роль Scrum-мастера. Сначала в ней поясняется назначение этой роли по отношению к другим ролям в Scrum. Затем определяются основные обязанности и характеристики роли Scrum-мастера. Далее в главе представлен типичный график работы Scrum-мастера, что, в свою очередь, приводит к необходимости обсудить, следует ли выполнять роль Scrum-мастера весь рабочий день. И в заключение поясняется, кто обычно выполняет роль Scrum-мастера.

Краткий обзор

Scrum-мастеру отведена одна их трех ролей в каждой Scrum-команде (к остальным относятся роли владельца продукта и команды разработчиков). Если владелец продукта сосредоточен на построении нужного продукта, а команда разработчиков — на правильном построении этого продукта, то Scrum-мастер — на помощи всем членам команды в понимании и восприятии ценностей, принципов и норм практики гибкой разработки по методике Scrum. В этом отношении Scrum-мастер служит своего рода наставником как для команды разработчиков, так и для владельца продукта. Scrum-мастер руководит также процессом разработки, помогая Scrum-команде и остальной организации выработать свой собственный, характерный, высокопроизводительный подход к методике Scrum.

Основные обязанности

Основные обязанности Scrum-мастера приведены на рис. 10.1.

Наставничество

Scrum-мастер является в Scrum-команде наставником в гибкой разработке как самой команды разработчиков, так и владельца продукта (подробное описание наставничества в гибкой разработке см. в [Adkins, Lyssa. 2010]). Наставляя коллег выполнять обе эти роли, Scrum-мастер может устранять препятствия между ролями, давая владельцу продукта возможность непосредственно вести процесс разработки.

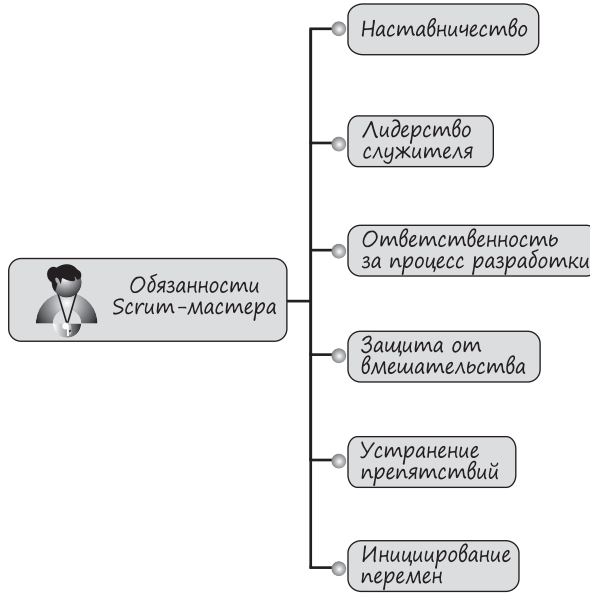


Рис. 10.1. Основные обязанности Scrum-мастера

Подобно тренеру спортивной команды, Scrum-мастер наблюдает за тем, как команда применяет методику Scrum, и делает все возможное, чтобы помочь ей повысить свою производительность до следующего уровня. Когда возникают затруднения, которые команда может и должна разрешить, Scrum-мастер, как любой хороший тренер, обращается к ней с такими словами: "Я здесь не для того, чтобы разрешать затруднения за вас, а для того, чтобы помочь вам их разрешить". Если же возникшее затруднение оказывается препятствием, которое команда не в состоянии устранить, Scrum-мастер берет на себя ответственность за его устранение.

Scrum-мастер наставляет нового владельца продукта, помогая ему лучше понять и исполнять свои обязанности. Помогая владельцу продукта утвердиться в своей роли, Scrum-мастер оказывает ему постоянную помощь в осуществлении таких видов деятельности, как упорядочение задела продукта. Если же продолжить аналогию со спортивным тренером, то отношения Scrum-мастера с владельцем продукта можно сравнить с отношениями главного тренера и владельца команды. Его обязанность — помочь владельцу продукта извлечь наибольшую коммерческую выгоду из методики Scrum, оправдывать ожидания, способствовать тому, чтобы владелец продукта обеспечил команду всем необходимым, а также прислушиваться к жалобам и просьбам владельца продукта внести коррективы, превратив их в практические усовершенствования в работе команды.

Лидерство служителя

Scrum-мастер нередко описывается как *лидер-служитель* Scrum-команды. Даже если Scrum-мастер и действует как наставник команды, он является первым и главным служителем для Scrum-команды, уважая ее наивысший приоритет. В частности, лидер-служитель никогда не обратится к команде с таким вопросом: “Итак, что вы собираетесь сделать для меня сегодня?” Вместо этого лидер-служитель обратится к команде со следующим вопросом: “Итак, что я могу сегодня сделать, чтобы помочь вам и команде работать эффективнее?”

Ответственность за процесс разработки

Scrum-мастер несет перед Scrum-командой ответственность за процесс разработки. Эта обязанность наделяет Scrum-мастера полномочиями предписывать Scrum-команде придерживаться ценностей, принципов и норм практики гибкой разработки по методике Scrum наряду с другими методиками, применяемыми в этой команде. В частности, Scrum-мастер постоянно помогает Scrum-команде совершенствовать процесс разработки при всякой возможности, чтобы доставлять наибольшую коммерческую ценность.

В этом отношении ответственность Scrum-мастера отличается от ответственности руководителя функционального подразделения или проекта. Например, Scrum-мастер не нанимает и не увольняет членов команды и не может навязывать команде, какие именно задачи и как их следует решать. Кроме того, Scrum-мастер не отвечает за то, что работа должна быть сделана, он помогает команде определить собственный процесс разработки и придерживаться его, чтобы сделать порученную ей работу.

Защита от вмешательства

Scrum-мастер защищает команду разработчиков от вмешательства, чтобы не отвлекать ее от доставки коммерческой ценности в каждом спринте. Такое вмешательство может происходить из самых разных источников: от руководства, стремящегося направить усилия команды в другое русло посредине спринта, до других команд, в которых возникают свои проблемы. Но независимо от источника вмешательства Scrum-мастер действует как перехватчик, принимая запросы, разбираясь с руководством и разрешая споры, чтобы команда была сосредоточена на доставке ценности.

Устранение препятствий

Кроме того, Scrum-мастер отвечает за устранение препятствий, мешающих команде работать продуктивно, когда ее члены сами не в состоянии объективно

устранить их. Мне, например, приходилось наблюдать за Scrum-командой, которая постоянно оказывалась не в состоянии достичь цели спринта. И помехой тому служила неустойчивая работа производственных серверов, которыми эта команда пользовалась во время тестирования как стадии проверки соответствия критерию готовности. Сама команда не контролировала работу этих серверов, поскольку это входило в обязанности вице-президента компании, ответственного за эксплуатацию. Команда не могла сама устранить данное препятствие, и поэтому Scrum-мастер взял на себя ответственность за повышение устойчивости работы серверов, сотрудничая с вице-президентом компании, ответственным за эксплуатацию, и прочими заинтересованными лицами, которые действительно могли решить данный вопрос.

Инициирование перемен

Scrum-мастер должен помогать не только в устранении неполадок на серверах и аналогичных им препятствий. Грамотный Scrum-мастер должен также помогать в перемене умонастроения. Он должен решительно бороться со сложившимся положением дел, инициируя перемены, в необходимости которых Scrum-команду не так-то просто убедить. Поэтому Scrum-мастер помогает остальным членам команды лучше понять необходимость предлагаемых им перемен. Этому могут способствовать благотворное влияние методики Scrum за пределами Scrum-команды и заманчивые преимущества, которые Scrum помогает достигнуть. Кроме того, Scrum-мастер обеспечивает эффективность перемен на всех уровнях организации, чтобы достичь успеха не только в краткосрочной перспективе, но извлечь выгоды от методики Scrum в долгосрочной перспективе, что важнее. В крупных организациях Scrum-мастера могут объединять свои усилия, чтобы стать более действенной силой, побуждающей к переменам.

Характеристики и навыки

Наиболее важные характеристики роли Scrum-мастера приведены на рис. 10.2.

Компетентность

Чтобы быть эффективным наставником в процессе разработки, Scrum-мастер должен очень хорошо разбираться в методике Scrum. Кроме того, он должен хорошо разбираться в технических вопросах, которые приходится решать команде разработчиков, а также в технологиях, применяемых командой для создания программных продуктов. Scrum-мастер совсем не обязательно должен разбираться в особенностях руководства проектными работами, но желательно,

чтобы он был технически грамотным специалистом. Он не обязан также разбираться в деловой сфере, поскольку это дело владельца продукта, но практические знания в этой сфере ему все же не помешают.

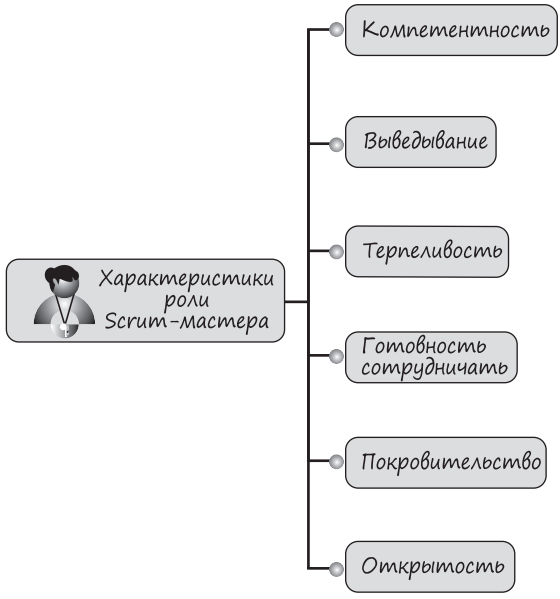


Рис. 10.2. Характеристики роли Scrum-мастера

Выведывание

Scrum-мастера пользуются своими наставническими навыками вместе со знаниями процесса разработки, деловой сферы и технической подготовкой, чтобы ставить перед собой насущные вопросы. Тем самым они проводят внутренний опрос, задавая себе вопросы, заставляющие задуматься и посмотреть на вещи иначе. Грамотный Scrum-мастер никогда не отвечает на вопрос прямо, но вместо ответа рефлексивно задает свой вопрос. Это не досадный или пустой вопрос, задаваемый для проформы, а обдуманный, глубоко прощупывающий, наводящий вопрос, помогающий людям осознать, что они должны проявить проницательность, чтобы самим найти ответ. Это своего рода форма выведывания, которой так славился Сократ.

Терпеливость

Scrum-мастера предпочитают не выдавать сразу же ответы, поэтому они должны быть терпеливы, давая команде возможность самой найти нужные ответы. Мне самому иногда с трудом дается роль Scrum-мастера, поскольку я вижу

вопрос, который команда пытается разрешить, и знаю на него ответ. Во всяком случае, мне кажется, что я знаю ответ! Мне (или любому другому Scrum-мастеру) было бы слишком самонадеянно считать себя умнее команды, как бы превосходя ее коллективный разум. Поэтому я иногда сдерживаю себя и проявляю терпеливость, позволяя команде самой выработать верное решение, но периодически я задаю наводящие вопросы, чтобы направить ее усилия в нужное русло.

Готовность сотрудничать

Scrum-мастер должен обладать отличными навыками сотрудничества, чтобы эффективно работать вместе с владельцем продукта, командой разработчиков и всеми остальными заинтересованными лицами — даже теми, кто, возможно, и не имеет прямого отношения к Scrum. А в качестве наставника в процессе разработки Scrum-мастер должен всегда искать возможность помочь членам Scrum-команды достичь завидного уровня сотрудничества в самой команде. Scrum-мастер может оказать в этом помощь, проявляя собственные навыки эффективного сотрудничества.

Покровительство

Scrum-мастер должен быть надежным покровителем команды. В этом отношении его роль можно сравнить с овчаркой, защищающей стадо овец от волков, которые могут на него напасть. В данном контексте волками могут служить организационные препятствия или люди с другими планами действий. Scrum-мастер должен умело защищать команду и в более широком контексте принятия экономически обоснованных деловых решений. Остро реагируя на необходимость защищать Scrum-команду и коммерческие потребности, Scrum-мастер помогает ей достичь золотой середины.

Кроме того, Scrum-мастер помогает тем членам команды, которые начинают выбиваться из коллектива. Когда возникают трудности, отдельные члены команды могут легко обратиться к уже знакомым, негибким методикам разработки. В таком случае Scrum-мастер обязан помочь этим членам не выбиваться из коллектива и преодолеть возникшие трудности, укрепляя их в умении пользоваться методикой Scrum более эффективно.

Открытость

И наконец, Scrum-мастер должен быть открыт ко всем формам общения. В работе с членами команды у него не должно быть никаких скрытых планов действий. То, что они видят и слышат от Scrum-мастера, то они и должны получить. Члены команды ожидают от Scrum-мастера как минимум действий

лидера-служителя. Кроме того, Scrum-мастер способствует открытому общению за пределами Scrum-команды. Если нет открытого доступа к информации, то организации очень трудно обследовать и адаптироваться, чтобы добиться желаемых коммерческих результатов от применения методики Scrum.

Типичный график работы Scrum-мастера

Какой же типичный график работы Scrum-мастера в течение спринта? На рис. 10.3 показано (хотя и не совсем точно), сколько времени Scrum-мастер вновь сформированной команды может уделять каждому мероприятию на протяжении спринта. В процентном отношение время, уделяемое Scrum-мастером высокопроизводительной команды, работающей вместе с ним несколько лет, может быть иным.

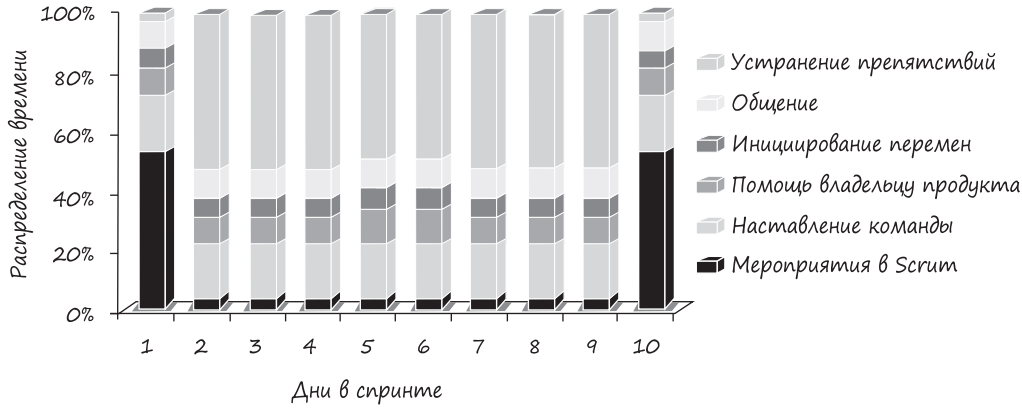


Рис. 10.3. Типичный график работы Scrum-мастера

Как показано на рис. 10.3, Scrum-мастер каждый день уделяет время организации и координации мероприятий в Scrum, включая планирование, выполнение, подведение итогов и ретроспективу спринта, а также ежедневные летучки. К этому относится подготовка мероприятий, надзор за их проведением и предоставление остальным членам Scrum-команды возможности выполнять работу на уровне, позволяющем добиваться весьма ценных результатов.

Кроме того, Scrum-мастер каждый день уделяет время наставлению членов команды, помогая им совершенствоваться в применении методики Scrum и нормах инженерной практики. Scrum-мастер может также проводить повторное обучение, чтобы, например, напомнить новой команде о правилах проведения покера планирования перед оцениванием элементов из задела продукта. Кроме того, часть рабочего дня он должен посвящать общению (например, по поводу обновления спринта, составления диаграмм сгорания и выгорания выпуска или обсуждений с участниками проекта, не являющимися членами Scrum-команды).

В течение спринта Scrum-мастер уделяет время работе вместе с владельцем продукта над упорядочением задела спринта (например, написанию и расстановке по приоритетам новых элементов задела продукта). Кроме того, Scrum-мастер находит вместе с владельцем продукта экономически обоснованные компромиссы среди таких важных факторов, как состав функциональных средств, сроки, бюджет и качество. Помимо этого, Scrum-мастер уделяет время инициированию перемен, чтобы помочь организации лучше охватить методикой Scrum всю цепочку создания ценности, включая сбыт, маркетинг, управление кадрами, субподряд и т.д.

Немало драгоценного времени Scrum-мастер ежедневно уделяет устранению препятствий. Он может даже зарезервировать каждый день фиксированное время специально для устранения препятствий. Разумеется, препятствия могут возникнуть в любой момент. Они могут быть крупными и требовать много времени для устранения, и поэтому у Scrum-мастера должна быть возможность динамически перераспределять время на другие мероприятия, чтобы устранить возникающие препятствия.

У большинства команд и организаций, только начинающих работать по методике Scrum, многие препятствия возникают в самом начале проекта, когда они стремятся сосредоточиться на очевидных и легко преодолеваемых препятствиях. Но это совсем не означает, что все препятствия можно легко преодолеть. В действительности на каждом последующем уровне преодоление препятствий нередко оказывается намного более трудным и отнимающим немало времени делом. Устранение препятствий является важной ежедневной обязанностью Scrum-мастера. Оно может легко изменить распределение времени, приведенное на рис. 10.3.

Выполнение роли Scrum-мастера

Рассматривая роль Scrum-мастера, следует пояснить, кто же лучше всего подходит на эту роль, должна ли она выполняться весь рабочий день и можно ли ее совмещать с другим ролями как в Scrum-процессе, так и вне его. Рассмотрим все эти вопросы по очереди.

Кто должен быть Scrum-мастером

В тех организациях, которые начинают работать по методике Scrum, можно и не найти людей на роль Scrum-мастера. Так где же искать Scrum-мастеров? Мне приходилось не раз видеть, как отличные Scrum-мастера получались из людей, выполнявших много других существующих ролей. Одни Scrum-мастера раньше выполняли функции руководителей проектов или ответственных за выпуск продукции, хотя последние больше подходят на роль владельца продукта, а другие были выходцами из среды разработчиков, тестировщиков и прочих

инженерных работников. Если кандидат на данную роль обладает упомянутыми выше характеристиками и готов справиться с теми обязанностями, которые она предусматривает, то он может стать эффективным Scrum-мастером.

В некоторых организациях считают, что Scrum-мастером должен быть технический руководитель или же руководитель разработки. На самом деле эти лица могут стать отличными Scrum-мастерами, но они могут и *не* подойти на эту роль. Те, кому поручено техническое руководство, занимаются этим потому, что они имеют для этого очень хорошую техническую подготовку. Но роль Scrum-мастера не ограничивается только совершенством технической подготовки. Когда технические руководители выполняют роль Scrum-мастера, они по необходимости обеспечивают меньшее техническое руководство. Поэтому, если сделать из них Scrum-мастеров, это может неблагоприятно сказаться на результатах технического руководства. Далее в этой главе обсуждается, может ли член команды разработчиков одновременно выполнять роль Scrum-мастера.

Руководители структурных подразделений или распорядители ресурсов также могут стать успешными Scrum-мастерами, если они обладают подходящими навыками для выполнения этой роли. Лучше всего, если бы такие руководители больше не выполняли обязанности руководства людьми — по крайней мере, не членами их Scrum-команд. У Scrum-мастера нет полномочий руководить людьми, и поэтому членов команды может ввести в заблуждение то обстоятельство, что их Scrum-мастер одновременно является их руководителем. Я предпочитаю избегать подобной ситуации и не вынуждать членов команды отчитываться перед Scrum-мастером. Но в некоторых организациях подобная ситуация может оказаться неизбежной, и тогда им приходится учиться как можно лучше справляться с потенциальным конфликтом интересов.

Должен ли Scrum-мастер выполнять свою роль весь рабочий день

У каждой Scrum-команды должен быть свой Scrum-мастер, но должен ли он выполнять свою роль весь рабочий день? Вероятно, нет. Если члены Scrum-команды работают вместе долгое время и приобрели солидные навыки, применяя методику Scrum, то им может потребоваться меньшее наставничество, чем новой команде людей, которые никогда не работали вместе и еще не освоили методику Scrum.

Несмотря на то что у Scrum-мастера может возникнуть потребность уделять команде каждый день меньше времени по мере приобретения ею опыта совместной работы по методике Scrum, роль Scrum-мастера остается очень важной для успешного внедрения данной методики в организации. Как правило, потребность Scrum-команды в Scrum-мастере со временем уменьшается,

и поэтому он может сосредоточить больше внимания на более обширных организационных препятствиях и стать инициатором перемен в цепочке создания ценности в масштабах всей организации.

Зачастую роли Scrum-мастера требуется посвящать немало рабочего времени. А в тех случаях, когда ей нельзя посвятить все рабочее время, допускаются некоторые совмещения разных ролей.

Сочетание роли Scrum-мастера с другими ролями

Если позволяют способности и возможности, один человек может, конечно, совмещать функции Scrum-мастера и члена команды разработчиков. Но такое совмещение может пострадать от конфликта интересов, когда один человек пытается выполнять две роли. Что, например, делать такому человеку, если он должен как Scrum-мастер провести важное мероприятие вроде устранения препятствий и в то же время выполнить работу на уровне неотложной задачи? И то и до другое важно сделать, поэтому откладывать ни то ни другое нельзя, чтобы не нарушить эффективность работы Scrum-команды. Возможность пойти на компромисс в данном случае затрудняется тем, что препятствия могут возникать непредсказуемо, а их устранение отнимать немало времени. Еще труднее предсказать, сколько времени Scrum-мастеру как члену команды понадобится для выполнения работы на уровне неотложной задачи.

Но в подобной ситуации зачастую применяется совсем другой подход. Так, если человек обладает хорошими задатками Scrum-мастера, то его лучше назначить Scrum-мастером для нескольких команд (рис. 10.4).

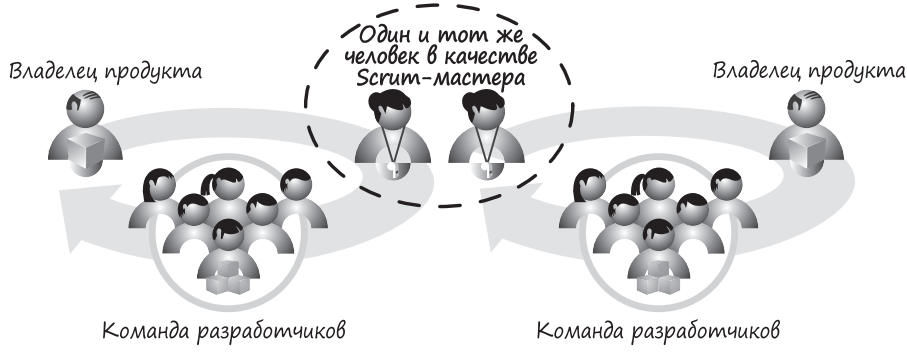


Рис. 10.4. Один и тот же человек может быть Scrum-мастером нескольких команд

Чтобы стать грамотным Scrum-мастером, требуется приобрести ценные и необычные навыки. На мой взгляд, человек с такими навыками должен опекать несколько команд, а не уделять время работе, не связанной с ролью Scrum-мастера. Впрочем, это дело личных предпочтений. Мне приходилось наблюдать

за Scrum-командами, в которых применяются оба рассматриваемых здесь подхода. Общего и правильного критерия выбора одного из этих подходов не существует, хотя такой выбор возможен в контексте конкретной организации.

Как упоминалось в главе 9, совмещать роли владельца продукта и Scrum-мастера настоятельно *не* рекомендуется. Ведь Scrum-мастер является наставником Scrum-команды, а это означает, что он является наставником и владельца продукта. В таком случае быть наставником крайне трудно. Кроме того, владелец продукта несет настоящую ответственность за продукт и может предъявлять соответствующие требования к команде. А Scrum-мастер зачастую действует как посредник, уравновешивающий требования владельца продукта с потребностями и возможностями команды разработчиков. Таким образом, совмещение ролей владельца продукта и Scrum-мастера может только внести путаницу там, где можно поступить проще.

Заключение

В этой главе описана роль Scrum-мастера. Сначала в ней были рассмотрены обязанности Scrum-мастера в качестве наставника, лидера-служителя, ответственного за процесс разработки, защитника от вмешательства, устраняющего препятствия, а также инициатора перемен. Затем в этой главе пояснялось, что Scrum-мастер должен быть компетентным в методике Scrum, уметь ставить перед собой важные вопросы, терпеливо ждать от команды решения ее проблем, сотрудничать со всеми участниками проекта, защищать команду от непомерного влияния и общаться в ясной и открытой форме. Далее в ней описывалось, каким образом рабочее время Scrum-мастера распределяется на протяжении спринта, чтобы глубже понять важность его роли. И в заключение обсуждалось, кто в организации должен выполнять роль Scrum-мастера, следует ли ее выполнять все рабочее время и каким образом роль Scrum-мастера может совмещаться с другими ролями в Scrum. А в следующей главе описывается роль, которая принадлежит команде разработчиков в Scrum.

Глава 11
Команда разработчиков

В этой главе описывается роль команды разработчиков. Сначала в ней рассматриваются пять основных обязанностей этой роли, а затем описываются десять характеристик, которыми должна обладать команда разработчиков.

Краткий обзор

В традиционной разработке программного обеспечения определены различные виды должностей, в том числе архитектора, программиста, тестировщика, администратора базы данных, разработчика пользовательского интерфейса и т.д. А в гибкой разработке по методике Scrum определена роль команды разработчиков, которая представляет собой межфункциональное собрание людей со всеми этими должностями. В частности, команде разработчиков принадлежит одна из трех ролей в каждой Scrum-команде. Совместно члены команды разработчиков обладают навыками, необходимыми для доставки коммерческой ценности, затребованной владельцем продукта.

Термином *команда разработчиков* может неверно обозначаться команда, состоящая не только из разработчиков. Для ее обозначения употребляются и другие термины, в том числе *команда доставки*, *команда разработки построения и тестирования* или просто *команда*. Не совсем очевидно, что любой из этих терминов более подходящий, ясный или простой в употреблении, чем термин *команда разработчиков*. Но поскольку в настоящее время в среде практикующих методику Scrum укоренился термин *команда разработчиков*, то именно он и употребляется в данной книге.

Команды с конкретными ролями

Во многих организациях привыкли намеренно разбивать разные должностные роли на специализированные команды с конкретными ролями. В таких организациях может быть одна команда проектировщиков, другая команда разработчиков и третья команда тестировщиков. Эти команды передают завершенную работу друг другу и функционируют более или менее независимо друг от друга.

А в гибкой разработке по методике Scrum команда разработчиков должна выполнять всю работу, производя в каждом спринте один или больше вертикальных срезов функциональных возможностей рабочего продукта, включая проектное решение, разработку, интеграцию и тестирование этих функциональных возможностей. Следовательно, требуется команда, обладающая навыками решать все эти задачи.

В некоторых организациях пытаются сохранить отдельные команды тестирования или контроля качества, применяя методику Scrum. Я допускаю, что иногда необходимо иметь отдельную команду, занимающуюся только тестированием, поскольку это может, например, предписывать нормативное требование. Но чаще всего в этом нет особой необходимости. Тестирование должно быть полностью вплетено в работу, выполняемую в течение каждого спринта. Поэтому команда разработчиков, выполняющая определенную работу в течение спринта, должна выполнять и тестирование.

Межфункциональные команды следует создавать при всякой возможности. Эффективность передачи работы от одной команды с конкретной ролью к другой вызывает сомнение и, вероятнее всего, служит серьезным препятствием для успешного применения методики Scrum. Поэтому следует непременно убедиться, действительно ли нужно, а не по привычке, сохранять команды с конкретными ролями.

Основные обязанности

Основные обязанности команды разработчиков, связанные с видами деятельности в Scrum, приведены на рис. 11.1. Каждая из этих обязанностей описывается в последующих разделах.

Выполнение спринта

При выполнении спринта члены команды разработчиков занимаются практической, творческой деятельностью по разработке, построению, интеграции и тестированию элементов из задела спринта, осуществляя прирост потенциально готового к поставке продукта по его функциональным возможностям. Для этого они самоорганизуются и коллективно решают, как спланировать, провести, выполнить и передать свою работу (подробнее об этом — в главе 20). Команда разработчиков уделяет б*о*льшую часть своего времени выполнению спринта.

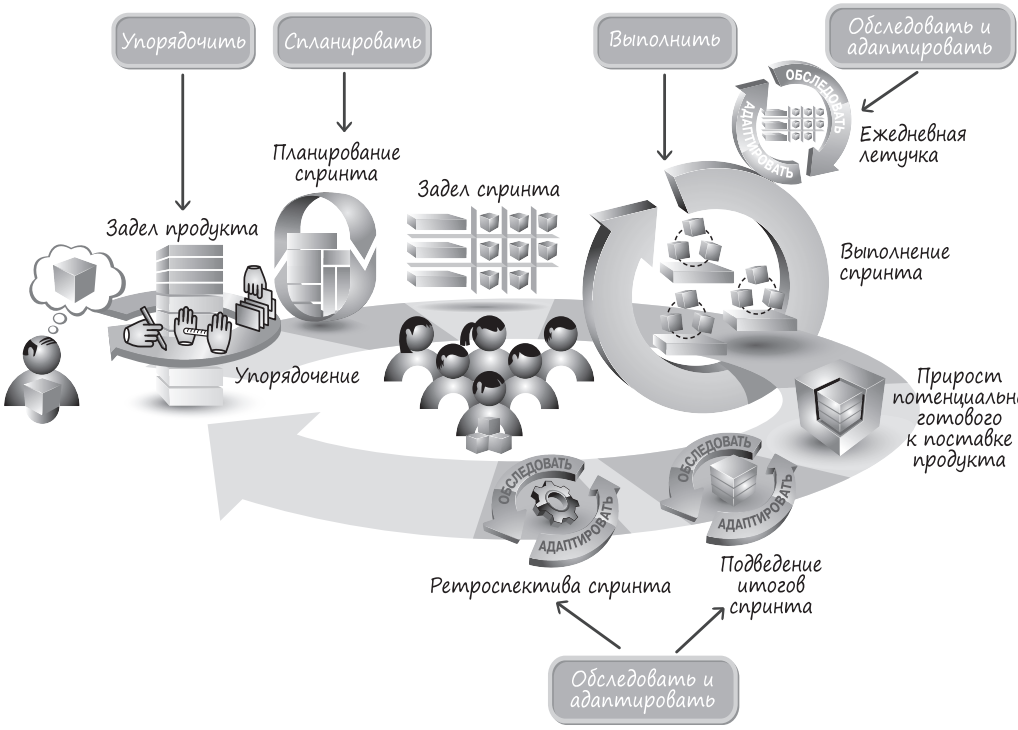


Рис. 11.1. Основные обязанности команды разработчиков и их связь с видами деятельности в Scrum

Обследование и адаптация каждый день

Предполагается, что каждый член команды разработчиков должен регулярно участвовать в ежедневных летучках, на которых члены команды совместно обследуют прогресс, достигнутый в продвижении к цели спринта, а также адаптируют план работы на текущий день. Если некоторые члены команды не принимают участие в ежедневных летучках, команда может упустить важные фрагменты из общей картины, что не позволит ей достичь поставленной цели спринта.

Упорядочение задела спринта

Часть одного спринта должна быть посвящена подготовке к следующему спринту. Большая часть этой работы сосредоточена на упорядочении задела спринта, которое включает в себя создание и уточнение, оценивание и расстановку по приоритетам элементов в заделе продукта (подробнее об этом — в главе 6). Команда разработчиков должна выделять в каждом спринте до 10% своих возможностей, чтобы помочь владельцу продукта в осуществлении этих видов деятельности.

Планирование спринта

В начале каждого спринта команда разработчиков принимает участие в планировании спринта. Сотрудничая с владельцем продукта, команда разработчиков, координируемая Scrum-мастером, помогает наметить цель следующего спринта. Затем команда определяет те высокоприоритетные элементы задела, которые следует создать для достижения данной цели (см. главу 19). Как правило, планирование двухнедельного спринта занимает полдня, тогда как для планирования четырехнедельного спринта требуется целый рабочий день.

Следует заметить, что планирование происходит итеративно. Вместо того чтобы уделять основное внимание очень крупному, неопределенному и слишком детализированному плану в начале проектных работ, команда своевременно составляет ряд более мелких, определенных и менее детализированных планов в начале каждого спринта.

Обследование и адаптация продукта и процесса его разработки

В конце каждого спринта команда разработчиков принимает участие в следующих двух мероприятиях по обследованию и адаптации: подведении итогов и ретроспективе спринта. При подведении итогов спринта команда разработчиков, владелец продукта, Scrum-мастер, попечители, заказчики и заинтересованные лица из других команд рассматривают только что завершенные функциональные средства в текущем спринте и обсуждают наилучшие пути продвижения вперед (см. главу 21). В ходе ретроспективы спринта Scrum-команда обследует и адаптирует свой Scrum-процесс и нормы инженерной практики, чтобы лучше пользоваться методикой Scrum для доставки коммерческой ценности (см. главу 22).

Характеристики и навыки

Наиболее важные характеристики команды разработчиков приведены на рис. 11.2.

Самоорганизация

Члены команды разработчиков самоорганизуются с целью определить наилучший способ достижения цели спринта. У них нет руководителя проекта или другого начальника, который скажет, как им выполнять свою работу, и даже Scrum-мастер никогда на это не отважится. *Самоорганизация* — это свойство организации системы, возникающее по восходящей в отсутствие внешней

господствующей силы, применяемой по нисходящей при традиционном командно-административном управлении.

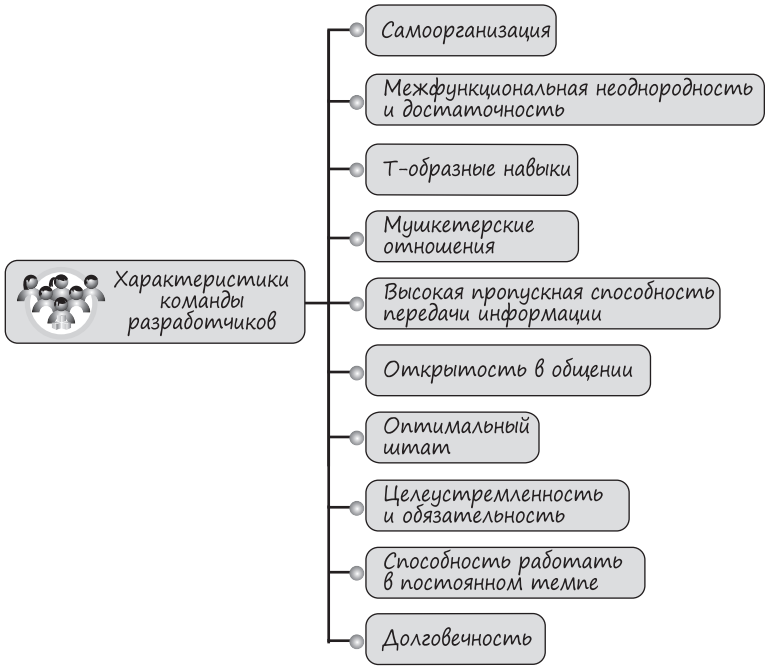


Рис. 11.2. Характеристики команды разработчиков

В качестве иллюстрации позвольте привести конкретный пример. Когда я жил в штате Колорадо, у въезда в мой микрорайон находился пруд, где устраивалась на зимовье стая канадских казарок. Таким образом, каждый год пара сотен казарок одновременно поднимала большой шум, на что было приятно посмотреть. У меня были две собаки, Лети и Тост. Как правило, они находились на заднем дворе за оградой. Но иногда мы разрешали им выйти за ограду и свободно побродить по окрестности. И если они видели казарок на пруду, то сбегали на берег поприветствовать их звонким лаем. Не думаю, что они хотели навредить казаркам, но когда те видели приближающихся к ним Летти и Тоста, то покидали на время пруд, взлетая всей стаей в небо.

Приходило ли вам когда-нибудь в голову, откуда взлетающие птицы знают, что им нужно выстроиться в V-образный клин? Считаете ли вы, что у них есть вожак, который выполняет роль руководителя, созывая птиц на совещание на пруду и инструктируя их, как им собраться в стаю (рис. 11.3)? Прожив у этого пруда много лет, я так и не припомню подобных совещаний, хотя мой сын Иона однажды заявил: “Отец, я никогда этого не видел, потому что они проводят совещание ночью!” Возможно, он в чем-то и прав.

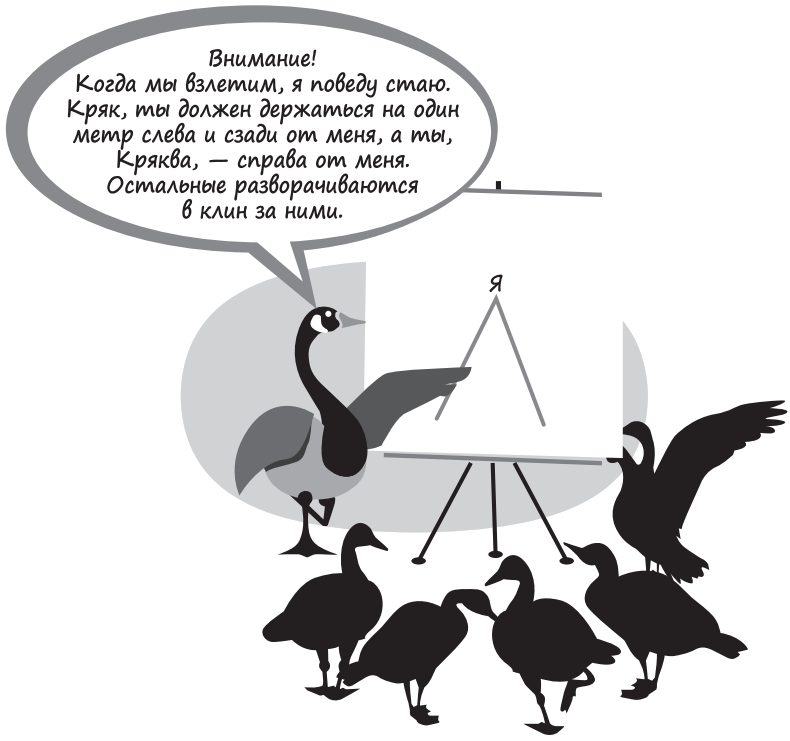


Рис. 11.3. Собирание в стаю не является результатом нисходящего планирования

Но даже если мой сын и прав, а птицы намного более смышленые, чем я думаю, казарки собираются в стаю, проявляя самоорганизацию — развивающееся по восходящей свойство *сложной адаптивной системы*. Многие элементы таких систем взаимодействуют друг с другом самыми разными способами, и эти взаимодействия происходят по простым правилам, действующим в конкретном контексте постоянной обратной связи (рис. 11.4). Подобного рода системы обладают любопытными характеристиками. В частности, они исключительно надежны и порождают поразительные новшества.

Как и у собирающихся в стаю птиц, у команды разработчиков нет командно-административного начальства, которое указывает, как ей следует выполнять свою работу. Вместо этого межфункционально неоднородная команда людей самоорганизуется наиболее подходящим способом для выполнения работы. По существу, самоорганизация команды равнозначна выстраиванию в свой V-образный клин.

Тем не менее руководителям принадлежит важная роль в Scrum. Они создают (или воссоздают) среду для самоорганизации команды. Более подробно роль руководителей рассматривается в главе 13.

Рис. 11.4. Собирание в стаю по простым правилам и с частой обратной связью

Межфункциональная неоднородность и достаточность

Состав команды разработчиков должен быть межфункционально неоднородным. Ее члены совместно должны обладать необходимыми и достаточными навыками, чтобы выполнить порученную им работу. Правильно сформированная команда способна извлечь элемент из задела продукта и произвести качественное, работоспособное функциональное средство, удовлетворяющее критерию готовности, выработанному Scrum-командой.

Команды, составленные из специалистов одинаковой квалификации (т.е. традиционные однородные команды), способны в лучшем случае выполнить лишь часть работы. В итоге одни однородные команды передают работу другим однородным командам. Например, команда разработки передает код команде тестирования или команда проектирования пользовательского интерфейса передает графически оформленные экраны команде построения бизнес-логики. Подобные передачи представляют отличную возможность для возникновения недоразумений и дорогостоящих ошибок. Неоднородность команды позволяет свести к минимуму передачи работ, но ничто не мешает формировать неоднородные команды из многих членов, обладающих высокой квалификацией в одной и той же дисциплине, например, в разработке и тестировании кода на Java или C++. Межфункционально неоднородные команды доставляют также немало перспектив, позволяя достигать лучших результатов (рис. 11.5).

Межфункционально неоднородные команды состоят из членов с разной квалификацией. Каждый член команды вносит свои познавательные средства в решение

проблемы. К числу таких средств относятся разные интерпретации одних и тех же данных, разные стратегии (или эвристические правила) решения проблем, разные модели осмысления механизмов работы и разные предпочтения методов и решений. Подобное разнообразие обычно приводит к лучшим результатам, ускоряя решения, повышая качество поставляемых продуктов и увеличивая нововведения, а все это вместе повышает экономическую ценность [Page, Scott. 2007].

К неоднородности команде следует стремиться и в отношении возрастного состава с оптимальным соотношением работников младшего и старшего возраста. Если в команде слишком много работников старшего возраста, то в ней могут возникать излишние волнения, как на кухне, где слишком много поваров. А если в команде слишком много работников младшего возраста, то их квалификации может оказаться недостаточно для выполнения порученной работы. Оптимальный возрастной состав команды способствует созданию здоровой атмосферы взаимного обучения.

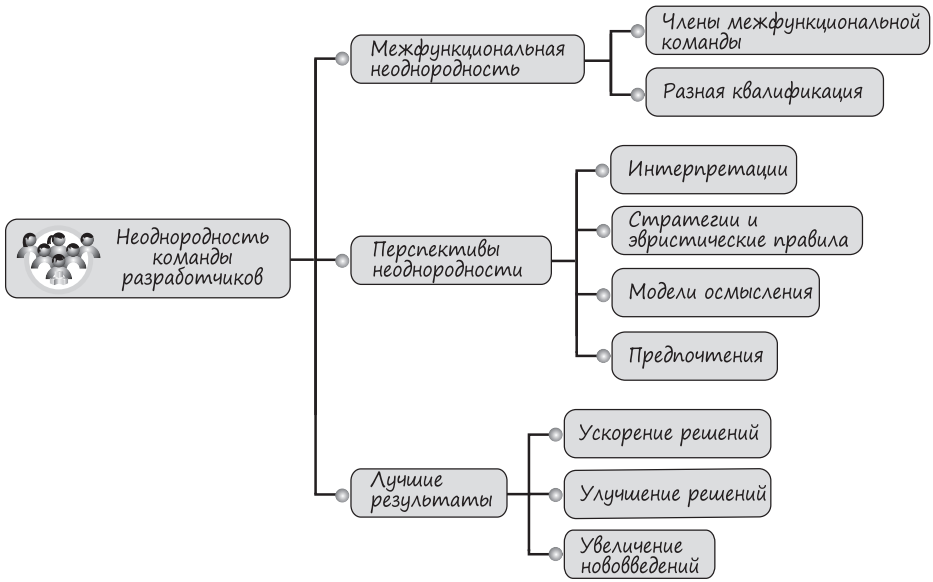


Рис. 11.5. Неоднородность команды разработчиков

T-образные навыки

Команды для гибкой разработки состоят из членов с T-образными навыками (рис. 11.6). *T-образные навыки* означают, что один из членов команды (скажем, Сью) обладает *глубокими* навыками в избранной функциональной области, дисциплине или специальности. Например, Сью является отличным проектировщиком взаимодействия с пользователем. Это ее специальность, по которой она хотела бы работать. Но Сью приходится также работать и не по своей основной специальности,

выполняя тестирование и составляя документацию. Она не обладает такими же навыками тестирования и составления документации, как те, кто специализируются в этих областях, но может помочь протестировать или составить документацию, если команда испытывает острую потребность поручить своим членам выполнение подобных работ. В этом отношении Сью обладает *широкими* навыками, которые позволяют ей работать не по своей основной специальности.

Рис. 11.6. Т-образные навыки

Маловероятно, чтобы каждый член команды мог работать над каждой задачей. Это слишком высокая цель. Например, в таких предметных областях с сильной специализацией, как разработка видеоигр, где команда может состоять из художника, аниматора, звукорежиссера, программиста искусственного интеллекта и тестировщика, было бы неблагоразумно предположить, что каждый член команды может выполнять всякую работу. Если бы я был членом команды, разрабатывающей видеоигру, то мог бы программировать искусственный интеллект и выполнять тестирование, но вряд ли взялся бы за работу над художественным оформлением видеоигры, не имея для этого подходящих навыков, да и мне бы никто этого позволил! Но в то же время я мог бы помочь художникам в выполнении какой-нибудь рутинной работы вроде преобразования файлов изображений из одного формата в другой или написания сценария для автоматизации обработки многих файлов изображений в Photoshop.

Руководители должны формировать команды разработчиков таким образом, чтобы их члены обладали наилучшими Т-образными навыками среди работников данной организации. Но сформировать команду с требующимися навыками не всегда удается сразу, и поэтому такие навыки могут развиваться со временем по мере появления потребностей в них при выполнении проектных работ. Следовательно, руководителям очень важно создать среду, где люди могли бы

постоянно учиться, развивая свои навыки, будь то знания предметной области, технические знания, навыки мышления или другие способности. Руководители должны предоставить членам команды разработчиков время для обучения и экспериментирования (см. главу 13).

Допускается ли в команде наличие узкого специалиста? Допустим, что в предыдущем примере Сью умеет отлично проектировать взаимодействие с пользователем, но и только. А поскольку таких специалистов немного, то вряд ли стоит поручать Сью какую-то другую важную работу, кроме проектирования взаимодействия с пользователем. Ее навыки затребованы в команде, но она может быть задействована в команде лишь на 10% своего рабочего времени. В подобных случаях вполне очевидным решением было бы распределить рабочее время Сью среди нескольких команд.

Но на практике получается совсем иначе. Сью пришлось бы разрываться на части, уделяя по 10% своего рабочего времени нескольким командам. Ее работа очень быстро стала бы узким местом для сотрудничающих с ней команд, как поясняется далее в разделе "Целеустремленность и обязательность". И как упоминалось в главе 3, главная цель состоит не в том, чтобы полностью занять таких специалистов, как Сью, но устранить простои в работе (т.е. поднять лежащую на земле эстафетную палочку), а ведь они возникают, если слишком полагаться на чрезмерно используемый ресурс. Таким образом, рабочее время Сью как узкого специалиста можно распределить среди оптимального количества продуктов, чтобы она не стала для их разработки узким местом (т.е. потерей эстафетной палочки).

С другой стороны, цель состоит в том, чтобы достичь плавного хода работ, выполняемых членами команды с широкими Т-образными навыками. Поэтому нужно поощрять Сью оказывать помощь другим членам команды в приобретении подходящих знаний в области проектирования взаимодействия с пользователем, чтобы больше не полагаться особенно на узких специалистов.

Таким образом, главная цель — сформировать команду из членов с подходящими навыками в основных специальных областях и некоторыми навыками в смежных областях, чтобы обеспечить б*о*льшую гибкость в разработке. Для достижения этой цели многие члены команды разработчиков должны иметь Т-образные навыки, хотя в ней все же допускаются и узкие специалисты.

Мушкетерские отношения

Члены команды разработчиков в частности и Scrum-команды вообще должны относиться друг к другу, как мушкетеры, у которых был девиз "Один за всех и все за одного". Такие *мушкетерские отношения* укрепляют членов команды в мысли, что все вместе они отвечают за выполнение работы. Они добиваются успеха или терпят неудачу не по отдельности, а как вся команда в целом.

В исправно функционирующей Scrum-команде вряд можно ожидать, что кто-нибудь скажет: "Я свою часть сделал, а ты свою часть не сделал. Поэтому

мы и потерпели неудачу". Такое отношение упускает из виду самое главное: все члены команды находятся в одной лодке (рис. 11.7).

Рис. 11.7. Члены команды должны действовать так, как будто они находятся в одной лодке

Члены команды должны понимать, что им нужно работать вместе, чтобы выполнить свои обязательства. Ведь если им этого не удастся, то в конечном итоге это коснется всех. Для достижения общего успеха очень важно, чтобы у всех членов команды были мушкетерские отношения.

Наличие у членов команды Т-образных навыков способствует такому отношению и его применению на практике, поскольку они могут работать не только над одной задачей. В таких командах вряд ли можно услышать от человека, способного сделать работу, что это не его работа.

Но поскольку один человек не всегда может делать всякую работу, то от него можно услышать, что он не в состоянии сделать данную работу. В таком случае команда может назначить для него помощника, не имеющего таких же навыков, как у него, чтобы тот, как подмастерье, смог чему-то у него научиться, а в перспективе совокупные возможности команды расширились бы.

И даже если ограниченные навыки препятствуют членам команды работать межфункционально, они все равно могут организовать свою работу таким образом, чтобы обеспечить ее плавный ход в течение спринта и чтобы никто из них не был перегружен. Например, откладывание всей работы по тестированию на конец спринта, чтобы взвалить ее в конечном итоге на тестировщика, скорее всего, предопределит неудачный исход спринта. Подробнее о том, как команда должна управлять ходом работ при выполнении спринта, речь пойдет в главе 20.

Таким образом, при мушкетерских отношениях никто не оказывается в стороне. Каждый член команды несет ответственность за то, что он задействован в процессе разработки полностью и все время. Зачастую это означает, что нужно высказывать

свое мнение и принимать участие в обсуждении смежных видов работ, чтобы внести в него разнообразие. Например, несмотря на то, что член команды специализируется на тестировании, он должен высказать свое мнение, если заметит недостаток в проектном решении, к которому пришла команда в отношении конкретного функционального средства. Он просто обязан высказать свое мнение, а не заявить, пожав плечами, что это не его работа, которую другие все равно знают лучше.

Высокая пропускная способность передачи информации

Члены команды разработчиков должны общаться друг с другом, а также с владельцем продукта и Scrum-мастером, обмениваясь ценной информацией с высокой пропускной способностью, т.е. быстро, эффективно и с минимальными издержками.

Высокая пропускная способность передачи информации повышает частоту и качество ее обмена. В итоге у Scrum-команды появляется возможность чаще обследовать и адаптировать процесс разработки, а следовательно, лучше и быстрее принимать решения. Экономическая ценность информации зависит от ее своевременности, и поэтому повышение скорости обмена информацией позволяет командам сделать ее максимально ценной. Быстро используя возникающие возможности и распознавая убыточные ситуации, команда может исключить расходование лишних ресурсов как следствие выбора неверного пути.

Команда может достичь высокой пропускной способности передачи информации самыми разными способами. В “Манифесте гибкой разработки” [Agile Manifesto, Beck et al. 2001] заявляется, что предпочтительнее всего личное общение. Безусловно, члены команды, физически разобщенные или пользующиеся недиалоговыми средствами передачи информации (например, документами), находятся в худшем положении, чем те члены команды, которые работают в одном месте и могут сотрудничать непосредственно и в реальном времени.

При всякой возможности я стараюсь, чтобы члены моей команды работали в одном месте. Но во многих организациях по разным коммерческим соображениям созданы распределенные команды, и поэтому их работа в одном месте может оказаться не всегда возможной или даже целесообразной. Мне приходилось работать со многими распределенными командами, выгодно пользовавшимися высокой пропускной способностью передачи информации, а следовательно, личное общение является не единственным способом достижения цели, но оно служит для этого отличной отправной точкой, если позволяют коммерческие условия.

Технологическая поддержка на определенном уровне помогает некоторым распределенным командам повысить пропускную способность передачи информации. У меня имеется опыт работы с организациями, где члены команды были широко рассредоточены. Пользуясь довольно впечатляющим оборудованием для телеконференций, я принимал участие в обсуждениях, которые создавали такое ощущение, будто все участники находились в одном месте. Было ли такое

удаленное общение лучше, чем непосредственное? Нет, не было. Но благодаря технологическим усовершенствованиям пропускная способность обмена информацией между членами команды значительно повысилась.

Наличие команд, состоящих из членов с межфункциональными навыками, является важным шагом вперед в достижении высокой пропускной способности передачи информации. Такие команды имеют более рационализированные каналы связи просто потому, что им легко доступны коллеги, вместе с которыми они должны выполнять работу. Кроме того, таким межфункционально неоднородным командам намного реже придется выполнять формальные передачи работ (как правило, в форме письменных документов) из одной команды в другую. Если же все разработчики относятся к одной и той же команде, частота и формальность передачи работ сокращается, а следовательно, повышается скорость передачи информации.

Следует также сократить время, отводимое на всякие церемонии, где члены команды выполняют процесс, привносящий малую ценность или вообще никакой ценности. Так, если члены команды должны пройти тремя окольными путями, прежде чем они смогут пообщаться с самим заказчиком или пользователем, то такая церемония общения с заказчиком, скорее всего, окажется серьезным препятствием для обмена информацией с высокой пропускной способностью. Необходимость составлять документы, имеющие малую ценность или вообще никакой ценности, или требование длительных и потенциально ненужных процедур утверждения и подписи заметно снижает пропускную способность. Следовательно, нужно выявить и устранить подобные препятствия, чтобы улучшить общую эффективность обмена информацией в команде.

И наконец, наличие небольших команд также улучшает пропускную способность. Каналы связи в команде увеличиваются вместе с ростом количества ее членов не по линейному закону, а нелинейно по следующей формуле: $N(N - 1)/2$. Так, если в команде 5 человек, то в ней имеется 10 каналов связи, а если в ней 10 человек — 45 каналов. Чем больше людей в команде, тем больше издержки на их общение, а следовательно, ниже пропускная способность каналов связи.

Открытость в общении

Помимо высокой пропускной способности (быстроты, эффективности и минимальных издержек), общение в команде должно быть открытым. Открытость в общении дает ясно представление о том, что на самом деле происходит, позволяет избежать неприятных неожиданностей и помогает создать атмосферу доверия среди членов команды. Мне всегда казалось, что команды должны общаться в духе, согласующемся с *принципом наименьшего удивления*. Проще говоря, люди должны общаться таким образом, чтобы как можно меньше удивлять друг друга.

Помню, как в одной Scrum-команде, которую я наставлял, был один такой особый человек, который постоянно подбирал слова на ежедневных летучках,

чтобы скрыть, что же он действительно сделал и что именно собирается делать дальше. Остальные члены команды нередко были удивлены, узнавая впоследствии, что его сообщения были намеренно неясными и вводящими в заблуждение. В конечном итоге они перестали доверять этому человеку, что, свою очередь, послужило препятствием для способности команды к самоорганизации и достижению намеченных целей спринтов.

Оптимальный штат

В методике Scrum предпочтение отдается небольшим командам. Как правило, в команду должно входить от пяти до девяти человек. Заявление о том, что небольшие команды оказываются наиболее эффективными, подтверждается опубликованными исследованиями [Putnam, Doug. 1996; Putnam, Lawrence H., and Ware Myers. 1998]. Мой 25-летний опыт показывает, что для быстрой доставки коммерческой ценности идеальным является штат в пять–семь человек.

Майк Кон [Cohn, Mike. 2009] перечисляет ряд причин для сохранения небольшого штата команд, в том числе следующие.

- Меньшая социальная леность, когда люди прилагают меньше усилий, поскольку они считают, что другие наверстают упущенное ними.
- Конструктивное взаимодействие, наиболее вероятное в небольшой команде.
- Меньшая трата времени на координацию усилий.
- Никто не отходит на задний план.
- Небольшие команды в большей степени удовлетворяют своих членов.
- Излишне узкая и поэтому вредная специализация менее вероятна.

Команды могут быть и совсем небольшими. Например, команда считается совсем небольшой, если в ней не хватает людей для выполнения работы или же если их слишком мало для эффективного функционирования команды.

Но если в методике Scrum предпочтение отдается небольшим командам, то это совсем не означает, что данную методику нельзя применять в более крупных проектных работах. Нередко методика Scrum применяется для построения продуктов, требующих больше 9 людей. Но вместо одной крупной Scrum-команды разработчиков, состоящей из 36 членов, лучше сформировать четыре или больше Scrum-команд, состоящих из 9 или еще меньше людей.

Масштабы Scrum-проекта определяются не крупной командой разработчиков, а количеством небольших Scrum-команд. Многие Scrum-команды могут координировать свои действия самыми разными способами. К числу наиболее распространенных способов относится так называемая *схватка над схватками*, когда члены каждой Scrum-команды собираются вместе для проведения совещания, похожего на ежедневную летучку, только на более высоком уровне (подробнее об этом — в главе 12).

Целеустремленность и обязательность

Члены команды разработчиков должны проявлять целеустремленность и обязательность в отношении цели спринта, намеченной командой. Целеустремленность означает, что каждый член команды поглощен работой, сосредоточив все свое внимание на цели спринта. А обязательность означает, что каждый член команды предан достижению общей для команды цели, как бы ни складывались (хорошо или плохо) обстоятельства.

Если человек работает только над одним продуктом, то ему намного проще проявлять целеустремленность и обязательность. А если его попросят принять участие в нескольких параллельно ведущихся проектных работах, то ему придется делить свое рабочее время между этими работами, сокращая свою целеустремленность и обязательность ко всем работам.

Если спросить любого человека, работающего над несколькими продуктами, о его обязательности, то он, скорее всего, ответит следующее: "У меня столько работы, что я просто стараюсь делать свою работу над каждым продуктом как можно лучше, а затем перехожу к следующему продукту. Я вообще не склонен тратить время, чтобы сосредоточиться на каком-то одном продукте и сделать его как следует. Если бы в работе над несколькими продуктами возникла чрезвычайная ситуация, я просто не смог бы справиться со всеми ними".

Члену команды намного труднее выполнять работу качественно, когда он переходит от одного продукта к другому. Еще труднее одновременно выполнять свои обязательства по отношению к нескольким продуктам. Вместо того чтобы быть в одной лодке с остальными членами своей команды, работник, выполняющий одновременно несколько задач, переходит из одной лодки в другую. Если дадут течь несколько лодок одновременно, то как такому работнику выбрать лодку, чтобы помочь находящейся в ней команде? Если такой член команды отсутствует в лодке, чтобы вычерпывать из нее воду, то он не *проявляет обязательность* к этой команде. В лучшем случае он *участвует* в этой команде. Чтобы быть честным с остальными членами команды, такой член должен ясно дать понять, что он *только* участвует в ней, а следовательно, на него не следует рассчитывать в критические моменты.

Имеется немало данных, подтверждающих мнение о том, что участие в работе над несколькими продуктами (или проектами) или в нескольких командах снижает производительность труда. На рис. 11.8 приведен график, построенный на основании таких данных [Wheelwright, Steven C., and Kim B. Clark. 1992].

Эти данные указывают на то, что никто не может работать продуктивно на 100% над несколькими проектами одновременно, поскольку социальная ответственность влечет за собой определенные издержки. Но, как следует из этих данных, производительность труда оказывается выше, если работать над двумя, а не над одним проектом одновременно. Это происходит потому, что

работа над одним проектом может застопориться, и тогда работник волен перейти к другому проекту, чтобы быть поступательно более продуктивным.

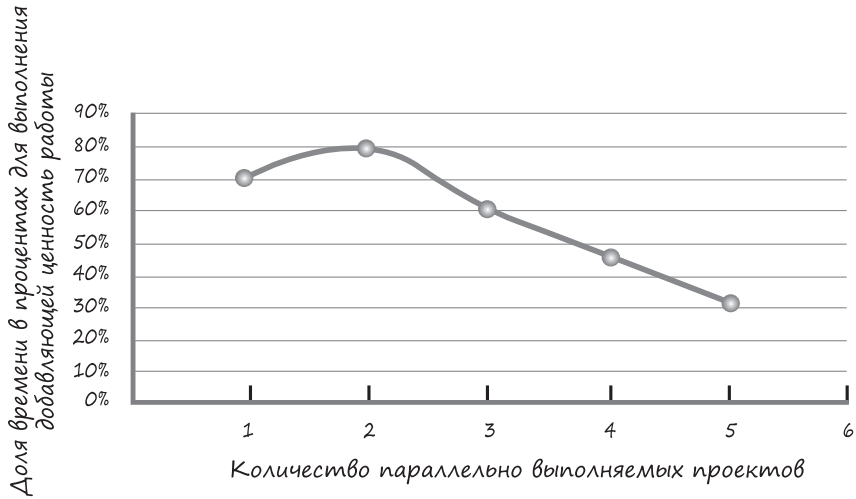


Рис. 11.8. Стоимость одновременной работы над несколькими задачами

Опираясь на эти данные, можно сделать следующий вывод: работать параллельно над тремя или больше проектами экономически невыгодно, поскольку больше времени тратится на координирование, вспоминание и отслеживание информации и меньше времени — на выполнение добавляющей ценность работы. Так над сколькими проектами или продуктами (возможно, в разных командах) должен одновременно трудиться работник? Вероятнее всего, не более, чем над двумя проектами. Я лично предпочитаю работу только над одним проектом или продуктом, потому что быть социально ответственным в современном мире, наполненном разнообразной информацией, получаемой по электронной почте, при обмене мгновенными сообщениями, в социальных сетях вроде Twitter, Facebook и прочих, вероятно, означает то же самое, что и работать над одним проектом!

А как же быть с теми работниками, которым приходится делить свое время между несколькими проектами? Ранее в этой главе был приведен пример Сью в качестве проектировщика взаимодействия с пользователем. Она уделяла 10% своего рабочего времени каждой команде, в работе которой она принимала участие. Конечно, было бы лучше, чтобы Сью работала лишь над одним или двумя продуктами, но что, если потребуется, чтобы она делила свое рабочее время между пятью проектами? С практической точки зрения следовало бы предоставить работнику самому решать, над сколькими продуктами он обязуется работать одновременно и целеустремленно. Если Сью скажет, что она не может больше брать на себя никаких обязательств, то ее не стоит назначать для следующего продукта или команды. А если решение Сью не брать на себя обязательств работать над еще одним продуктом,

поскольку ей удобнее работать, скажем, лишь над тремя продуктами одновременно, оказывается невыгодным с коммерческой точки зрения, то придется, скорее всего, искать альтернативное решение данной проблемы.

Таких решений может быть несколько. Прежде всего, нужно выполнять меньше проектов одновременно. И зачастую такое решение оказывается правильным, поскольку многие организации берутся сразу за слишком много проектов (более подробно этот вопрос обсуждается в главе 16). Другое решение состоит в том, чтобы нанять дополнительных работников, чтобы они разделили между собой бремя ответственности. Третье решение состоит в том, чтобы помочь другим работникам расширить арсенал своих навыков еще одним специальным навыком. И, конечно, четвертое решение состоит в определенном сочетании первых трех решений. В конечном итоге принуждать людей работать одновременно в слишком большом количестве проектов или команд не стоит, поскольку это снижает их целеустремленность и обязательность, а также ставит под угрозу достижение коммерческих результатов.

Способность работать в постоянном темпе

Один из руководящих принципов Scrum заключается в том, что члены команды должны работать в постоянном темпе, и никакой штурмовщины! Благодаря этому они способны выпускать первоклассные продукты, сохраняя здоровую и приятную атмосферу в коллективе.

Применяя методику последовательной разработки, мы откладываем такие важные виды работ, как интеграция и тестирование, ближе к концу, когда обычно накапливается немало вопросов, которые приходится решать по мере приближения срока сдачи. В итоге резко возрастает интенсивность труда на завершающих стадиях проекта (рис. 11.9).

Этот период невероятно интенсивной работы символически выражается сверхгероическими усилиями работников, трудящихся ночами напролет и в выходные дни, чтобы выпустить продукт в срок. Некоторые люди стремятся именно к такого рода работе, им нравится внимание, которое к ним проявляет начальство, а также вознаграждения за их выдающиеся усилия. Но нагрузка на всех остальных оказывается непомерной. Поэтому руководители такой организации должны задать себе вопрос: “Зачем нам работать по ночам и в выходные дни и что мы должны для этого изменить?”

Сравним с этим типичную интенсивность труда по методике Scrum, где разработка, тестирование и интеграция рабочих функциональных средств происходит непрерывно в каждом спринте. В течение каждого спринта члены команды разработчиков должны применять нормы надлежащей практики, в том числе реорганизацию кода, непрерывную интеграцию и автоматизированное тестирование, чтобы доставлять ценность через частые, регулярные промежутки времени, не губя себя.

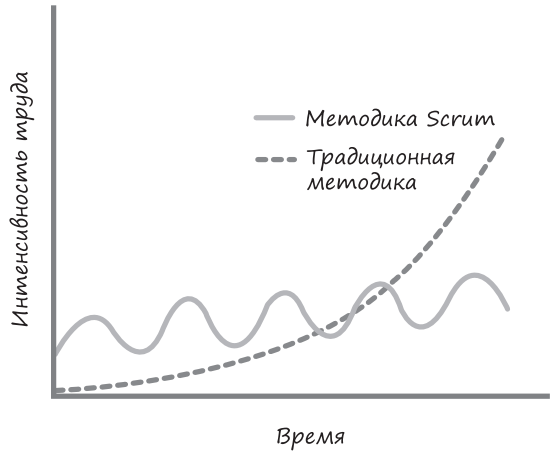


Рис. 11.9. Постоянный темп работы во времени

Таким образом, в течение отдельного спринта интенсивность труда может немного увеличиться ближе к его концу, когда требуется проверить всю связанную с данным спринтом работу на соответствие критерию готовности. Но общая интенсивность труда в течение каждого спринта должна быть приблизительно такой же, как и в предыдущем спринте, укрепляя стремление команды работать в постоянном темпе.

В конечном итоге темп работы выравнивается. Работа выполняется плавно, а не в рваном темпе огромными частями или интенсивными порывами, что особенно вредно на завершающих стадиях. Такое выравнивание темпа означает, что Scrum-командам придется меньше работать сверхурочно, а следовательно, их трудовые ресурсы будут меньше истощены.

Долговечность

Для эффективного применения методики Scrum требуются команды, а не группы. *Команда* состоит из неоднородного межфункционального собрания сотрудничающих вместе специалистов, имеющих общее представление и работающих вместе над воплощением этого представления. А *группа* состоит из собрания людей под общим названием. Помимо общего названия, членов группы больше ничего не связывает, и поэтому они не могут эффективно выполнять обязанности описываемой здесь роли команды разработчиков.

Как правило, команды должны существовать долго. Я лично сохраняю свои команды до тех пор, пока это экономически обоснованно. А экономический анализ показывает, что предпочтение следует отдавать долговечным командам. В исследовании, проведенном Катцом, показано, что долговечные команды более продуктивны, чем вновь сформированные группы [Katz, Ralph. 1982]. Более того, в своем исследовании Штаатс демонстрирует, что знакомство членов команды (т.е.

предыдущий их опыт совместной работы) может положительно сказаться на эффективности и качестве ее конечного результата [Staats, Bradley R. 2011]. А повышение производительности труда, эффективности и качества приводит к улучшению коммерческих результатов.

Если начинать с группы людей, которые никогда не работали вместе, то придется потратить время и средства, чтобы сплотить их в настоящую команду. Большинство групп должны пройти такие стадии, как формирование, штормовое возмущение, нормализация и выполнение, чтобы стать эффективно функционирующими командами [Tuckman, Bruce W. 1965]. Как только хорошо функционирующая команда будет сформирована, ее можно считать настоящим производственным активом. Ее члены знают, как работать вместе, и пользуются взаимным доверием. Кроме того, за время совместной работы команда накапливает важные хронологические данные вроде скорости работы и общей предыстории оценивания (см. главу 7). Если же расформировать команду или значительно изменить ее состав, эти ценные для команды хронологические данные утратят контекст для непосредственного применения.

Слишком часто мне приходилось наблюдать организации, неспособные правильно оценить стоимость активов своих команд. В большинстве организаций выработались привычки и процедуры перемещения людей для формирования "команд", а на самом деле — групп, в динамическом режиме. На мой взгляд, такая практика страдает отсутствием следующей важной особенности Scrum: ценности самой команды, которая является *денежной единицей* гибкой разработки. В действительности одна из основных ценностей, заявленных в "Манифесте гибкой разработки", заключается в "отдельных личностях и их взаимодействиях". Иными словами, команда является ценным производственным активом.

Перемещение людей из одной команды в другую нарушает целостность команды. Сомневаюсь, что состав отряда особого назначения (SWAT) нью-йоркской полиции часто меняется. Члены их команды научились работать вместе, и в трудной ситуации они готовы подставить друг другу плечо. Поэтому перемещение людей из одной команды в другую, скорее всего, нанесет вред доверию, целостности и производительности труда (в данном случае — скорости работы команды, а в таких особых случаях, как отряд полиции особого назначения, — безопасности).

Большинство организаций будут действовать намного лучше, если они возьмут за правило сохранять хотя бы ядро своих команд, чтобы перемещать их от одного продукта к другому. Экономические показатели правильно сформированных команд почти всегда выше, чем те, что дает перемещение людей.

Нельзя сказать, что всегда можно и должно сохранять состав команд в течение продолжительных периодов времени. Так, если имеется команда, которая не сформировалась должным образом или не справляется со своими обязанностями, то ее расформирование нередко оказывается менее вредным и более экономически обоснованным.

В еще одном случае я инструктировал организацию, где высокопроизводительная Scrum-команда была сознательно расформирована в соответствии со стратегией дробления и сеяния, чтобы распространить как можно шире методику Scrum по всей организации. Эта команда была разделена не потому, что она завершила свою работу и настало время сформировать из ее членов новые команды для следующих проектных работ, а потому, что ценнее было не сохранять команду в первоначальном составе, но сформировать из нее шесть новых Scrum-команд, введя в состав каждой из них человека, имеющего опыт работы по методике Scrum

И наконец, команды являются производственными активами, и поэтому они представляют собой единицу производительности, которую следует использовать для установления надлежащего предела на объем незавершенных работ с целью выяснить, сколько и какие виды проектных работ должны выполняться одновременно. Более подробно этот вопрос рассматривается в главе 16.

Заключение

В этой главе описана роль команды разработчиков. Сначала в ней было подчеркнуто, что команда отвечает за превращение элементов задела продукта в приросты потенциально готового к поставке продукта. Затем в ней обсуждались обязанности команды разработчиков в течение каждого спринта. Далее были перечислены десять характеристик, которыми должны обладать команда разработчиков. В частности, от членов команды разработчиков требуется самоорганизация, межфункциональная неоднородность и достаточность профессиональных навыков для выполнения порученной работы. Принимая во внимание характер работы, которую должна выполнить команда, от нее требуется оптимальное сочетание Т-образных навыков, чтобы обеспечить эффективное роящееся поведение. Если люди еще не обладают необходимой широтой своих навыков, их нужно заинтересовать в приобретении такой широты.

От членов команды разработчиков требуются также мушкетерские отношения, когда один за всех и все за одного. Команды следует формировать таким образом, чтобы практиковать и поощрять в них высокую пропускную способность передачи информации. А предпочтение следует отдавать небольшим командам над большими. Чтобы сохранять целеустремленность и обязательность, члены команды должны одновременно работать только над одним или двумя проектами. С точки зрения долговечности команды в нее лучше подбирать таких членов, которые способны работать вместе продолжительный период времени.

Следующая глава посвящена различным структурам Scrum-команд. Этими структурами можно пользоваться на практике, расширяя масштабы применения методики Scrum.

Глава 12
Структуры Scrum-команд

Scrum-команды являются важными производственными активами в организации Scrum-процесса. Их структура и взаимосвязь может оказывать значительное влияние на успешное применение методики Scrum в организации. В этой главе обсуждаются различные способы структурирования Scrum-команд. Сначала поясняется отличие команды для функциональных средств от команды для компонентов, а затем рассматривается вопрос координирования работы нескольких сотрудничающих вместе Scrum-команд.

Краткий обзор

Если вам требуется разработать один небольшой продукт, то материал этой главы вряд ли вас заинтересует. Для целей своего проекта вам достаточно создать одну межфункциональную команду разработчиков, руководствуясь характеристиками, описанными в главе 11, и постараться надлежащим образом выполнить роли владельца продукта и Scrum-мастера. Сделав все это, с точки зрения Scrum-команды вы готовы приступить к разработке!

Но допустим, что созданная вами единственная межфункциональная Scrum-команда превращается в высокопроизводительный механизм для доставки коммерческой ценности, а ваша организация начинает разрастаться. А возможно, вы уже работаете в крупной организации и после разработки первого продукта по методике Scrum ваш опыт ее применения начинает постепенно распространяться по всей организации. В обоих случаях вы довольно скоро обнаружите, что вам приходится координировать работу нескольких Scrum-команд, совместные усилия которых требуются для доставки постоянно увеличивающейся коммерческой ценности.

Как же структурировать эти команды таким образом, чтобы они работали высокоэффективно и согласованно? Для ответа на этот вопрос нужно выяснить, следует ли создавать команды для функциональных средств или же команды для компонентов и какими методами можно воспользоваться для координирования деятельности нескольких команд.

Команды для функциональных средств и компонентов

С одной стороны, *команда для функциональных средств* является межфункциональной и межкомпонентной командой, которая может извлечь функциональные средства конечного потребителя из задела продукта и завершить их. А с другой стороны, *команда для компонентов* сосредоточивает все свое внимание на разработке компонентов или подсистем, которые могут быть использованы для создания только части функционального средства конечного потребителя.

Как обсуждалось в главе 6, производитель устройств GPS может сформировать команду для компонента маршрутизации, чтобы управлять сложным кодом, связанным с определением маршрута от пункта отправления до пункта назначения. Всякий раз, когда появляется запрос на новые функциональные средства, предназначенные для устройства GPS и связанные с маршрутизацией, части этих средств, имеющие отношение к маршрутизации, передаются на разработку команде для компонента маршрутизации.

Команды для компонентов иногда еще называют *командами для ресурсов* или *подсистем*. Зачастую они формируются из людей со сходными специальными навыками (рис. 13.4 в главе 13) и функционируют как команды для компонентов. Все члены таких команд обычно отчитываются перед одним и тем же руководителем функционального подразделения и могут служить в качестве общего, централизованного ресурса для остальных команд. Одним из примеров такой команды может служить централизованный отдел проектирования взаимодействия с пользователем, создающий графическое оформление пользовательских интерфейсов для других команд.

В методике Scrum предпочтение отдается командам для функциональных средств. К сожалению, многие организации предпочитают им команды для компонентов зачастую потому, что, на их взгляд, специалисты, которым доверено вносить безопасные и эффективные коррективы на конкретном участке кода, должны владеть этим участком. Они считают, что люди, незнакомые с кодом, могут неумышленно нарушить его совершенно непредсказуемым образом. Поэтому они предпочитают, чтобы за разработку этого кода и внесение в него корректив от имени других отвечала команда для компонентов.

Допустим, что разрабатывается продукт, функциональные средства которого часто пересекают участки трех компонентов (рис. 12.1).

В данном примере отсутствует команда для функциональных средств, которая могла бы работать над целым элементом задела продукта. Вместо этого функциональное средство извлекается из вершины задела продукта и разбивается на части уровня отдельных компонентов (эти три части обведены пунктиром на рис. 12.1). Такое разбиение делается коллективно членами Scrum-команд для компонентов или же архитектором.

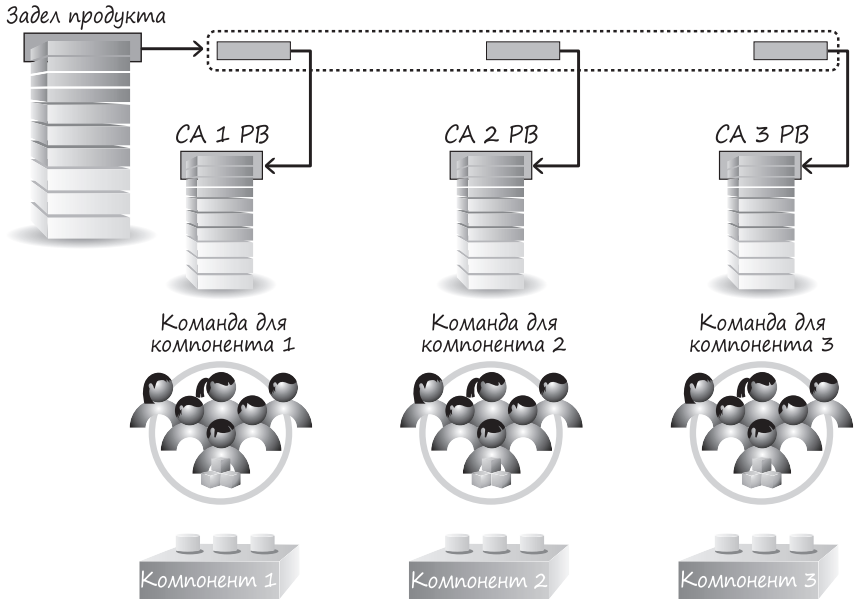


Рис. 12.1. Один продукт и несколько команд для компонентов

Отдельные части функционального средства затем размещаются в заделах соответствующих команд для компонентов (например, первая часть размещается в заделе продукта на участке компонента 1; на рис. 12.1 этот задел обозначен как **СА 1 РВ**). Команда для каждого компонента выполняет Scrum-процесс по отношению к заделу на своем участке компонента, завершая части функциональных средств конечного потребителя, относящиеся к данному конкретному компоненту, но не все функциональное средство в целом. Применяя такую методику, как схватка над схватками, которая рассматривается далее в этой главе, команды для компонентов интегрируют свои части на уровне отдельных компонентов в единое целое и доставляют конечному потребителю полностью готовое функциональное средство.

Если команды для компонентов получают запрос на канализирование только одного продукта, то такая методика, возможно, и окажется пригодной. Но большинство организаций нередко формируют команды вокруг отдельных участков компонентов, которые они намерены повторно использовать в нескольких продуктах. В качестве примера на рис. 12.2 показан ход работ в том случае, если у одних и тех же команд для компонентов имеются запросы на канализирование двух продуктов.

Каждый задел продукта на уровне функционального средства содержит ценные для конечного потребителя элементы, которые могут охватывать участки нескольких компонентов. Поэтому на рис. 12.2 показаны два продукта, которым требуются команды для компонентов, чтобы работать над отдельными их частями на уровне компонентов.

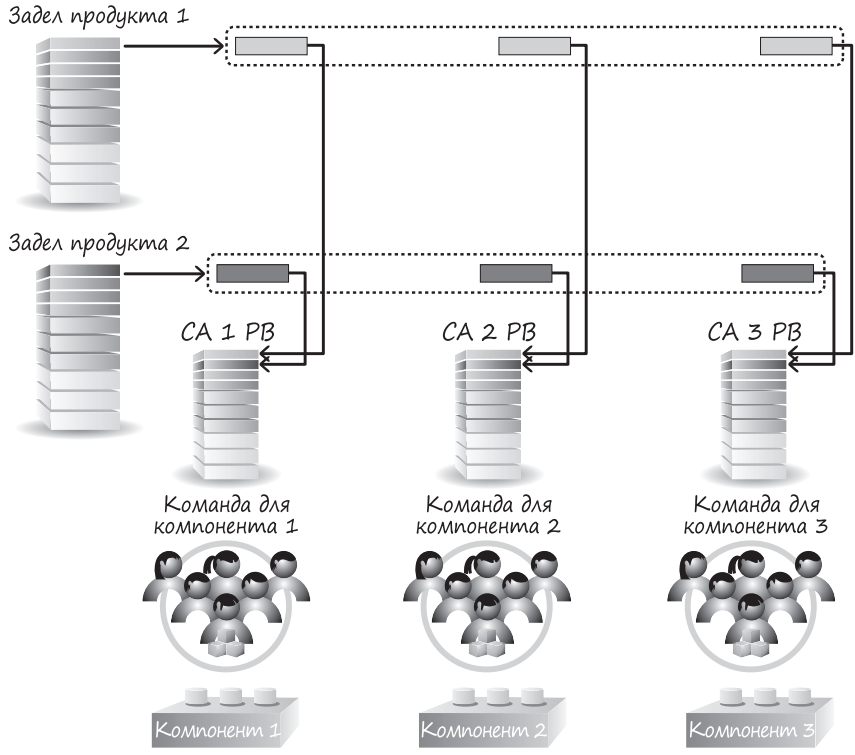


Рис. 12.2. Два продукта и несколько команд для компонентов

Представьте себя на месте владельца продукта одной из команд для компонентов. Вам нужно теперь расставить по приоритетам соперничающие запросы от двух продуктов и в то же время скоординировать работу своей команды с остальными командами на уровне компонентов, чтобы своевременно интегрировать различные части в единое целое.

При наличии двух продуктов логика организации работы еще поддается управлению. Но что, если в организации ведутся работы одновременно над 10 или 15 продуктами и каждый из них разбивается на части уровня отдельных компонентов в заделах продуктов команд для компонентов? При таких масштабах логика организации работ уже не поддается управлению, поскольку нужно организовать в определенном порядке работу над отдельными частями в заделе продукта команды для конкретного компонента и в то же время скоординировать работу с командами для других компонентов, чтобы своевременно произвести интеграцию отдельных частей в единое целое.

Как показывает мой опыт, в большинстве организаций, предпочитающих команды для компонентов, признают трудности, возникающие в тех случаях, когда начинаются простои в работе (эстафетная палочка падает на землю, прерывая

поток доставки ценности). Обычно это происходит следующим образом. Старший руководитель обращается к владельцу продукта на уровне функциональных средств с таким вопросом: "Почему функциональное средство еще не готово для заказчика?" А тот отвечает: "Дело в том, что все команды для компонентов, кроме одной, завершили порученные им части. А поскольку эта последняя команда еще не завершила свою часть, то функциональное средство еще не готово". Тогда руководитель может спросить: "Почему эта команда не завершила часть, которую вы ей поручили?" В ответ он может услышать следующее: "Я их уже спрашивал, и они мне сказали, что у них было 15 соперничающих запросов на изменения, которые следует внести на участке их компонента. И по техническим причинам они посчитали, что целесообразнее обработать запросы от других продуктов прежде нашего. Но они все еще обещают завершить свою частьозможно, в следующем спринте".

Так вести дело нельзя. Ведь нет никакой уверенности, когда именно будет выпущено функциональное средство (если это вообще произойдет), поскольку ответственность за его выпуск распределена среди двух или больше команд для компонентов, у каждой из которых могут быть совершенно разные приоритеты. При такой организации команд для компонентов многократно увеличивается вероятность того, что работа не будет завершена из-за наличия многих уязвимых мест (отдельных команд для компонентов) вместо одной (единой команды для функциональных средств).

Имеется ли выход из этого затруднительного положения? В качестве хорошего выхода можно было бы создать межфункциональные команды для функциональных средств, обладающие всеми навыками, необходимыми для того, чтобы работать над несколькими функциональными средствами конечного потребителя и завершить их, не передавая их по частям командам для компонентов. Но что, если главной причиной, по которой в большинстве организаций создаются команды для компонентов, является наличие единой команды, которой можно доверить работу на участке компонента? Не приведут ли команды для функциональных средств к хаотичности в разработке и сопровождении повторно используемых компонентов с большими объемами технического долга? Не приведут, если имеются правильно сформированные команды для функциональных средств, которые со временем будут разделять общее владение кодом и коллективно станут опекать код. В качестве промежуточного метода на пути к такой модели команды для многих функциональных средств с общим владением кодом служит организация команд, приведенная на рис. 12.3.

Такой метод позволяет снова ввести понятие команды для функциональных средств. Теперь имеется команда для единого функционального средства, которая может извлекать ценное для конечного потребителя функциональное средство из задела продукта. Эта команда несет полную ответственность за выполнение порученной ей работы и управление логикой завершения функционального средства.

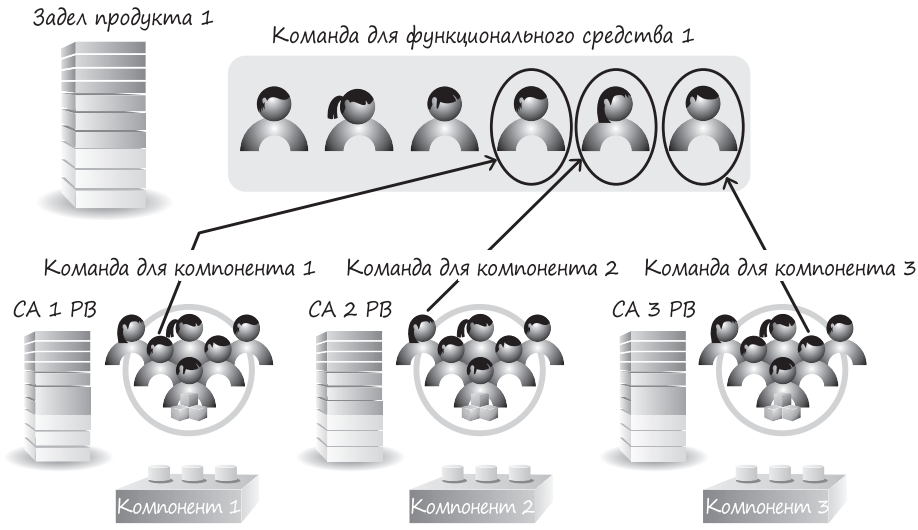


Рис. 12.3. Сочетание команды для функциональных средств с командами для компонентов

В этой модели остаются также заслуживающие доверия команды для компонентов, чтобы помочь сохранить целостность участков отдельных компонентов. Такие команды для компонентов по-прежнему имеют задел продукта, содержащий, как правило, технически ориентированную работу, которая должна выполняться на участке компонента (возможно, это будет работа по возмещению технического долга).

Кроме того, как показано на рис. 12.3, член команды для компонента может быть назначен членом команды для функционального средства. Этот человек несет двойную ответственность как сеятель и жнец [Goldberg, Adele, and Kenneth S. Rubin. 1995].

В роли сеятелей члены команд для компонентов сеют в командах для функциональных средств знания об участках компонентов, чтобы помочь им оказывать большее содействие владению общим кодом в командах для функциональных средств. А в роли жнецов члены команд для компонентов собирают изменения, которые командам для функциональных средств требуется внести на участках компонентов, и обсуждают изменения со своими коллегами из команд для компонентов, каждая из которых также собирает изменения на тех же самых участках компонентов.

В ходе подобных обсуждений члены команд для компонентов могут лучше скоординировать изменения на участках компонентов, которые должны удовлетворять запросам от нескольких команд для функциональных средств. Кроме того, люди, вносящие изменения на участках компонентов, могут делать это согласованно и непротиворечиво, обеспечивая тем самым принципиальную

целостность участков компонентов. А члены команд для компонентов могут также постоянно извещать друг друга о возможностях повторного использования компонентов, поскольку у каждого из них имеется общее представление об изменениях, собираемых на участках компонентов.

Как и при использовании только команд для компонентов, рассматриваемый здесь метод может не действовать в больших масштабах, хотя и по другим причинам, которые можно выяснить. Когда я, например, представлял данный метод в одной крупной компании, они заметили следующее: "Но ведь наши функциональные средства могут охватывать до 50 разных систем [т.е. компонентов]. Мы не можем переместить 50 людей в одну команду для функциональных средств". И хотя функциональное средство может действительно охватывать 50 компонентов, очень редко все эти компоненты должны взаимодействовать друг с другом. В итоге нужно создать не одну команду из 50 человек, но несколько "команд для функциональных средств", формирующихся вокруг более мелких групп компонентов, которым действительно требуется высокая степень взаимодействия (см. соответствующие примеры на рис. 13.5 и 13.6 в главе 13), а затем скоординировать усилия этих команд методами, описываемыми далее в этой главе.

Метод, представленный на рис. 12.3, может не действовать и в том случае, если организация работает над 40 разными продуктами и на участке компонента задействованы только четыре члена команды. В таком случае нет никакого смысла одновременно назначать людей из десяти разных команд для функциональных средств. Но это затруднение можно разрешить, сократив количество параллельно разрабатываемых продуктов (см. главу 16), обучив (или наняв) больше людей, имеющих опыт работы на данном участке компонента, а еще лучше — оказывать большее содействие владению общим кодом, хотя это и долгосрочная задача.

Как показывает мой опыт, универсального решения вопроса выбора между командами для функциональных средств и компонентов не существует. Большинство крупных организаций, успешно работающих по методике Scrum, придерживаются смешанной модели, состоящей, главным образом, из команд для функциональных средств и порой дополняемой командой для компонентов, когда экономически целесообразно иметь такую команду в качестве централизованного ресурса. К сожалению, многие организации предпочитают совершенно противоположное: в основном команды для компонентов, а порой — команду для функциональных средств. Такие организации платят немалую цену в форме задержек, возникающих в часто прерываемом ходе работ.

Координирование работы многих команд

Масштабирование в Scrum осуществляется не укрупнением команд разработчиков, а формированием многих Scrum-команд с оптимальным штатом. Но если

имеется не одна Scrum-команда, то встает вопрос координирования их работы. Для этой цели имеются два метода: схватка над схватками и более полная форма координирования работ многих команд, называемая выпускным поездом.

Схватка над схватками

Как упоминалось в главе 2, каждый день в течение спринта команда разработчиков проводит ежедневную летучку. В каждой такой летучке принимают участие только члены данной Scrum-команды. А для координирования работы многих команд применяется метод, называемый *схваткой над схватками* (SoS; рис. 12.4).

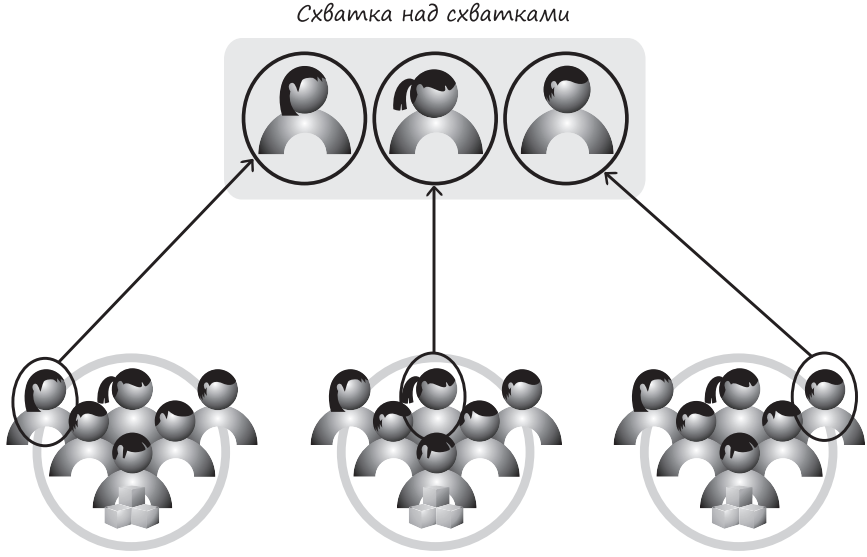


Рис. 12.4. Схватка над схватками

Этот метод позволяет скоординировать работу во многих командах. Та команда, которая проводит схватку над схватками, состоит из отдельных членов из различных команд разработчиков. Каждая команда разработчиков назначает своего члена, чтобы послать его на схватку над схватками, исходя из того, кто способен обсуждать вопросы, возникающие в команде. И хотя я лично предпочитаю иметь постоянного представителя своей команды в схватке над схватками, этот представитель может со временем меняться в зависимости от того, кто лучше всего может представлять команду и обсуждать насущные вопросы в данный момент времени.

Некоторые команды разработчиков посылают на схватку над схватками как своего члена, так и своего Scrum-мастера, который может опекать две или более Scrum-команды. Следует, однако, не допускать, чтобы общее число участников

данного мероприятия оказалось слишком большим. Имеет даже смысл назначить Scrum-мастера на уровне схватки над схватками. Если такая роль существует, то ее может выполнять Scrum-мастер одной из команд или Scrum-мастер, не работающий непосредственно ни с одной из этих команд.

Имеется несколько способов проведения схватки над схватками, и поэтому ее участники должны выбрать наиболее подходящий для них способ. Чаще всего схватка над схватками проводится не каждый день, а несколько раз в неделю по мере надобности. Участники схватки над схватками отвечают на следующие вопросы, аналогичные тем, что обсуждаются на ежедневных летучках.

- Что с момента последней встречи моя команда сделала такого, что могло бы повлиять на работу других команд?
- Что до следующей встречи моя команда собирается сделать такого, что могло бы повлиять на работу других команд?
- Какие проблемы имеются у моей команды, чтобы другие команды могли помочь их разрешить?

Некоторые команды ограничивают время проведения схватки над схватками не более, чем 15 минутами, аналогично ежедневным летучкам Scrum-команд. Они откладывают решение проблемы до тех пор, пока не состоится схватка над схватками, чтобы ее посетили только те участники, которые требуются для решения данной проблемы.

Другой способ состоит в том, чтобы расширить временны*е* рамки схватки над схватками за пределы 15 минут. И хотя участники могут начинать схватку над схватками в пределах 15 минут, отвечая на три перечисленных выше вопроса, она продолжается дальше, чтобы ее участники могли обсудить пути разрешения насущных вопросов.

Теоретически масштабы схватки над схватками можно расширить на несколько уровней. Допустим, что продукт разрабатывается многими командами. Как правило, эти команды собираются вместе в группы по участкам функциональных средств. В каждой группе команд традиционная схватка над схватками может использоваться для координирования работы отдельного участка функционального средства. Имеет также смысл проводить схватку над схватками и на более высоком уровне, чтобы скоординировать работу групп команд. Это так называемая *схватка над схватками схваток*, ради простоты произношения обозначаемая как *схватка на уровне программы*. И хотя такой способ вполне работоспособен, имеются и другие методы координирования работы большого числа команд. Одним из них является обсуждаемый далее выпускной поезд.

Выпускной поезд

Выпускной поезд — это метод согласования представления, планирования и взаимозависимостей многих команд путем синхронизации работы команд, исходя из общего размеренного темпа. Основное внимание в выпускном поезде уделяется быстроте и гибкости хода работ на уровне крупного продукта.

Употребление метафоры поезда означает, что имеется официально опубликованное расписание "отправки со станции" каждого функционального средства. Все команды, участвующие в разработке продукта, должны успеть сесть в свои вагоны поезда в назначенное время. Как и в любой стране с надежным железнодорожным сообщением, выпускной поезд всегда отправляется вовремя и никого не ждет. Если же команда опоздает на выпускной поезд, ей не стоит отчаиваться, поскольку имеется еще один поезд, отправляющийся в известное время в будущем.

Леффингуэлл [Leffingwell, Dean. 2011] определяет следующие правила выпускного поезда.

- Частые, периодически планируемые сроки выпуска решения (или прироста потенциально готового к поставке продукта — PSI) фиксированы (т.е. фиксированы даты и качество, а объемы работ — переменные).
- Команды придерживаются общей продолжительности итераций.
- Устанавливаются промежуточные, глобальные, объективные этапы.
- Непрерывная интеграция системы реализуется на верхнем (или системном) уровне, а также на уровнях отдельных функциональных средств и компонентов.
- Приросты выпусков (или PSI) появляются через регулярные промежутки времени (как правило, через 60–120 дней) для демонстрации заказчикам, внутреннего просмотра и контроля качества на уровне системы.
- Повышающие надежность итерации на уровне системы служат (или могут служить) для сокращения технического долга и предоставления времени для специальной проверки достоверности и тестирования на уровне выпуска.
- Надстройку сходных конструкций и некоторых компонентов инфраструктуры (интерфейсов, наборов инструментальных средств разработки систем, общих утилит установки и лицензирования, каркасов для взаимодействия с пользователем, информационных и веб-служб и т.п.) команды должны, как правило, предусматривать заранее.

На рис. 12.5 показана часть общей картины выпускного поезда, построенной по определению Леффингуэлла.

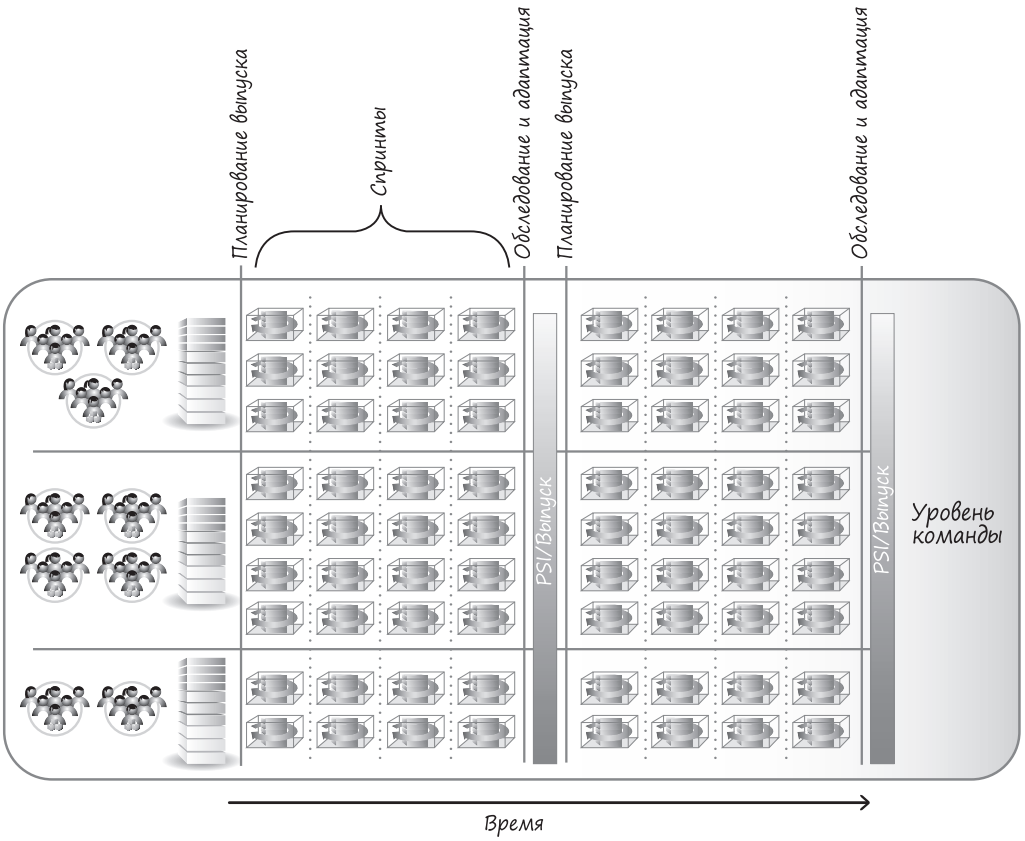


Рис. 12.5. Структура выпускного поезда

Понятие выпускного поезда наполнено многими уровнями детализации, включая уровни портфеля заказов и выпуска. Как упоминалось в главе 6, выпускной поезд опирается на модель задела предприятия, состоящую из следующих трех уровней заделов: задела портфеля заказов, где эпические истории принадлежат руководству портфелем заказов; задела программы, где функциональные средства принадлежат руководству программой; а также заделов команд, где поддающиеся спринту истории принадлежат владельцам продуктов. На рис. 12.5 показан лишь уровень команд. Подробнее планирование на уровнях портфеля заказов и выпуска обсуждается в главах 16 и 18 соответственно.

Выпускной поезд на уровне команд, приведенный на рис. 12.5, состоит из девяти команд, сгруппированных в три участка функциональных средств. В пределах участка функционального средства каждая команда выполняет свой спринт, извлекая работу из задела, связанного с ней участка функционального средства. Применяя такой метод, как схватка над схватками, все команды координируют и интегрируют свою работу в каждом спринте.

Нередко имеет также практический смысл выполнять интеграцию и тестирование по участкам функциональных средств на уровне системы. Некоторые команды резервируют последний спринт перед отправлением выпускного поезда для *повышения надежности* того, что было разработано в предыдущих спринтах, а также для интеграции и тестирования результатов на разных участках функциональных средств (например, четвертый спринт на рис. 12.5 может быть посвящен повышению надежности). По мере совершенствования навыков команды потребность в спринте для повышения надежности сокращается.

Продолжительность всех спринтов для команд, участвующих в выпускном поезде, одинакова, и поэтому все спринты выравниваются. В итоге каждая команда начинает и завершает спринты в одни и те же сроки. Благодаря этому *синхронизация* осуществляется не только на отдельном участке функционального средства, но и среди всех команд, работающих над продуктом.

И наконец, прирост выпуска (PSI — прирост потенциально готового к поставке продукта) появляется после фиксированного количества спринтов (в данном случае их четыре, как показано на рис. 12.5). Зная, что выпуск происходит в надежно установленные моменты времени, организация может синхронизировать свою деятельность по известным, заранее намеченным срокам. В эти моменты выпуска организация может выбрать развертывание прироста выпуска для своих заказчиков, если это коммерчески оправданно, или же воспользоваться этим приростом, чтобы убедиться, что работа, выполненная на отдельных участках функциональных средств, интегрирована и протестирована по этим участкам, а следовательно, можно ходатайствовать о внутреннем просмотре.

Каждый выпускной поезд начинается с совещания по планированию выпуска, в котором принимают участие все команды, работающие над приростом выпуска (PSI; см. рис. 12.5). Это означает, что в мероприятии по совместному планированию могут участвовать сотни людей. Следует признать, что такое зрелище впечатляет. Ниже дается краткий обзор планирования на таком уровне.

Прежде всего, для проведения такого мероприятия требуется большое помещение. Главный владелец продукта (см. рис. 9.13 в главе 9) ведет совещание и, как правило, начинает его. Отдельные члены команд сидят за одним столом или в одной части помещения (желательно рядом с открытым пространством стены, где они могут вывешивать свои артефакты). Scrum-команды из одного участка функционального средства группируются рядом. Как только главный владелец опишет общее положение дел с приростами выпуска, команды собираются по своим участкам функциональных средств. Затем владельцы продукта на участках функциональных средств описывают общее положение дел на своих участках для последующего выпускного поезда.

Далее отдельные Scrum-команды начинают планировать свои спринты, распределяя функциональные средства по отдельным спринтам. Этот вид деятельности называется *построением карты спринтов* и подробнее рассматривается в главе 18. А поскольку Scrum-команды фактически работают над созданием более крупного продукта, поставляемого многими командами, то между ними возникают зависимости. Чтобы справиться с этими зависимостями, в любой момент времени член одной Scrum-команды может встать, подойти к другой Scrum-команде (возможно, неся с собой карточку или наклейку для заметок) и спросить, может ли эта Scrum-команда завершить часть работы, указанную на карточке, в предстоящем выпускном поезде. Если она может это сделать, то запрашивающая команда вправе взять на себя обязательство в отношении зависимого функционального средства.

Между тем люди, отвечающие за многие команды, в том числе главный владелец продукта, владельцы продукта на участках функциональных средств и общие архитекторы проекта, переходят от одного стола к другому, чтобы убедиться, что все участники совещания правильно представляют общее положение дел в проекте и что согласованный общий план на предстоящий выпускной поезд имеет какой-то смысл. Разумеется, любая Scrum-команда может всегда попросить о помощи любого из этих лиц.

Как только спринты в выпускном поезде будут завершены и наступит момент выпуска прироста потенциально готового к поставке продукта (отправление поезда), осуществляются мероприятия по обследованию и адаптации на уровне выпускного поезда. Первым из них является осмотр всего содержимого вагона (т.е. прироста потенциально готового к поставке продукта), входящего в состав выпускного поезда. Затем следует ретроспектива на уровне выпускного поезда, цель которой — сделать более эффективным последующий выпускной поезд. А далее начинается планирование выпуска для следующего выпускного поезда.

Заключение

В этой главе обсуждались разные способы структурирования Scrum-команд. Сначала в ней были рассмотрены команды для функциональных средств, обладающие межфункциональной неоднородностью и достаточностью, чтобы извлечь функциональное средство конечного потребителя из задела продукта и завершить его. Затем было проведено сравнение команд для функциональных средств с командами для компонентов, работающими над конкретными компонентами, ресурсами или участками архитектуры, представляющими лишь отдельные части того, что требуется объединить в функциональные средства конечного потребителя. После этого была продемонстрирована смешанная модель команд для функциональных средств и компонентов и показано, каким образом

она помогает организации перейти на работу только командами для функциональных средств, каждая из которых отлично владеет общим кодом.

Далее в главе было показано, как координируется работа многих Scrum-команд, начиная с метода, называемого схваткой над схватками, и продолжая описанием метода выпускного поезда, которым можно пользоваться для координирования работы большого количества Scrum-команд. А в следующей главе будет рассмотрена роль руководителей в организации Scrum-процесса.

Глава 13
Руководители

Если команды способны самоорганизоваться, то есть ли место руководству в Scrum-процессе? Разумеется, есть. Несмотря на то что в инфраструктуре Scrum не упоминается роль руководства, руководители по-прежнему являются важной частью организации гибкой разработки. Ведь в организациях существуют многие роли, не присущие методике Scrum, но не менее важные для деятельности организации. В частности, бухгалтеру не отведена роль в Scrum-процессе, но мне еще не встречались члены Scrum-команд, которые хотели бы работать бесплатно!

В этой главе обсуждаются обязанности руководителей отдельных структурных подразделений или функциональных областей, называемых также администраторами ресурсов, руководителей разработки, руководителей контроля качества и художественных руководителей в организации Scrum-процесса. И в завершении рассматривается роль руководителя проекта в организации Scrum-процесса.

Материал этой главы имеет непосредственное отношение к тем организациям, которые имеют руководителей структурных подразделений и руководителей проектов. Если ваша организация небольшая, а ее руководство относительно малочисленно, можете пропустить эту главу. Тем не менее вам будет полезно прочитать данную главу, чтобы составить представление о том, что потребуется в дальнейшем, когда ваша организация разрастется.

Краткий обзор

Согласно опросу, проведенному в отрасли гибкой разработки за 2011 год, главным препятствием к внедрению методики Scrum служит ощущение потери административного контроля, как показано рис. 13.1, взятом из [VersionOne. 2011]).

Опасения, что роль руководства станет менее значимой, не обоснованы. В организации, практикующей методику Scrum, руководители продолжают исполнять важные обязанности (рис. 13.2). В частности, руководители функциональных подразделений в такой организации отвечают за формирование команд, опеку над ними, согласование и адаптацию производственной среды и управление ходом работ по созданию ценности.

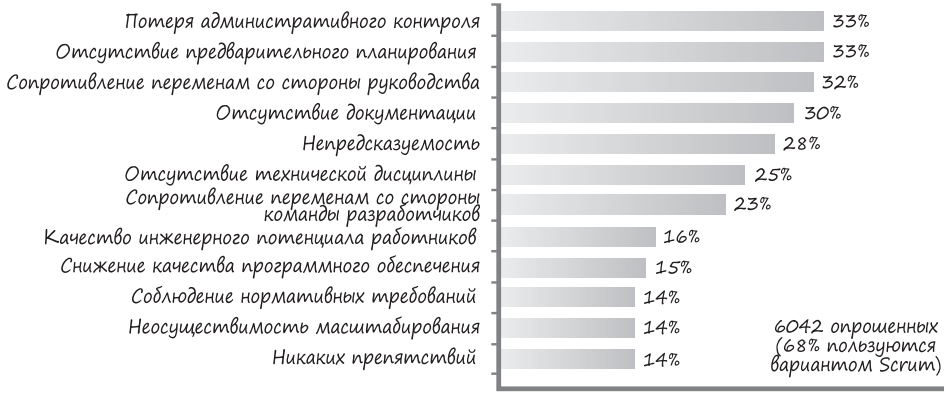


Рис. 13.1. Главные препятствия к внедрению методики Scrum

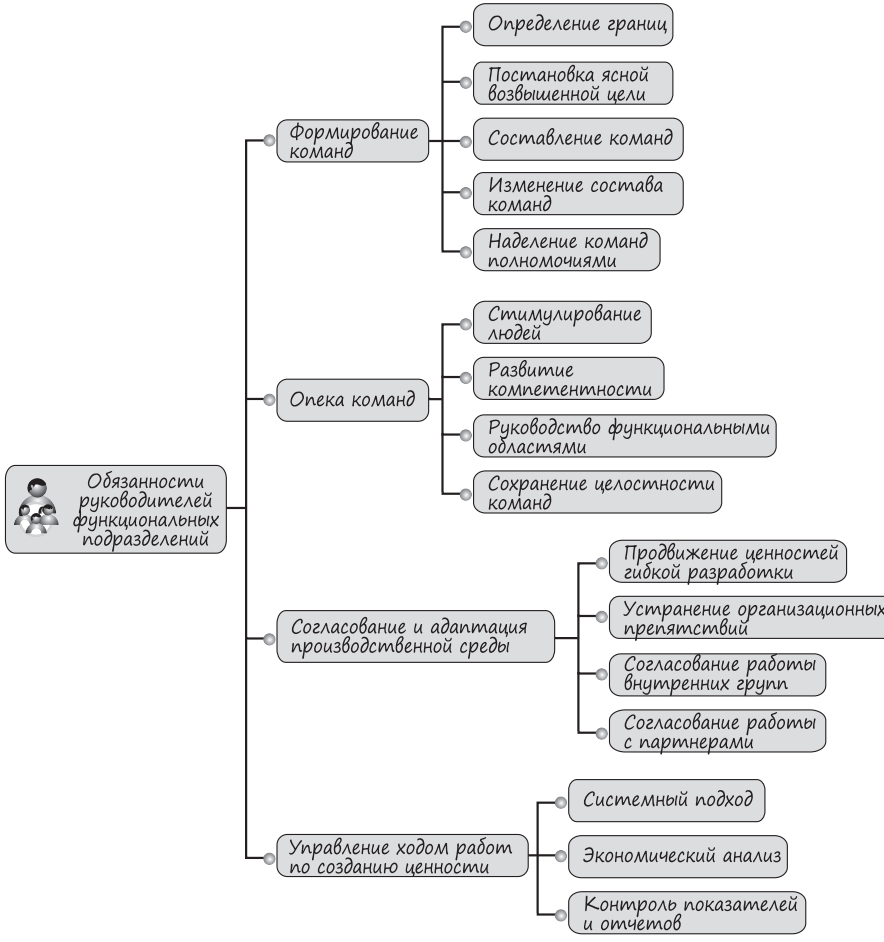


Рис. 13.2. Обязанности руководителей функциональных подразделений в организации, практикующей методику Scrum

Формирование команд

Руководители формируют команды. Этот процесс включает в себя определение границ, постановку ясной возвышенной цели, создание команд, изменение их состава и наделение команд полномочиями.

Определение границ

Как пояснялось в главе 11, самоорганизация команды управляет ее реакцией на среду, в которой она находится. Но сама среда находится под влиянием руководителей (рис. 13.3).

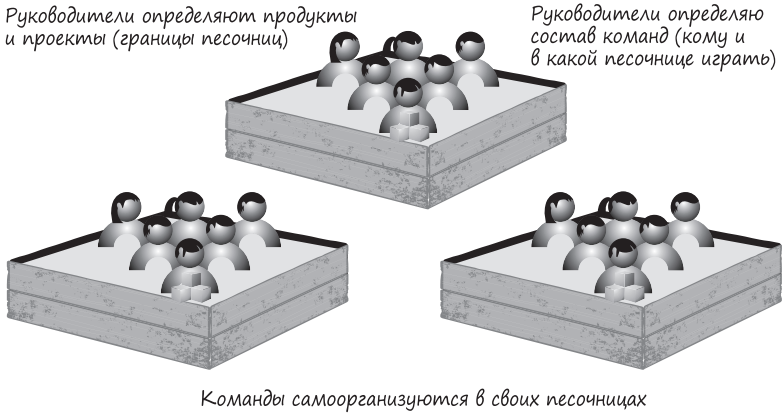


Рис. 13.3. Границы определяют руководители

Самоорганизующимся командам редко приходится решать, какие именно продукты или проекты следует разрабатывать. Так, если организация разрабатывает бухгалтерское программное обеспечение, то команда не может решить, что ей вместо этого хотелось быть разрабатывать программное обеспечение для управления светофорами. Такие решения практически всегда принимают руководители. А для этого им приходится определять границы песочниц, в пределах которых командам разрешается самоорганизовываться.

Если, например, команды строят замки на песке, то именно руководители решают, сколько таких замков построить, а также определяют границы каждой песочницы, в которой команда может самоорганизоваться и построить свой замок на песке. А если обратиться к более практическому примеру, то руководители организации, разрабатывающей бухгалтерское программное обеспечение, могут решать, какие именно бухгалтерские приложения разрабатывать, а также определять границы, в пределах которых команды разработчиков должны передавать свою работу командам развертывания или же выполнять развертывание сами в каждом спринте.

Постановка ясной возвышенной цели

Руководители ставят также ясные возвышенные цели для каждой команды, определяющие назначение и направление деятельности команды. Если снова обратиться к аналогии песочницы, то руководители могут решить, что им требуется построить на песке замок, который должен оказаться лучшим в соревновании песчаных замков в конце недели, а владелец продукта, работающий в Scrum-команде, может дополнительно уточнить цель "создать средневековый замок с башнями и окружающим его рвом".

Составление команд

Как правило, команды сами не формируются (т.е. члены команд сами не подбирают свой состав). Команды составляются руководителями. Если еще раз обратиться к аналогии песочницы, то именно руководители, а не сами члены команд почти всегда решают, кому и в какой песочнице играть. Безусловно, члены команд могут и должны внести свой вклад в процесс формирования своих команд, потребовав, например, участия в конкретной команде или проведя собеседование с новыми кандидатами в члены уже существующей команды. Тем не менее в большинстве организаций окончательное решение принимают руководители, чтобы составы команд должным образом уравновешивали коммерческие потребности и ограничения.

В среде Scrum руководители функциональных подразделений, представляющие разные дисциплины или деятельные сообщества профессионалов, сотрудничают друг с другом в подборе членов для межфункциональных Scrum-команд (рис. 13.4).

Горизонтальными пунктирными линиями на рис. 13.4 обозначены функциональные области или деятельные сообщества, состоящие из людей со сходными специальными навыками (например, сообщества разработчиков, проектировщиков графических пользовательских интерфейсов (ГПИ), тестировщиков, администраторов баз данных). У каждой функциональной области имеется свой руководитель соответствующего функционального подразделения.

Руководители функциональных подразделений несут коллективную ответственность за выбор подходящих людей из каждой функциональной области для формирования Scrum-команд, которые расположены на рис. 13.4 по вертикали. Руководители стремятся составить команды таким образом, чтобы они были межфункционально неоднородными и достаточными, а их члены обладали дополняющими друг друга Т-образными навыками (см. главу 11).

Рис. 13.4. Руководители функциональных подразделений совместно составляют Scrum-команды

Изменение состава команд

Руководители обязаны также поменять состав команды, если они считают, что это улучшит общее состояние и производительность команды в частности и организации вообще. Допустим, что Фред неэффективно работает в команде разработчиков, и его отношение к работе отрицательно сказывается на способности команды выполнять свои обязанности. Как разрешить ситуацию с Фредом?

Прежде всего, коллеги Фреда должны обсудить с ним сложившуюся ситуацию, постаравшись помочь ему и команде. Если они не добьются успеха, то Scrum-мастер как наставник Scrum-команды должен поработать с Фредом, чтобы помочь ему стать более эффективным членом команды. Если и наставления Scrum-мастера не помогут, то ситуация с Фредом, скорее всего, выйдет за пределы Scrum-команды, дойдя до администратора ее ресурсов, т.е. человека, перед которым Фред отчитывается в данной организации. Ведь Scrum-мастер не имеет полномочий нанимать и увольнять людей.

На данном этапе администратор ресурсов, которому подчиняется Фред, должен разрешить вопросы производительности труда Фреда соответствующим образом и по-человечески — возможно, вместе с кем-нибудь из отдела кадров.

С этой целью администратор ресурсов, вероятнее всего, обратится к Scrum-мастеру и членам команды разработчиков, чтобы лучше понять сложившуюся ситуацию, а затем принять решение переместить Фреда из одной Scrum-команды в другую команду, которой он, возможно, больше подойдет. С другой стороны, он может внести Фреда в план повышения производительности труда (в его текущей или новой команде), и если Фред не повысит производительность своего труда по этому плану, то его, скорее всего, уволят.

Несмотря на то что руководители имеют право увольнять людей, а члены команд и Scrum-мастера такого права не имеют, члены команд все равно участвуют в данном процессе, способствуя формированию команды должным образом. Кроме того, руководители могут поменять состав команды с целью оптимизировать способность организации выпускать продукты из своего портфеля заказов. Например, несмотря на то, что нужно стремиться к долговечности команд, время от времени может возникать потребность переместить работника со специальными навыками из одной команды в другую команду, которой срочно требуется именно такой специалист. Но руководители должны производить такие перемены аккуратно, поскольку изменение состава команды может отрицательно сказаться на ее работе.

Наделение команд полномочиями

Чтобы самоорганизоваться, команды должны быть наделены соответствующими полномочиями, а для этого требуется разрешение и доверие руководителей. Один из основных способов наделения команд полномочиями состоит в том, чтобы руководители поручили им часть своих обязанностей, преследуя главную цель — разрешить командам самоорганизоваться и тем самым улучшить самоуправление. С одной стороны, команды не должны принимать административные решения (как пояснялось ранее, члены команды Фреда не могут его уволить за плохую работу.) А с другой стороны, команды могут быть наделены полномочиями осуществлять типичные административные виды деятельности.

Чтобы поручить команде какой-нибудь вид административной деятельности или право принимать решение, руководитель выбирает подходящий уровень полномочий и наделяет ими команду. Апелло [Appelo, Jurgen. 2011] определил семь уровней таких полномочий, каждый из которых перечислен в табл. 13.1 и сопровождается соответствующим примером. Эти уровни простираются от одного крайнего предела (*приказ*), где руководитель сам принимает решение и извещает о нем команду, до другого крайнего предела (*поручение*), где руководитель наделяет команду всеми полномочиями принимать решения.

Таблица 13.1. Уровни полномочий с примерами по Апелло

Уровень	Наименование	Описание	Пример
1	Приказ	Руководитель принимает решение и отдает приказ команде	Переезд в новое учрежденческое здание
2	Убеждение	Руководитель убеждает команду в правильности решения	Решение применять методику Scrum
3	Обращение	Руководитель обращается к команде за помощью, прежде чем принять решение	Отбор новых членов команды
4	Согласие	Руководитель и команда принимают решение вместе	Выбор логотипа для нового структурного подразделения
5	Совет	Руководитель дает совет, чтобы повлиять на решение, принимаемое командой	Выбор архитектуры или компонента
6	Запрос	Руководитель делает запрос после того, как команда приняла решение	Продолжительность спринта
7	Поручение	Руководитель полностью поручает команде принять решение	Рекомендации по программированию

Когда руководители поручают командам задачи, они должны быть уверены, что команды выполнят эти задачи как следует. А команды должны быть уверены, что их руководители не предпримут никаких действий, противоречащих полномочиям, которыми они наделены. Например, руководителям не следует наделять сначала команды полномочиями принимать решение, а затем передумать и принять решение самостоятельно.

Кроме того, руководители должны помогать членам команд доверять друг другу. С этой целью руководитель может определить подходящие границы для той среды, в которой действует команда. Это будет способствовать формированию взаимного доверия в команде, устанавливая пределы, до которых доверие должно простираться. Руководители должны также помогать членам команд уяснить важность для самоорганизующейся команды выполнять личные обязательства, поскольку в самой команде отсутствует руководитель, заставляющий ее членов сделать работу. И руководители должны укреплять мушкетерские отношения среди членов команд, чтобы быть уверенными, что все они обязуются работать вместе для достижения целей, намеченных командой.

Опека команд

Сформировав Scrum-команды, руководители должны опекать их. Но опека совсем не означает *руководство* командами. Напротив, руководители должны

стимулировать людей, уделять внимание развитию их компетентности, руководить функциональными областями и сохранять целостность команд.

Стимулирование людей

Постановка ясной возвышенной цели служит основанием для стимулирования членов команды. Стимулирование означает, что руководители должны постоянно искать способы мотивировать людей к внутренней потребности делать отличную работу. Всем нам хочется работать в интересной, творческой атмосфере, способствующей доставке ценности, и руководители несут ответственность за взращивание такой атмосферы. Посредством административного управления руководители могут оказывать положительное влияние на внутреннюю мотивацию и энергию членов команд.

Но руководители могут предпринять действия, приводящие к совершенно противоположному результату, истощая энергию членов команд и лишая их всякой мотивации. Например, руководители функциональных подразделений по традиции привыкли поручать работу на уровне отдельных задач людям на своих участках. В среде Scrum это лишает людей энергии, подрывая основы самоорганизации команд и компрометируя их способность доставлять ценность.

Развитие компетентности

В организациях, практикующих методику Scrum, каждый член команды по-прежнему подчиняется руководителю функционального подразделения или администратору ресурсов, который, как правило, не является ни Scrum-мастером, ни владельцем продукта. И как и в средах без Scrum, эти руководители играют активную роль в наставничестве и содействии их непосредственным подчиненным в повышении по службе, предоставляя им возможность развивать свою компетентность и часто реагируя конструктивно на их трудовые успехи соответствующими практическими мерами.

Такие меры, как предоставление членам команд времени для обучения или посещения конференций, говорят за себя больше, чем любые слова. В среде, где поощряется постоянное обучение, руководители наставляют членов команд развивать свою компетентность в их предметной области, расширять технические знания, мыслительные способности и прочие полезные навыки.

Руководители должны часто и конструктивно реагировать на эффективность работы команд и отдельных работников. Во многих организациях, не практикующих методику Scrum, подобная реакция на эффективность работы приобретает форму ежегодной аттестации работников. А в организациях, практикующих методику Scrum и подобные аттестации, руководители функциональных подразделений, скорее всего, будут продолжать данный процесс. Но в тех организациях, где

усвоены ценности и принципы Scrum, вскоре осознают, что реагирование на эффективность работы отдельных работников один или даже два раза в год не согласуется с размеренным темпом работы Scrum-команд, которые исполняют свои обязанности и обучаются в течение краткосрочных спринтов.

Подобные ежегодные аттестации работников могут также способствовать соперничеству из-за слабого взаимного доверия в команде вместо самоорганизации с мушкетерскими отношениями. А критерии оценки эффективности труда отдельных работников могут только мешать высокой производительности всей команды, поощряя независимое поведение, когда люди склонны оптимизировать оценки их личного труда за счет команды. В организациях, успешно применяющих методику Scrum, быстро ставят под сомнение целесообразность проведения ежегодных аттестаций работников, осознавая, что они могут принести больше вреда, чем пользы.

Это совсем не означает, что эффективность труда отдельных работников не стоит оценивать в организациях, практикующих методику Scrum. Руководители должны просто привести частоту своей реакции на эффективность труда отдельных работников в соответствие с петлями обучения тех команд, членами которых являются их непосредственные подчиненные. В частности, руководители могут реагировать на каждый спринт. Кроме того, индивидуальное реагирование может быть поставлено в зависимость от того, насколько эффективность труда отдельных работников способствует (или не способствует) эффективности работы команды в целом.

Руководство функциональными областями

Как и в организациях, не практикующих методику Scrum, руководители структурных подразделений в тех организациях, где практикуется методика Scrum, продолжают руководить отдельными функциональными областями. Как правило, руководители структурных подразделений обладают достаточными практическими знаниями в своих функциональных областях и способны обеспечить идейное руководство ими. Такое руководство совсем не означает, что руководитель структурного подразделения поручает задачи своим непосредственным подчиненным или указывает им, как решать эти задачи. Подобные действия способны нанести ущерб самоорганизации команд. Напротив, такой тип руководства удовлетворяет насущную потребность в согласованности и логичности действий, а также наставничестве в функциональной области.

Например, в организациях, занимающихся разработкой компьютерных игр, художники подчиняются художественному руководителю, который сам является опытным художником. Художественный руководитель осуществляет общее руководство художниками, помогая им установить художественные нормы для игры

и просматривая их работы, чтобы обеспечить принципиальную целостность последних. Вряд ли для этой цели в Scrum-команде потребуется один художник, специализирующийся на готическом искусстве, а другой — на карикатурах. Осуществляя общее руководство, художественный руководитель помогает достижению художниками согласованных результатов, имеющих большую ценность.

Руководители структурных подразделений обеспечивают общее руководство, устанавливая нормы и поощряя инициативы, отвечающие характеру отдельных функциональных областей. Допустим, что руководителю подразделения контроля качества требуется выбрать новые инструментальные средства для автоматизации тестирования, чтобы пользоваться ими во многих проектных работах. С этой целью он может попросить специалистов по контролю качества, которые подчинены ему непосредственно, но в то же время являются членами разных Scrum-команд, совместно выбрать подходящие инструментальные средства, как показано на рис. 13.4.

Сохранение целостности команд

Как упоминалось в главе 11, Scrum-команда является денежной единицей гибкой разработки. Команда заменяет собой отдельных работников как единицу производительности, и поэтому руководители должны принимать упреждающие меры для сохранения целостности команд. Это означает, что они не должны отвлекать людей из команд посредине спринта ради работы в своих излюбленных проектах или поручать людям без особой необходимости работать одновременно в нескольких командах.

В силу привлекательности экономических показателей долгосрочных групп руководители должны также стараться сохранять их состав настолько долго, насколько это экономически оправдано. По завершении проектных работ руководители должны постараться поручить всей команде следующие проектные работы. И сделать это они должны еще до того, как попытаются разрушить столь ценный производственный актив, разделив его на части и разорвав ценные и полезные связи в команде.

Согласование и адаптация производственной среды

Внедрить методику Scrum в отдельной команде или даже целом отделе информационных технологий или разработки совсем неплохо, но недостаточно. Нужно еще охватить процессом гибкой разработки всю цепочку создания ценности: от исполнителей до заказчиков, чтобы получить особые выгоды от Scrum. Руководители отвечают за согласование и адаптацию производственной среды (т.е. цепочки создания ценности), продвигая ценности гибкой разработки, устраняя организационные препятствия, согласуя работу внутренних групп и работу с партнерами.

Продвижение ценностей гибкой разработки

Руководители должны воспринять ценности и принципы гибкой разработки. Они должны их понять и поверить в них, жить ими и поощрять других делать то же самое. Когда я веду курсы или инструктирую Scrum-команды, то зачастую слышу нечто подобное: “Все это, конечно, имеет смысл для нас, но нам нужно еще убедить свое начальство, иначе мы не сможем внедрить методику Scrum. Хотелось бы, что они были здесь и слышали все это”. И эти команды совершенно правы. Ведь им потребуется поддержка руководства, если они собираются успешно работать долгое время.

Однажды в обеденный перерыв мне пришлось вступить в дискуссию с руководством организации, которая только собиралась внедрять методику Scrum. Во время этой дискуссии я заметил, что руководители не должны отвлекать членов команды, работающей над одним проектом, к работе над каким-нибудь другим проектом, поскольку это нарушает нормальный ход работ. На это замечание один из присутствовавших руководителей робко, но искренне отреагировал так: “Да, но я делаю это постоянно и даже не думал, что это так плохо. Что еще мне нужно знать как руководителю грядущей гибкой разработки в нашей организации, чтобы лучше согласовать свое поведение и среду для содействия гибкости в разработке?”

В ответ на его вопрос я начал пояснять основные ценности и принципы гибкой разработки аналогично тому, как это было сделано в главе 3, чтобы он и его коллеги осознали, каким образом руководитель может способствовать укреплению этих ценностей и принципов вместо того, чтобы бессознательно действовать вопреки им. Разумеется, только ежедневными усилиями согласовать свое поведение руководители могут действительно продвигать ценности и принципы гибкой разработки в своей среде.

Устранение организационных препятствий

Руководители должны также тесно сотрудничать с Scrum-мастерами над устранением организационных препятствий. И хотя Scrum-мастер является инициатором в устранении организационных препятствий, многие из этих препятствий, особенно обусловленные внешними условиями, требуют вмешательства со стороны руководителей, чтобы устранить их.

Согласование работы во внутренних группах

Методика Scrum внедряется, прежде всего, в технических или информационно-технологических группах. Допустим, что через некоторый достаточный период времени группа, первой внедрившая методику Scrum, приобрела немалые навыки в создании ценных для потребителя функциональных средств в каждом

спринте. Но до тех пор, пока эти средства не станут доступными потребителям, говорить о доставке настоящей ценности не приходится. Что, если группа разработчиков действует не гибко и если она сдает функциональные средства в эксплуатацию через каждые несколько недель, хотя это совсем не то, на что она действительно способна или желает делать? Можно ли требовать от организации высокой производительности по методике Scrum, если она не в состоянии своевременно передавать ценность в руки заказчиков?

Что, если такое же рассогласование происходит по восходящей в процессе разработки? Возможно, отделы сбыта и маркетинга действуют по разным принципам. А что, если их отношение таково, что им все равно, каким процессом пользуются разработчики в своей работе, — главное, чтобы они вовремя укладывались в сроки доставки продукции заказчикам. Вполне возможно, что и сотрудники отдела кадров по-прежнему нанимают людей, руководствуясь старыми должностными инструкциями, а не отыскивая людей, имеющих T-образные навыки и желание работать в самоорганизующихся командах.

В таких средах трудно понять до конца преимущества методики Scrum в долгосрочной перспективе. Руководство, в том числе и высшее, обязано сформировать производственную среду таким образом, чтобы добиться хорошей внутренней согласованности в работе различных групп, включая администрацию, финансы, сбыт, маркетинг, разработку и техническую поддержку. Руководители должны видеть всю производственную среду в целом и согласовывать ее полностью с принципами гибкой разработки.

Согласование работы с партнерами

А стоит ли останавливаться на внутренней согласованности? Руководители организации должны помогать в распространении гибкого подхода и на административное управление работой с поставщиками и субподрядчиками. Если привлечение внешних партнеров осуществляется традиционным официальным способом, опирающимся на заключение контрактов и ведение переговоров, то организация вряд ли сможет реализовать весь свой потенциал, применяя методику Scrum.

Напротив, руководители должны способствовать применению гибких принципов в привлечении партнеров. Например, простейшей формой соглашения о субподрядных работах служит аренда Scrum-команды у сторонней организации. Вместо того чтобы выполнять всю нелегкую работу по созданию высокопроизводительной команды, руководители приобретают доступ к высокопроизводительной группе, уже сформированной другими. На данном этапе организация применяет методику Scrum, как описано в данной книге, но такая команда разработчиков, а возможно, и Scrum-мастер, фактически принадлежит сторонней, а не данной организации.

Чтобы достичь такого уровня согласованности работы с партнерами, руководители должны рассматривать альтернативы составлению с субподрядчиками контрактов, имеющих фиксированную стоимость. Такие контракты сразу же портят отношения между сторонами контракта, поскольку подрядчик стремится доставить как можно меньшую ценность, чтобы удовлетворить условия контракта и в то же время получить максимальную валовую прибыль, а подряжающей организации требуется получить как можно больше по фиксированной стоимости контракта. Едва ли такой порядок ведения дел можно считать гибким. Руководители должны изменить подобный способ привлечения субподрядчиков.

Управление ходом работ по созданию ценности

В целом руководители организаций, применяющих методику Scrum в своей среде, отвечают на выбор стратегического направления и размещение организационных ресурсов таким образом, чтобы достичь стратегических целей экономически обоснованным способом. Руководители осуществляют административное управление ходом работ по созданию ценности, применяя системный подход, выполняя экономический анализ и контролируя количественные показатели и отчеты.

Системный подход

Для эффективного управления ходом работ по созданию ценности руководители должны применять системный подход. Одно из главных препятствий к успешному внедрению методики Scrum возникает в тех случаях, когда руководители отказываются мыслить системно и вместо этого сосредоточивают основное внимание на своих узких участках работы. Мне часто приходится слышать следующее: “Да, но для того чтобы сделать то, что вы предлагаете, придется изменить структуру подчиненности или основные должностные инструкции”. Люди, говорящие это, обычно добавляют следующее: “Мне трудно себе представить, что мы вообще сможем это сделать, и поэтому я не могу [или не хочу] ничего менять на своем участке работы, чтобы то, что мы делаем, лучше соответствовало ценностям и принципам Scrum, а также остальной гибкой организации”.

При таком ограниченном местными интересами мышлении нелегко добиться сколько-нибудь ощутимого внутреннего согласования с гибкой методикой, а разные подразделения организации могут просто работать во вред общей пользе системы. Чтобы реализовать преимущества высокой производительности по методике Scrum в долгосрочной перспективе, руководители организации, применяющей эту методику, должны стремиться видеть общую перспективу.

Экономический анализ

Как правило, руководителями организаций являются администраторы, которым доверяют имеющиеся финансовые ресурсы. Следовательно, руководители высшего звена в организации, практикующей Scrum, должны выполнять экономический анализ (например, прибылей и убытков) на своих участках работы. Руководители структурных подразделений или администраторы ресурсов могут и не нести непосредственную ответственность за прибыль, но они все равно отчитываются за расходование вверенных им финансовых ресурсов.

Руководители (возможно, высшего звена) должны также следить за экономическими показателями на высшем уровне организации. С этой целью они зачастую управляют портфелем заказов и организацией в целом. Управляя портфелем заказов, они определяют финансирование конкретных проектных работ и порядок их выполнения. А по ходу проектных работ руководители анализируют и реагируют на непрерывный поток ответной реакции, поступающей в реальном времени, исходя из результатов итеративной и инкрементной разработки, и если потребуется, то они могут прекратить те проектные работы, экономические показатели которых больше не оправдывают дополнительные расходы (подробнее об этом — в главе 16).

Контроль показателей и отчетов

Многие показатели и отчеты накапливаются и составляются по запросу руководителей, и поэтому у них имеется настоящая возможность убедиться, что фиксируются и сообщаются только те показатели, которые способствуют потоку создания ценности. Чтобы достичь этой цели, необходимо добиться того, чтобы показатели и отчеты вполне соответствовали основным ценностям и принципам Scrum.

В главе 3 были описаны некоторые принципы Scrum, по которым руководители могут организовать определение показателей и составление отчетов. Ниже приведены характерные тому примеры.

- Акцент на простаивающей работе, а не на простаивающих работниках. С этой целью определяется время и частота прерывания нормального хода работ, а не степень занятости работников. Такой показатель, как время цикла, позволяет выяснить продолжительность промежутка времени от начала и до конца работ. Если время цикла увеличивается, необходимо исследовать причины, по которым это происходит.
- Определение прогресса путем проверки достоверности рабочих ресурсов. Действительно ли так важно выпустить вовремя и в рамках бюджета продукт, который никому не нужен? Акцент следует сделать на определении доставляемой ценности (рабочих и достоверных ресурсов), но не теряя из

поля зрения показатели (сроки, объем работ, бюджет и качество), которые требуются для доставки ценности.

- Организация рабочего процесса для получения быстрой ответной реакции. Показатели должны быть согласованы таким образом, чтобы можно было выяснить, насколько быстро завершится цикл обучения (предположение, построение, ответная реакция, обследование и адаптация).

Последний показатель положен в основу *учета нововведений*, который оказывается эффективным в тех организациях, где продукт или служба создается в условиях крайней неопределенности [Ries, Eric. 2011]. Для учета нововведений применяются действенные количественные показатели, чтобы оценить быстроту обучения как важную меру прогресса в достижении коммерчески ценного результата. Учет нововведений производится в следующие три этапа.

1. Создание минимально жизнеспособного продукта (MVP) с целью установить конкретные исходные значения действенных количественных показателей текущего состояния организации и разрабатываемого продукта.
2. Осуществление ряда постепенных усовершенствований продукта с целью улучшить исходные значения действенных количественных показателей в сторону идеальных или желательных значений.
3. Если действенные количественные показатели свидетельствуют, что разработка продукта демонстрирует прогресс в сторону конечной цели, то нужно и далее упорно следовать по текущему пути к ней, а иначе происходит резкая смена стратегии и процесс начинается снова.

Руководители проектов

До сих пор была рассмотрена роль руководителя функционального подразделения или администратора ресурсов. А какова роль руководителя проекта? Имеется ли такая роль в организации, практикующей методику Scrum?

Обязанности руководителя проекта в Scrum-команде

Существует типичное заблуждение, что Scrum-мастер является руководителем проекта гибкой разработки, только его роль называется иначе. На первый взгляд, у ролей Scrum-мастера и руководителя проекта имеются некоторые сходства, например, оба должны устранять препятствия, мешающие нормальному ходу проектных работ. Но роль лидера-служителя существенно отличается от административно-управленческой роли руководителя проекта.

Чтобы ответить на вопрос о роли руководителя проекта, рассмотрим основные обязанности руководства проектом, определяемые Институтом руководства проектами (Project Management Institute; [PMI. 2008]). Они сведены в табл. 13.2.

Таблица 13.2. Традиционные обязанности руководства проектом

Вид деятельности по руководству проектом	Описание
Интеграция	Выявление, определение, сочетание, объединение и координация различных процессов и видов деятельности по руководству проектом
Объем работ	Выбор того, что следует включать в проект, а также контроль над тем, чтобы в проект были включены все требующиеся работы
Время	Своевременное руководство завершением проекта, которое состоит в том, чтобы определить, что и когда нужно делать и какие для этого потребуются ресурсы
Стоимость	Оценка, составление бюджета и управление затратами в рамках утвержденного бюджета
Качество	Составление требований и/или норм качества, контроль качества, проверка и запись результатов деятельности, направленной на достижение качества
Команда (кадры)	Организация и руководство проектной командой
Обмен информацией	Формирование, накопление, распространение, хранение, извлечение и уничтожение проектной информации
Риск	Планирование, выявление, анализ, реагирование, контроль и управление проектными рисками
Снабжение	Приобретение продуктов и услуг или получение извне результатов, требующихся для проектной команды

Безусловно, эти обязанности остаются важными и нужными. Ведь если нет руководителя проекта, то кто будет следить за всеми этими видами деятельности? Как следует из табл. 13.3, традиционные обязанности руководителя проекта распределяются среди ролей в Scrum-команде, а возможно, и среди других руководителей.

Как следует из табл. 13.3, человек, который раньше был руководителем проекта, теперь может выполнять любую из ролей в Scrum-команде в зависимости от его профессиональных навыков и желания. Из многих руководителей проектов получаются отличные Scrum-мастера, если они способны оставить свои административно-управленческие привычки.

Но, как следует из той же табл. 13.3, владельцу продукта приходится выполнять не меньше обязанностей, чем Scrum-мастеру. Таким образом, традиционные руководители проектов могут вполне справиться с ролью владельца продукта, если они обладают достаточными знаниями предметной области и прочими навыками для выполнения этой роли. И намного реже руководители проектов с технической подготовкой решают стать членами команды разработчиков.

Таблица 13.3. Распределение обязанностей руководителя проекта в организации, практикующей методику Scrum

Вид деятельности по руководству проектом	Владелец продукта	Scrum-мастер	Команда разработчиков	Другой руководитель
Интеграция	✓			✓
Объем работ	Макро-уровень		Уровень спринта	
Время	Макро-уровень	Помогает Scrum-команде эффективно использовать время	Уровень спринта	
Стоимость	✓		Оценивание историй или задач	
Качество	✓	✓	✓	✓
Команда (кадры)			✓	Формирование команды
Обмен информацией	✓	✓	✓	✓
Риск	✓	✓	✓	✓
Снабжение	✓			✓

Сохранение отдельной роли руководителя проекта

На первый взгляд, руководители проектов могут стать Scrum-мастерами, владельцами продуктов или членами команды разработчиков. Но это не всегда именно так. В организациях, ведущих крупные и сложные проектные работы, иногда решают сохранить отдельную роль руководителя проекта, когда логика и координирование задач оказываются настолько сложными, что командам трудно самим справиться с ними.

Как правило, я стремлюсь к тому, чтобы Scrum-команды сами справлялись с логикой и координированием проектных работ. Ведь Scrum-команды не должны ожидать, что кто-нибудь со стороны возьмет на себя ответственность за координирование работ от их имени. Такое ожидание заставляет членов команды думать, что если кто-нибудь другой отвечает за координирование работ, то они за это уже не отвечают.

Логика и зависимости в мелких проектных работах, которые выполняются лишь несколькими Scrum-командами, легко координируются между командами ежедневно с помощью таких методов, как схватка над схватками (см. главу 12). Но что, если проектные работы выполняются десятками, а то и сотнями Scrum-команд или даже тысячами разработчиков?

Аналогично упоминавшемуся в главе 6 правилу единственного задела на каждый продукт, правило самостоятельного координирования командами своих работ может стать верной отправной точкой в данном случае. Но как и вопрос масштаба может заставить ослабить правило единственного задела на каждый продукт, так и для координации хода работ, возможно, придется оставить одного или нескольких руководителей проектов или программ.

Прежде чем пойти по пути сохранения особой координирующей роли руководителя проекта только из-за большого числа команд, нужно отступить назад и бросить взгляд на общую картину каналов связи между командами. Как показывает мой опыт, в подобных ситуациях каналы очень редко обеспечивают полную связь между командами (рис. 13.5).

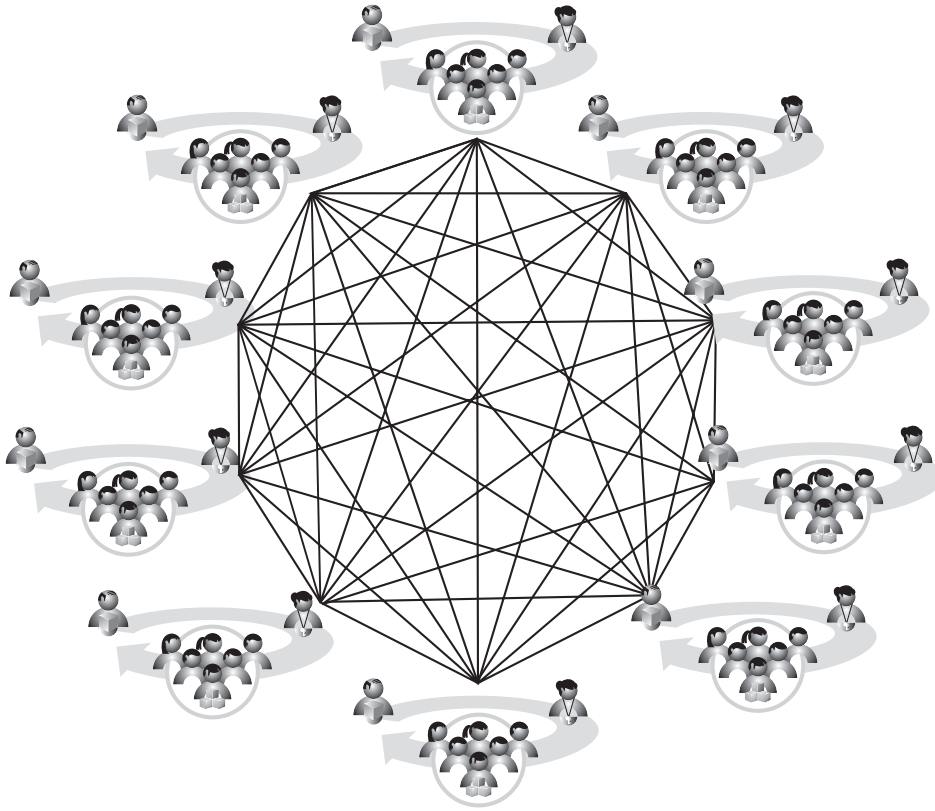


Рис. 13.5. Каналы очень редко обеспечивают полную связь между командами

Команды могут (или должны) группироваться вместе по участкам функциональных средств или их эквивалентам, где каналы связи более распространены в отдельной группе и менее тесно связаны через группы (рис. 13.6).

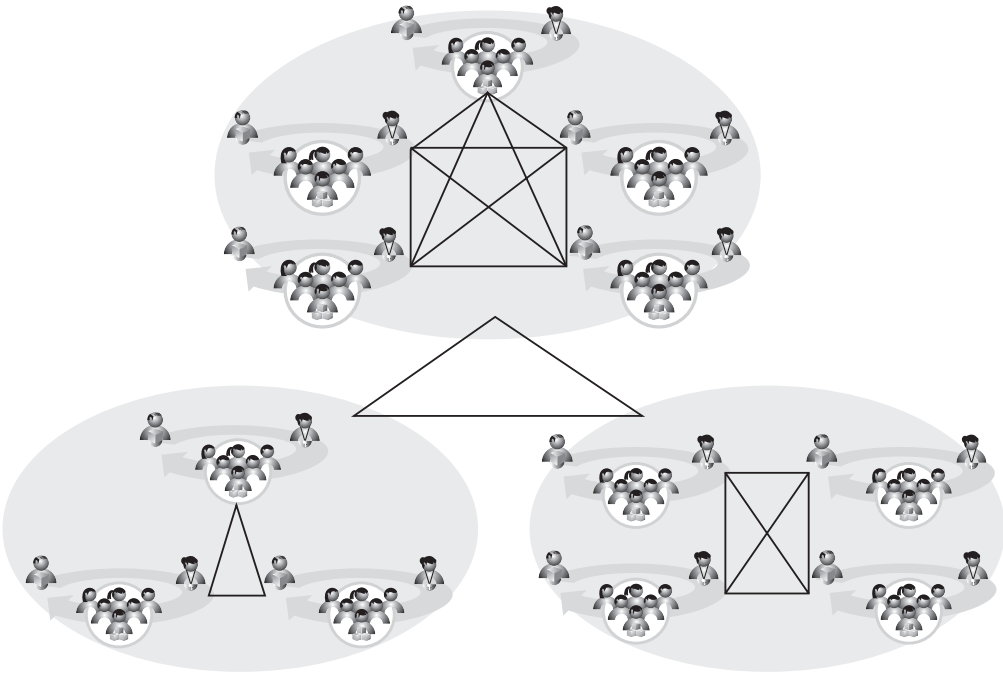


Рис. 13.6. Команды нередко объединяются в группы для более тесного сотрудничества

В подобных случаях Scrum-команды могут легко координировать работу между собой. Но кому принадлежит координирование работы между группами команд? В качестве стандартного ответа на этот вопрос можно сказать, что команды должны делать это сами. И, как правило, такой подход оказывается вполне работоспособным. Координирование можно проводить аналогично схватке над схватками, где представитель каждой группы команд встречается с представителями других групп для обсуждения вопросов координирования зависимостей.

Но при наличии самых разных групп команд даже такой вид координирования, как схватка над схватками, может оказаться затруднительным. В подобных случаях мне приходилось наблюдать, как организации поручали обязанности координировать ход работ руководителю проекта или программы (рис. 13.7).

Я лично предпочитаю не ставить руководителя проекта в центр координирования работ. Такому методу присущ риск, связанный с тем, что отдельные Scrum-команды могут передавать ответственность за координирование третьей стороне.

Но при достаточно больших масштабах работ я все же признаю, что если поручить одному или двум людям следить все время за логикой и координированием хода работ, то можно избежать простоев (падения эстафетной палочки на землю). Чтобы стало понятнее, почему отдельные команды не могут поручить обязанности по координированию работ с другими командами кому-то другому,

я предлагаю рассматривать роль руководителя проекта как помощника Scrum-команд (аналогично лидеру-служителю). В этой роли руководитель проекта должен мыслить системно и работать прилежно с каждой группой команд или отдельными командами, чтобы все они ясно понимали, какая именно требуется координация их действий, хотя сама координация должна принадлежать командам.

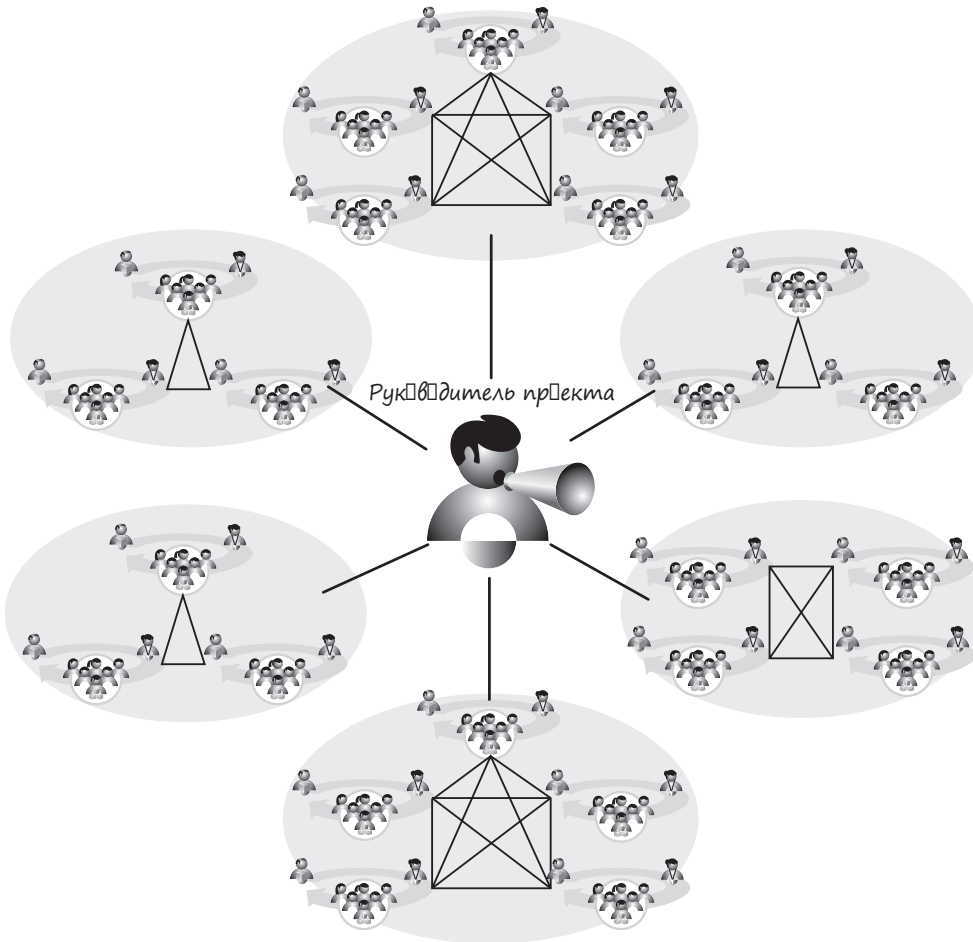


Рис. 13.7. Поручение обязанности координировать ход работ руководителю проекта или программы

Аналогичным образом роль руководителя проекта может оказаться полезной и в тех проектных работах, где применение методики Scrum является лишь небольшой частью более крупной разработки продуктов или служб. Например, в проектных работах могут принимать участие субподрядчики, внутренние команды, не практикующие методику Scrum, а также другие внутренние организации, связанные с выпуском продукции. В частности, логика сотрудничества с субподрядчиками или

поставщиками может оказаться довольно сложной и отнимающей немало времени. При наличии стольких участников в проектных работах полезно иметь человека, уделяющего основное внимание логике ведения проектных работ (рис. 13.8).

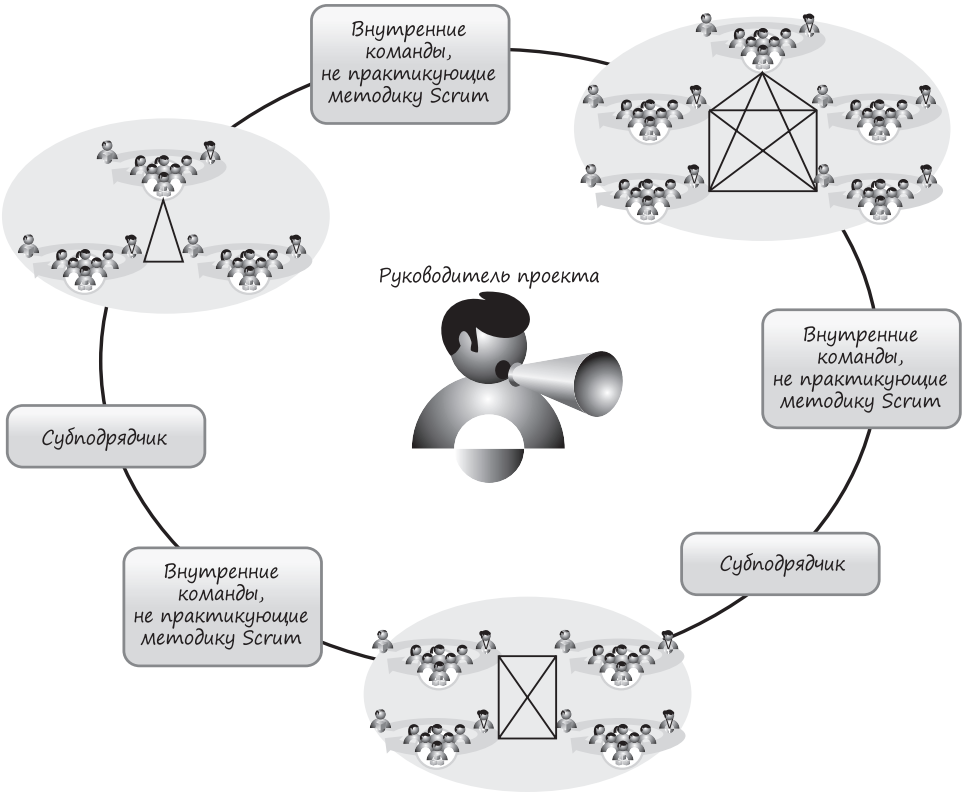


Рис. 13.8. Роль руководителя проекта в сложных проектных работах со многими участниками

И в этом случае цель состоит не в назначении человека специально на должность руководителя проекта, а в поручении ему заботиться о разрешении зависимостей между различными участками таким образом, чтобы одни команды хорошо понимали их и наиболее эффективно координировали свою работу с другими командами.

Заключение

В этой главе описана роль руководителей функциональных подразделений в организации, практикующих методику Scrum. Обязанности руководителей разделены на категории формирования команд, опеки команд, согласования и адаптации среды и управления ходом работ по созданию ценности. В табл. 13.4 сведены

для сравнения обязанности руководителей функциональных подразделений в традиционной организации и в организации, практикующей методику Scrum.

Таблица 13.4. Сравнение обязанностей руководителей функциональных подразделений в традиционной и Scrum-среде

Традиционная среда	Scrum-среда
Назначение людей для проектов	Эффективные команды формируются коллективно
Наем и увольнение	То же самое
Забота о развитии профессиональных навыков подчиненных	То же самое
Анализ производительности	По-прежнему требуется, но ответная реакция возникает значительно чаще и должна быть привязана к команде
Поручение (иногда) задач членам команды	Разрешение членам команды самоорганизовываться, определять и выбирать себе задачи
Установление межпроектных норм в функциональной области	То же самое
Поощрение инициатив в конкретной функциональной области	То же самое
Достаточные практические знания функциональной области и оказание помощи по мере надобности	То же самое
Умение и опыт перемещать непосредственных подчиненных из одной команды в другую	Забота о сохранении целостности команды
Устранение препятствий	То же самое
Забота о своей функциональной области	Умение видеть общую перспективу для согласования и создания ценности
Экономический анализ (прибылей и убытков)	То же самое
Контроль показателей и отчетов	Согласование показателей и отчетов с принципами гибкой разработки, чтобы уделить основное внимание непрерывности потока создания ценности

Несмотря на то что бо́льшая часть этой главы была посвящена роли руководителя функционального подразделения, в ее конце обсуждалась также роль руководителя проекта. Основное внимание при этом было уделено распределению традиционных обязанностей этой роли среди трех ролей в Scrum-команде, а также пользе от наличия одного или нескольких руководителей проектов в дополнение к этим трем ролям в организациях, ведущих сложные проектные работы.

Этой главой завершается часть II данной книги. А со следующей главы начинается обсуждение вопросов планирования. С этой целью в ней описываются важные принципы планирования в Scrum.

Часть III

Планирование

Глава 14
Принципы планирования в Scrum

Существует давний миф о том, что разработка по методике Scrum начинается без всякого планирования. Достаточно начать первый спринт, а подробности станут ясны по ходу дела. В действительности дело обстоит совсем иначе. И в Scrum требуется настоящее планирование. На самом деле планирование происходит на нескольких уровнях детализации и во многие моменты времени. Некоторым может показаться, что в Scrum планированию придается меньшее значение, поскольку бо́льшая часть планирования происходит оперативно, а не заблаговременно. Но, как показывает мой опыт, команды нередко тратят больше времени на планирование разработки по методике Scrum, чем по традиционной методике, хотя это и происходит совсем иначе.

В этой главе дается расширенное изложение нескольких принципов Scrum, описанных в главе 3, с акцентом на их применение в планировании. Этим закладывается прочное основание для обсуждения в главе 15 нескольких уровней, на которых происходит планирование в Scrum. А в последующих главах более подробно рассматриваются вопросы планирования портфеля заказов, продукта, выпуска и спринта.

Краткий обзор

Основные принципы были описаны в главе 3. От их количества зависит выбор подхода к планированию по методике Scrum. В этой главе подробно рассматриваются принципы планирования, приведенные на рис. 14.1. Другие принципы Scrum, в том числе работа в течение кратких промежутков времени, эффективное использование размеренного темпа и прочие, будут рассмотрены в последующих главах, посвященных планированию.

Рис. 14.1. Принципы планирования в Scrum

Не следует допускать, что планы можно составить заранее

Традиционный прогнозируемый подход к планированию состоит в том, чтобы составить подробный план заранее, прежде чем начнутся проектные работы. Цель такого планирования — действовать правильно, чтобы остальную часть работы можно было продолжить по плану. Некоторые считают, что без такого плана нельзя заранее знать, куда двигаться дальше, и скоординировать работу людей, особенно в крупных плановых работах с участием многих команд. И в этом аргументе есть своя истина.

Подход к планированию в Scrum имеет подлинно эмпирические корни в обследовании и адаптации. Выполняя разработку по методике Scrum, мы не можем спрогнозировать ее заранее, и поэтому даже не пытаемся планировать *все* артефакты заранее. Тем не менее мы планируем *некоторые* артефакты на ранней стадии, чтобы достичь золотой середины между предварительным и оперативным планированием.

Предварительное планирование должно быть полезным, но не чрезмерным

Рассмотрим пример, иллюстрирующий следующий принцип: *предварительное планирование должно быть полезным, но не чрезмерным*. Я живу в штате

Колорадо — отличном месте для катания на горных лыжах. Иногда я отдыхаю, катаясь на лыжах, хотя и не обладаю достаточным опытом, тогда как мой друг Джон является искусным горнолыжником. Откровенно говоря, я порой мечтаю стать искусным горнолыжником, но мне не хватает такого же умения и фанатизма, как у Джона. Однажды Джон показывал фотографии своих приключений на особенно сногсшибательной горе. Ради любопытства я задал ему следующий простой вопрос: "Когда ты находишься на вершине горы, готовясь к спуску вниз, планируешь ли ты весь маршрут своего спуска?"

Ухмыльнувшись, он ответил: "Нет. Если бы я это делал, просто разбился бы на смерть". И продолжил: "Я выбираю место на некотором расстоянии спуска с горы. Моя первая цель — доехать на лыжах до этого места. Может быть, я и планирую таким образом первых два или три поворота, но, по правде говоря, дальнейшее планирование было бы просто невозможным и опасным".

Я спросил: "Почему?"

Он пояснил: "Горная местность оказывается не такой, как ее видишь с вершины, из-за освещения и прочих факторов, обманывающих зрение. Кроме того, где-то на пути могут попадаться деревья, но с вершины мне их не видно. Если, стоя на вершине, я решу повернуть в каком-то месте вправо и поступлю именно так, то могу налететь прямо на эти деревья. Помимо этого, трудно предугадать, что какой-нибудь юный сноубордист не вылетит прямо над моей головой, крича: "Берегись, старик!" Спускаясь вниз, никогда не знаешь, когда и почему придется сменить курс".

После этого пояснения я, помнится, подумал: "Вот здорово. Это очень похоже на интересные проектные работы, над которыми я работал". Трудно предсказать заранее с достаточной определенностью, когда именно и почему придется изменить курс. Когда меня просили заранее составить подробный план разработки продуктов, я его составлял. Но я не припомню ни одного случая, когда такой метод оказывался бы работоспособным. Я вообще не помню, чтобы по завершении работ мы, возвращаясь к первоначальному плану, восклицали: "Составлен идеально!" В каком-то отношении попытку чрезмерного планирования можно сравнить с попыткой спланировать каждый поворот, стоя на вершине горы. Планирование на таком уровне детализации является расточительством. Надеяться на то, что план окажется правильным до такой степени, что можно пренебречь данными реального времени, было бы опасно.

Большинству из нас приходилось принимать участие в разработке продуктов, где уровень детализации связан с предварительным прогнозирующим планированием. Означает ли это, что мы не должны выполнять предварительное планирование? Разумеется, нет. Ведь это было бы небрежностью и безрассудством. Безусловно, Джон выполняет некоторое предварительное планирование, изучая главные особенности горной местности, чтобы приобрести уверенность,

прежде чем начать спуск. Но в равной степени было бы безрассудно спланировать момент, когда реагировать на реальность трудно или дорого. Как и Джон, мы должны найти правильное равновесие между предварительным прогнозированием и оперативной адаптацией.

Варианты планирования остаются в силе до самого последнего момента

Чтобы достичь правильного равновесия между предварительным и оперативным планированием, мы руководствуемся следующим принципом: *варианты планирования остаются в силе до самого последнего момента*. Это означает, что мы сохраняем планирование, которое лучше всего выполняется оперативно именно в тот момент, когда мы получаем более качественную информацию. Зачем же принимать решение о предварительном планировании на основании некачественной информации? Помимо того, что преждевременное планирование очень дорого, оно еще и опасно, как заметил Джон.

Акцент на адаптации и перепланировании, а не на соответствии плану

В ходе многих проектных работ нередко оказывается, что слишком много внимания уделяется предварительному планированию и недостаточно внимания непрерывному планированию. Если потратить много времени на предварительную разработку сильно спрогнозированного плана, полагая, что он составлен правильно, то возникнет значительная тяга соблюдать этот план, а не обновлять его, реагируя на перемены. Если же, напротив, посчитать, что план нельзя составить заранее, поскольку нельзя исключить перемены, как это и принято в Scrum, то лучше почесть за благо своевременно реагировать на изменения и перепланировать, чем составлять предварительный план.

В 1980-е годы я помогал разрабатывать крупные планы или консультировал компании, разрабатывавшие такие планы, поэтому они хорошо мне известны. Их результатом обычно становился крупный календарный график Гантта, распечатывавшийся на нескольких страницах, которые затем склеивали вместе и вывешивали на стене (рис. 14.2).

В ходе нескольких подобных проектных работ мы тратили до шести месяцев на разработку подробно спрогнозированных предварительных планов. Как только такие планы были составлены, они становились настоящими картами для выполнения проектов. Как и в юридической системе, человек невиновен до тех

пор, пока его вина не доказана, подобные планы считались верными до тех пор, пока действительность не доказывала их ошибочность. И здесь уместно было бы упомянуть о следующем благоразумном совете, который приписывается швейцарской армии, но, скорее всего, почерпнут из руководства по выживанию для парашютно-десантных частей особого назначения (*SAS Survival Guide*; [Wiseman, John "Lofty." 2010]; рис. 14.3): "Если вы заблудитесь в лесу, а карта не соответствует местности, всегда полагайтесь на местность".

Ошибочная вера в такую карту при разработке любой продукции вынуждает нас делать вывод, что прогресс в работе можно и должно измерять как соответствие или отклонение от плана. Так, если происходит отклонение от плана, наше желание строго придерживаться плана заслоняет от нас тот факт, что сама карта может оказаться неверной. Если же карта оказывается для нас важнее местности, мы просто теряем ощущение реальности, в которой нам нужно каким-то образом перемещаться.

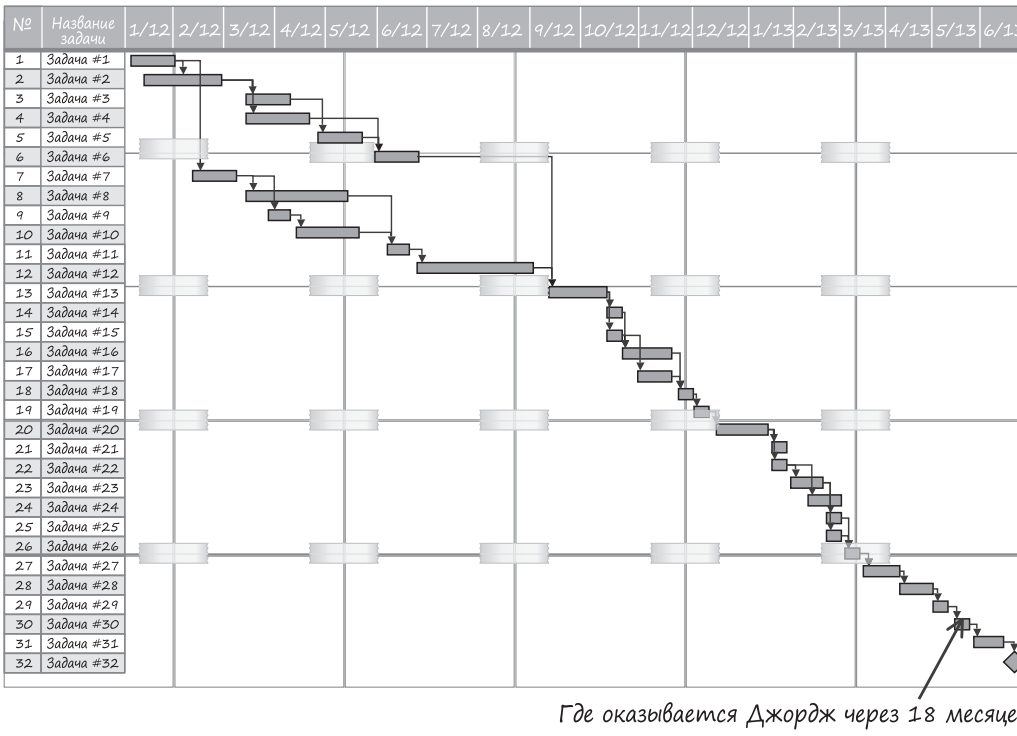


Рис. 14.2. Крупный календарный график Гантта, составленный по предварительному плану работ

Рис. 14.3. Если карта не соответствует местности, всегда полагайтесь на местность

Применяя методику Scrum, мы не отрицаем полезность предварительного плана, но считаем, что нужно не только уметь читать карту, но и ориентироваться на местности, приспосабливая к ней карту. И это вполне благоразумно, если учесть, что любой предварительный план составляется при наличии минимальных знаний о разрабатываемом продукте, а следовательно, предварительные планы очень точно отражают наше невежество на ранней стадии процесса разработки.

Применяя методику Scrum, мы нередко отдаем предпочтение перепланированию, проверяя свои предположения. При этом мы не особенно переживаем, если наши планы окажутся неверными, поскольку знаем, что нам вскоре придется заменить их более точными планами. Работая короткими спринтами в течение нескольких недель, но не больше месяца, мы не тратим слишком много времени на выбор неверного пути, прежде чем откорректировать свой курс.

Несмотря на то что большинство Scrum-команд, которые мне приходилось наблюдать, не применяют календарный график Гантта, они все же занимаются в той или иной степени формой долгосрочного планирования. В действительности, как обсуждается в главе 15, Scrum-команды занимаются планированием на нескольких уровнях детализации. Но при этом мы не должны особенно привязываться к своим планам, которые мы не желаем перепланировать, когда

происходят перемены или же когда мы узнаем важную информацию, на которую нам приходится реагировать.

Правильное управление запасами планирования

В главе 3 обсуждался один из главных принципов Scrum, предполагающий управление запасами, которые также называются *незавершенными работами*. Находя правильное равновесие между предварительным и оперативным планированием, очень важно понять, что создание крупных запасов прогнозируемых, еще не проверенных артефактов планирования является потенциально весьма убыточным занятием. А правильное управление запасами артефактов планирования экономически выгодно.

Если вернуться к примеру крупного календарного графика Гантта, составляемого заранее, то по мере развертывания проектных работ и проверки предположений на основании приобретенных знаний о разрабатываемой продукции можно узнать, где именно первоначальный план оказался ошибочным. К сожалению, к этому времени бремя убытков от необходимости возвращаться назад и переделывать перспективные планы таково, что полученные знания просто делают эти планы недействительными.

В итоге возникают убытки по меньшей мере в трех формах. Во-первых, это напрасные усилия, которые были затрачены на составление тех частей плана, которые теперь оказались отвергнутыми. Во-вторых, это потенциально значительные убытки от необходимости обновлять план. И в-третьих, это утраченная возможность потратить время на более полезные виды деятельности вроде выпуска очень ценного рабочего программного обеспечения вместо выполнения предварительных работ, которые затем придется все равно переделывать.

Я всегда стараюсь уравновешивать планирование работы, которую делаю в какой-то момент времени, с вероятностью того, что эта работа может в какой-то мере оказаться напрасной, если произойдут перемены. Например, на календарном графике Гантта, приведенном на рис. 14.2, имя Джорджа указано рядом с задачей, работу над которой требуется начать через 18 месяцев. Каковы шансы, что Джордж будет заниматься этой задачей через 18 месяцев? Они, вероятнее всего, близки к нулю!

Если такова вероятность, что план окажется настолько ошибочным в перспективе, то зачем вообще планировать насколько далеко? Обычно это делается для того, чтобы ответить на вопросы вроде следующих: “Когда мы это сделаем?” или “Сколько людей нам потребуется для этих проектных работ?” Как ответить на подобные вопросы с какой-то определенностью, если не прогнозировать всю работу?

На самом деле нужно найти ответы на следующие вопросы: когда будет выпущен продукт и какие функциональные средства можно внедрить в него

к предопределенной дате? Но нельзя обманывать себя, думая, что у нас имеется правильный ответ, только потому, что мы осуществили долгосрочное прогнозирование с малой степенью определенности. Эти вопросы планирования будут обсуждаться в следующей главе.

Предпочтение более мелких и частых выпусков

Предпочтение отдается более мелким и частым выпускам в Scrum потому, что они обеспечивают более скорую ответную реакцию и повышают окупаемость инвестиций в разрабатываемый продукт. Доходность продукта можно всегда увеличить в течение срока его службы, эффективно используя инкрементную разработку и частые выпуски с более мелкими подмножествами пригодных для продажи функциональных средств.

Проведем в качестве примера экономический анализ единожды выпускаемого продукта, как показано на рис. 14.4, взятом из [Denne, Mark, and Jane Cleland-Huang. 2003]. В начале разработки мы тратим средства без всякой окупаемости, начиная тем самым инвестиционный период. Выпуск продукта происходит в течение инвестиционного периода, как показано на спуске кривой, приведенной на рис. 14.4. В конечном итоге мы достигаем самофинансирования, когда доход от продукта сравнивается с затратами на его разработку. И как только доходы превысят затраты, мы входим в период окупаемости инвестиций. А когда совокупный доход сравнивается с общими расходами, то достигается точка безубыточности, начиная с которой наш проект наконец-то начинает приносить прибыль!

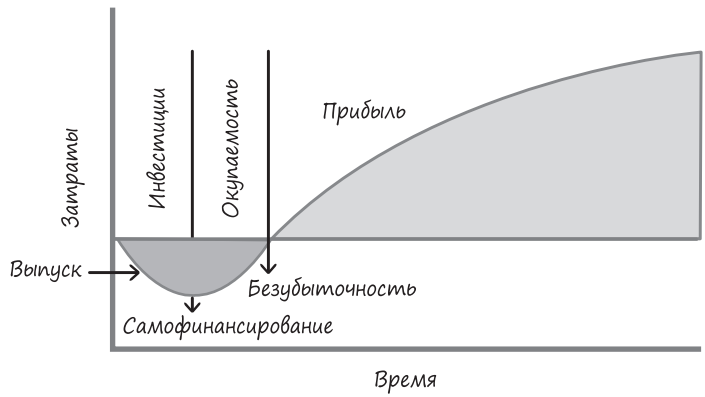


Рис. 14.4. Экономический анализ единичного выпуска

Чтобы продемонстрировать преимущества более мелких и частых выпусков, допустим, что продукт выпускается дважды, а не единожды (рис. 14.5). В этом случае мы достигаем самофинансирования, безубыточности и прибыльности

быстрее, а следовательно, и общей окупаемости инвестиций в разработку продукта. В качестве конкретного примера, адаптированного из [Patton, Jeff. 2008], рассмотрим модель повышения окупаемости инвестиций с предположениями, приведенными в табл. 14.1.

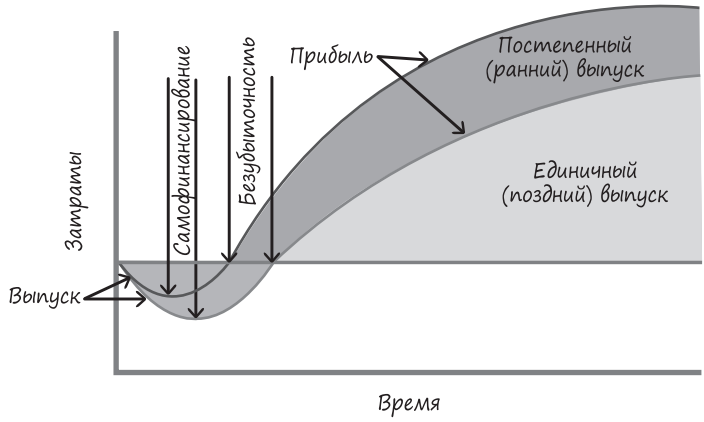


Рис. 14.5. Экономический анализ многократного выпуска

Таблица 14.1. Предположения в модели повышения окупаемости инвестиций

Показатель	Значение (в долларах США)
Доход (со всеми функциональными средствами)	300 тыс. в месяц
Доход (с половиной функциональных средств)	200 тыс. в месяц
Доход (с третью функциональных средств)	150 тыс. в месяц
Задержка между выпуском и получением дохода	1 месяц
Затраты на разработку	100 тыс. в месяц
Затраты на выпуск	100 тыс. на каждый выпуск

Результаты, приведенные в табл. 14.2, показывают, что при единственном выпуске окупаемость инвестиций через 12 месяцев составит 9,1 %. Если же делать два выпуска каждое полугодие, то окупаемость инвестиций повысится до 15,7%, а при ежеквартальных выпусках — до 19,5%.

Данному методу присущи некоторые ограничения. Прежде всего, у каждого продукта имеется минимальный набор выпускаемых и пригодных для продажи функциональных средств. Следовательно, продолжать дальше уменьшение первоначального выпуска нельзя, поскольку он в конечном итоге станет насколько мелким, что окажется непригодным для продажи. Кроме того, если рынок открыт для получения частичной ценности раньше, то доставка более мелких, но частых и пригодных для продажи выпусков может стать очень важным для наследования правилом.

Таблица 14.2. Окупаемость инвестиций при разных циклах выпусков

	Единственный выпуск (через 12 месяцев)	Полугодовые выпуски	Ежеквартальные выпуски
Общие затраты (в долларах)	1,3 млн.	1,4 млн.	1,6 млн.
Общая прибыль за два года (в долларах)	3,6 млн.	4,8 млн.	5,25 млн.
Чистая прибыль за два года (в долларах)	2,3 млн.	3,4 млн.	3,65 млн.
Денежные вложения (в долларах)	1,3 млн.	0,7 млн.	0,45 млн.
Внутренний коэффициент окупаемости (как заменитель окупаемости инвестиций)	9,1%	15,7%	19,5%

Планирование с целью быстрого обучения и резкая смена стратегии по мере надобности

Никакое предварительное прогнозирование или предсказание неспособно заменить выполнение конкретной работы, быстрое обучение и резкую смену стратегии по мере надобности. Под резкой сменой стратегии по мере надобности подразумевается изменение направления, опираясь на то, чему удалось научиться. Риез определяет резкую смену стратегии как "структурированную коррекцию курса для проверки новой основополагающей гипотезы в отношении продукта, стратегии и движущей силы роста" [Ries, Eric. 2011]. Подобно горнолыжнику Джону, мы должны быть готовы быстро и резко сменить направление нашего движения, когда узнаем, что наш текущий план больше не действителен.

Как упоминалось в главе 3, наша цель — продвигаться вперед через петли обучения быстро и экономично. Поэтому мы должны непременно ввести обучение в структуру наших планов как главную цель. Получая скорую ответную реакцию, мы определяем, ведут ли наши планы в нужном направлении. И если это не так, то мы резко меняем стратегию или направление нашего дальнейшего движения.

Заключение

В этой главе сделан краткий обзор нескольких принципов планирования в Scrum. Эти принципы позволяют планировать экономически обоснованно, выполняя полезный объем предварительного планирования, уравновешиваемый более подробным оперативным планированием по мере получения путем обучения бо*л*ьших знаний о создаваемом продукте и порядке его разработки. А в следующей главе на конкретных примерах будет показано, как эффективно пользоваться этими принципами для планирования в Scrum на нескольких уровнях.

Глава 15

Многоуровневое планирование

Планирование в Scrum-проектах осуществляется на нескольких уровнях детализации и неоднократно на протяжении всей разработки продукции. В этой главе различные виды деятельности по планированию в Scrum описываются по нисходящей и во взаимной связи. А в ряде последующих глав подробно рассматривается планирование портфеля заказов, продукта (выработка общего замысла), выпуска и спринта.

Краткий обзор

Когда продукция разрабатывается по методике Scrum, планирование этого процесса происходит на нескольких уровнях (рис. 15.1).

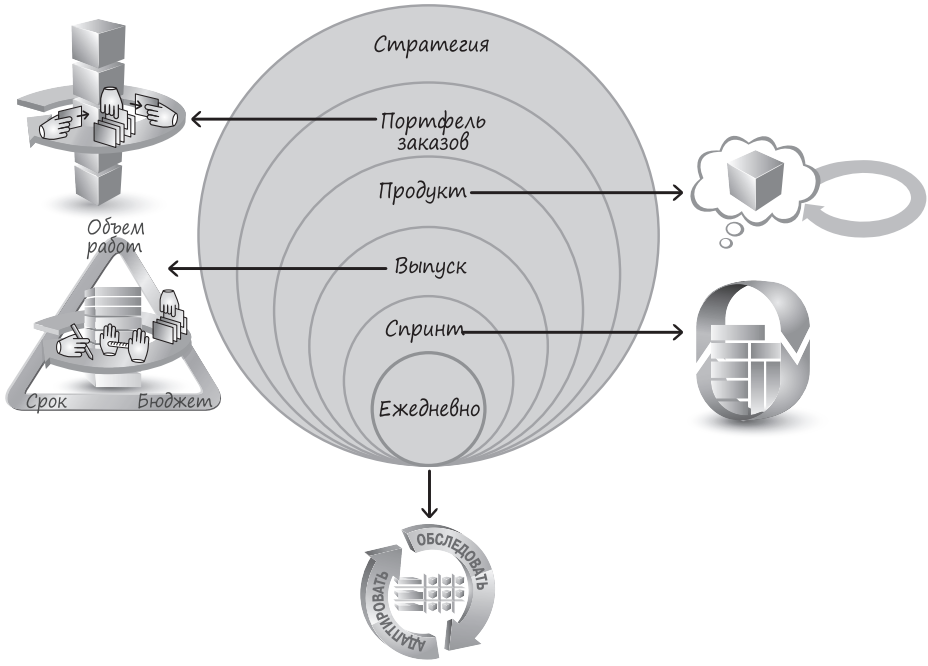


Рис. 15.1. Различные уровни планирования

На самом верхнем уровне происходит стратегическое планирование, которое не рассматривается в данной книге, хотя и считается очень важным для успешной деятельности организации. Формально в Scrum определяется только планирование спринта и ежедневное планирование (на ежедневных летучках). Но большинство организаций могут извлечь немалую пользу из планирования портфеля заказов, продукта и выпуска, и поэтому эти виды планирования упоминаются вкратце в данной главе, а более подробно рассматриваются в последующих главах.

Все виды планирования сведены в табл. 15.1, где указывается промежуток времени, который охватывается каждым из них, кто в них задействован, на что они нацелены и каковы конечные результаты на каждом уровне планирования.

Таблица 15.1. Особенности планирования на каждом уровне

Уровень	Протяженность	Участники	Цель	Конечные результаты
Портфель заказов	Год, а то и больше	Участники проекта и владельцы продуктов	Ведение портфеля заказов на продукты	Задел портфеля заказов и совокупность внутрипроцессных продуктов
Продукт (выработка общего замысла)	До нескольких месяцев и больше	Владелец продукта, участники проекта	Концепция продукта и его развитие во времени	Концепция продукта, график выпуска продукта и функциональных средств верхнего уровня
Выпуск	От трех (или меньше) до девяти месяцев	Вся Scrum-команда, участники проекта	Непрерывное уравешивание потребительской ценности и общего качества с ограничениями, накладываемыми объемом работ, бюджетом и сроками	План выпуска
Спринт	Каждая итерация (от одной недели до календарного месяца)	Вся Scrum-команда	Функциональные средства для выпуска в следующем спринте	Цель и задел спринта
Ежедневно	Каждый день	Scrum-мастер, команда разработчиков	Порядок завершения взятых к обязательству функциональных средств	Обследование текущего положения дел и адаптация с целью организовать работу наилучшим образом на предстоящий день

Чтобы проиллюстрировать каждый из этих уровней планирования, рассмотрим в качестве примера переделку веб-сайта организации Scrum Alliance (`www.`

scrumalliance.org). В 2006 году деятельность этой некоммерческой организации была направлена на распространение методики Scrum по всему миру, и для этой цели был создан веб-сайт, хотя это и было сделано неудачно, поскольку перемещаться по нему было неудобно, а его содержимое было бедным. Когда я стал управляющим директором Scrum Alliance в конце 2006 года, то совет директоров попросил меня прежде всего усовершенствовать веб-сайт. В этих проектных работах я выполнял роль владельца продукта, и поэтому опишу далее иерархию планирования, которое мы выполняли для реализации нового веб-сайта.

Планирование портфеля заказов

Планирование (или ведение) портфеля заказов — это вид деятельности, направленный на выбор тех продуктов, над которыми требуется работать, определение порядка и продолжительности выполнения работ. Несмотря на то что планирование портфеля заказов осуществляется на принципиально более высоком уровне, чем планирование продукта, поскольку оно связано с совокупностью продуктов, одной из главных предпосылок для такого планирования является новый замысел продукта, выработанный в результате планирования продукта.

В 2006 году организация Scrum Alliance была относительно новой, а ее портфель заказов содержал только текущую разработку существующего веб-сайта. Как только был выработан первоначальный замысел веб-сайта Scrum Alliance, совет директоров, принимавший участие в формировании задела портфеля заказов организации Scrum Alliance, утвердил разработку первого выпуска нового веб-сайта.

Планирование продукта (выработка общего замысла)

Цель планирования на уровне продукта, которое иначе называется *выработкой общего замысла*, — выявить суть потенциального продукта и составить приблизительный план для создания данного продукта. Выработка общего замысла начинается с создания сначала представления, а затем задела продукта на верхнем уровне и зачастую — графика выпуска продукта.

Концепция продукта

Концепция продукта дает ясное описание тех областей, где такие участники проекта, как пользователи и заказчики, получают определенную ценность. В данном случае пользователями тогда были 10 тысяч членов организации Scrum Alliance по всему миру, а теперь их насчитывается 150 тысяч, тогда как заказчиками, платившими за новый продукт, — совет директоров Scrum Alliance, представлявший всех членов данной организации.

У нас сложилась следующая концепция веб-сайта Scrum Alliance.

Для людей, интересующихся во всем мире методикой Scrum, новый веб-сайт Scrum Alliance должен стать достоверным источником знаний о Scrum. Он должен быть настолько насыщенным функционально и информационно, чтобы послужить им отправной точкой для ознакомления с методикой Scrum через Интернет и сотрудничества по интересующим темам ее практического применения.

Задел продукта на верхнем уровне

Как только будет составлена концепция продукта, нужно сформировать на верхнем уровне первоначальный задел продукта. В данном случае требовалось переделать веб-сайт Scrum Alliance, и для этого к концу 2006 года уже накопился немалый задел продукта, состоявший из функциональных средств, которые участникам проекта и пользователям хотелось бы иметь на новом и усовершенствованном веб-сайте.

Элементы задела продукта включали в себя следующие эпические пользовательские истории.

- Мне как аттестованному инструктору по Scrum требуется иметь возможность публиковать сведения о моем курсе преподавания методики Scrum на веб-сайте Scrum Alliance, чтобы ознакомить сообщество пользователей с подробностями места и времени проведения этого курса.
- А как потенциальному слушателю мне требуется иметь возможность просматривать подробности проведения всех общедоступных курсов по Scrum, чтобы выбрать среди них наиболее подходящий моим критериям посещения.

Если бы наш продукт был совершенно новым, нам пришлось бы составить хотя бы минимальные предварительные требования, чтобы сформировать задел продукта и оценить наиболее приоритетные его элементы. Но в данном случае в заделе продукта уже имелся ряд элементов, которые послужили нам в качестве отправной точки для составления общей концепции нового веб-сайта.

График выпуска продукта

Как только концепция и задел продукта на верхнем уровне будут составлены, целесообразно построить график выпуска продукта, иногда еще называемый *графиком выпуска целевого продукта*. В частности, *график выпуска продукта* сообщает постепенный характер разработки и своевременного выпуска продукта вместе с важными факторами, побуждающими к отдельным его выпускам.

Ныне многие организации стремятся к *непрерывной разработке*, в ходе которой рабочие функциональные средства развертываются на месте их

эксплуатации, как только они выпускаются. Если ваша организация придерживается именно такой нормы практики, то вам, возможно, и не потребуется строить график выпуска продукта. Но даже если ваша организация намерена развертывать выпускаемые функциональные средства непрерывно, график выпуска продукта может навести ее на мысль о более крупных совокупностях функциональных средств, а также об ограничениях, которые должны определять, какие именно функциональные средства следует разрабатывать одновременно и когда некоторые из них должны быть выпущены. На рис. 15.2 приведен график выпуска продукта в форме, предложенной Люком Хоманном [Hohmann, Luke. 2003].

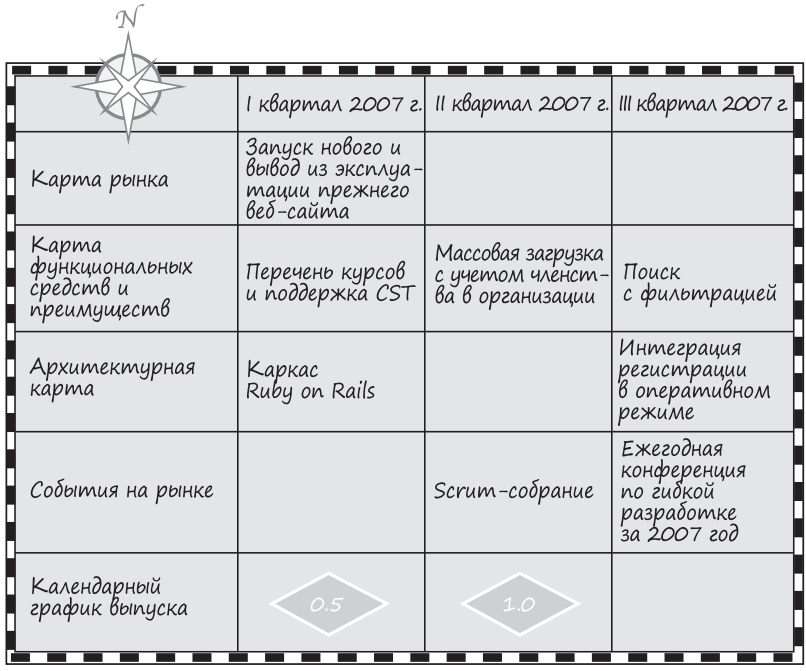


Рис. 15.2. График выпуска продукта для веб-сайта Scrum Alliance

График выпуска продукта, приведенный на рис. 15.2, предусматривал два выпуска: по одному в первом и втором кварталах 2007 года. В частности, выпуск версии 0.5 нового веб-сайта был намечен на первый квартал 2007 года. Такой номер первой версии был выбран потому, что по плану она должна была содержать меньше половины функциональных средств прежнего веб-сайта Scrum Alliance, но включать в себя новые функциональные средства, которые были лучше, чем у прежнего веб-сайта. Желательные средства были сосредоточены на составлении перечня общедоступных по всему миру курсов по Scrum и элементарной поддержке аттестованных инструкторов по Scrum (CST). Выпуск версии 0.5 имел *жестко заданные сроки*, поскольку мы знали, какие именно

функциональные средства нам требовались для нового веб-сайта, прежде чем вывести из эксплуатации прежний веб-сайт. Но в то же время мы не знали, сколько времени потребуется для разработки этих средств, чтобы запустить новый веб-сайт. Более подробно вопросы определения даты отгрузки для выпуска с жестко заданными сроками обсуждаются в главе 18.

Выпуск версии 1.0 также имел жестко заданные сроки. Мы знали, что нам требуется согласовать этот выпуск с конференцией Scrum Alliance, называемой Scrum-собранием, которая была намечена на 7 мая 2007 года в г. Портланд, шт. Орегон. Наша цель состояла в том, чтобы выпустить ряд привлекательных функциональных средств и сделать их доступными к первому дню этой конференции. Но мы не знали, сколько функциональных средств мы можем включить в данный выпуск. Более подробно вопросы определения содержимого выпуска с жестко заданными сроками обсуждаются в главе 18. Таким образом, в графике выпуска продукта для веб-сайта Scrum Alliance мы определили выпуск версии 0.5 с жестко заданным объемом работ, а также выпуск версии 1.0 с жестко заданными сроками.

Независимо от того, какой именно продукт разрабатывается, к концу планирования на уровне продукта должны быть составлены концепция и задел продукта на верхнем уровне с оцененными пользовательскими историями, а дополнительно, хотя и не обязательно, — график выпуска продукта. Могут быть произведены и другие артефакты, чтобы вселить в тех, кто принимают решение, достаточную уверенность в целесообразности разработки продукта. Результаты планирования на уровне продукта в конечном счете послужили предпосылками для планирования портфеля заказов, где первоначальная версия 0.5 переделанного веб-сайта была одобрена советом директоров.

Планирование выпуска

Планирование выпуска служит для поиска компромиссов в отношении объема работ, сроков и бюджета постепенных выпусков продукции. Для проведения большинства проектных работ очень важно и нужно выполнить первоначальное планирование выпуска после выработки общего замысла (на стадии планирования продукта) и до начала первого спринта, связанного с данным выпуском. На данной стадии составляется первоначальный *план выпуска*, где объем проектных работ уравновешивается со сроками выпуска. Чтобы составить ясное представление о том, что и когда можно выпустить к конкретной дате, нужно создать и оценить достаточное количество элементов задела продукта.

Чтобы наглядно представить выпуск, проще всего провести разделительную линию через задел продукта, как показано на рис. 15.3. Все элементы выше разделительной линии планируются в текущем выпуске, а все элементы ниже этой линии — в следующем. Эта линия может смещаться вверх или вниз по заделу продукта по мере составления более полного представления о продукте. Более

подробно порядок определения конкретного положения разделительной линии обсуждается в главе 18.

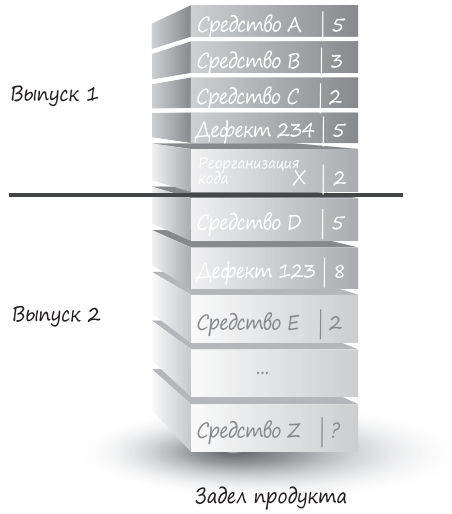


Рис. 15.3. Линия, разделяющая выпуски в заделе продукта

Теперь график выпуска продукта нетрудно связать с его заделом, чтобы более детально проработать содержимое хотя бы ближайшего по срокам выпуска, указанного в графике выпуска продукта (рис. 15.4). Выпуску по этому графику соответствует ряд функциональных средств в заделе продукта.

План выпуска должен также иметь определенные временны*е* рамки, которые могут быть выражены в виде количества спринтов, требующихся для завершения выпуска. Большинство выпусков довольно велики и содержат больше функциональных средств, чем можно создать в течение одного спринта (рис. 15.5).

Планируя выпуск, можно пойти еще дальше, прогнозируя функциональные средства, которые следует выпустить в течение первых двух спринтов. Это может принести определенную пользу, когда нескольким командам потребуется скоординировать свою работу или же когда команде нужно будет заблаговременно затребовать дополнительное оборудование, инструментальные средства или помощь. Но прогнозировать больше, чем на две спринта вперед, практически всегда не имеет особого смысла, поскольку это нарушает принцип своевременности и достаточности планирования.

Планирование спринта

Отдельные элементы из задела продукта, над которыми Scrum-команде предстоит работать в следующем спринте, согласовываются при планировании спринта, которое происходит в начале каждого спринта. В течение этого

мероприятия команда формирует задел спринта, т.е. описание работ на уровне отдельных задач, которые должны быть выполнены, чтобы завершить готовые элементы задела продукта (рис. 15.6). В ходе планирования спринта команда выполняет оперативное и подробное планирование на следующем уровне. Более подробно вопросы планирования спринта обсуждаются в главе 19.

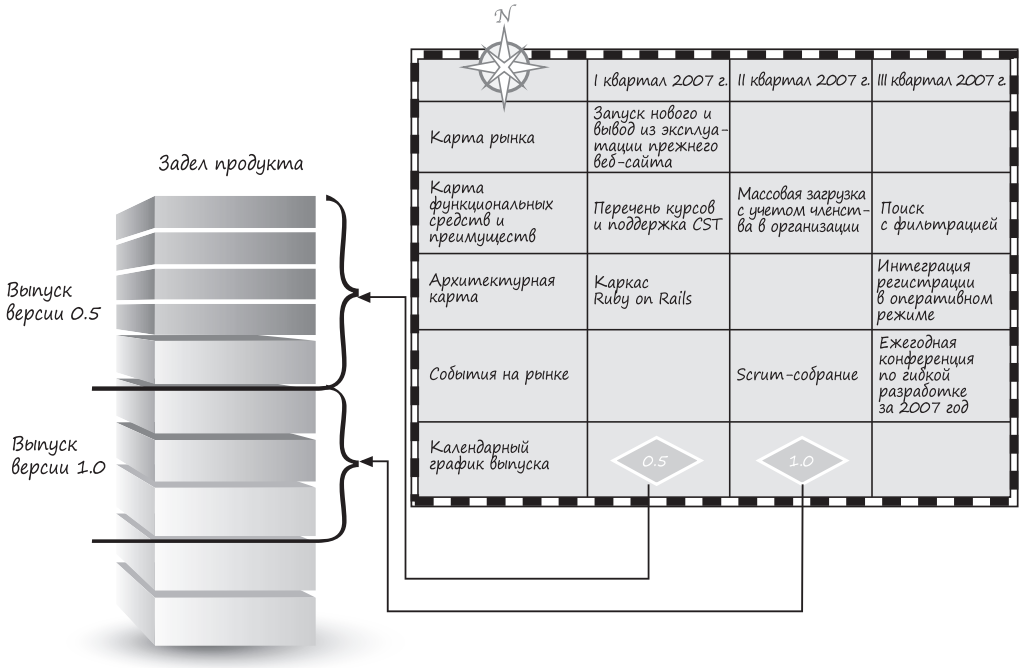


Рис. 15.4. Выпуски по графику выпуска продукта увязываются с заделом продукта

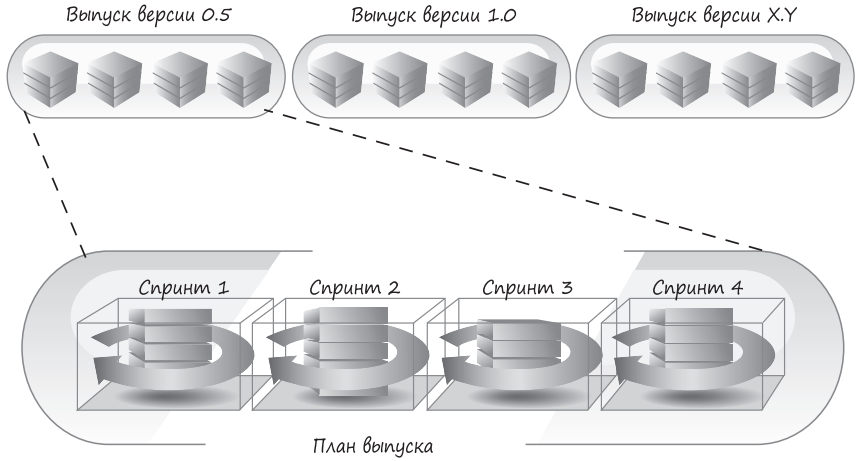


Рис. 15.5. Выпуск может охватывать один спринт или больше

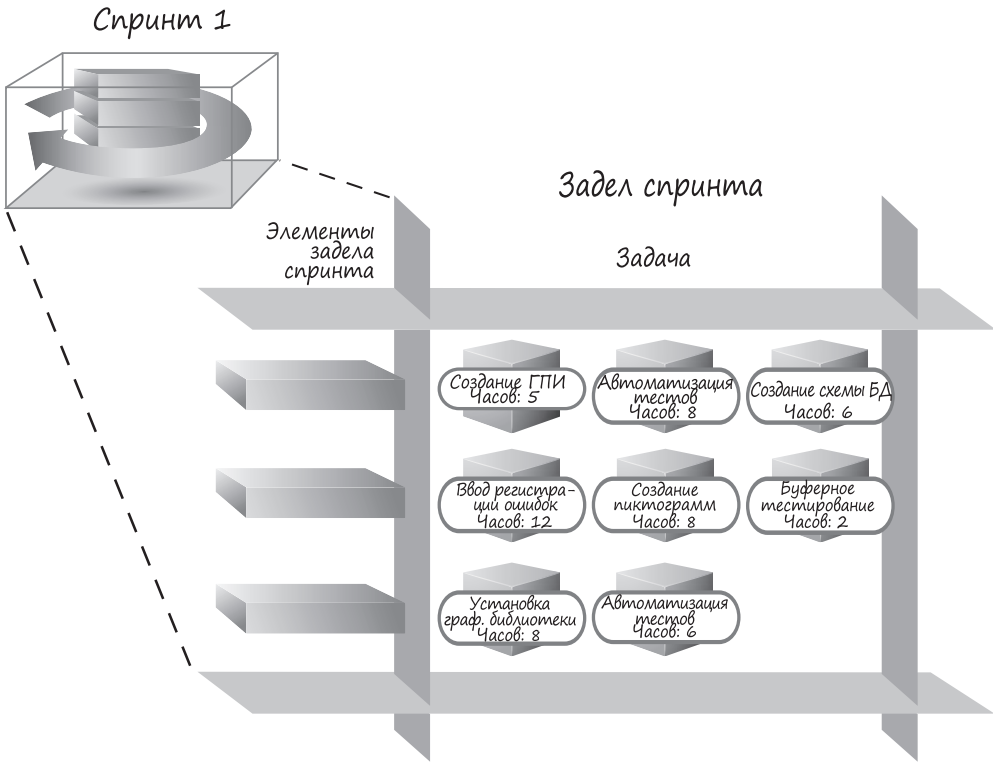


Рис. 15.6. У каждого спринта имеется свой задел

Ежедневное планирование

Наиболее подробный уровень планирования происходит во время ежедневных летучек команды разработчиков. Напомним, что на данном мероприятии члены команды разработчиков собираются вместе, и каждый из них по очереди сообщает, что он сделал с момента последней ежедневной летучки, что он собирается сделать в нынешний день и что ему в этом препятствует.

В ходе ежедневной летучки члены команды разработчиков совместно и довольно наглядно описывают общий план на текущий день. Это позволяет также команде предупреждать о трудных участках в работе. Например, кто-нибудь из присутствующих на летучке может сказать: “Сегодня я собираюсь работать над задачей хранимой процедуры и должен сделать это до обеда. Тем, кто собирается работать над задачей бизнес-логики, следует иметь в виду, что задача бизнес-логики имеет решающее значение для выполнения всей нашей работы в этом спринте, и поэтому они должны приступить к этой задаче сразу после обеда”. Подобные сообщения позволяют очень быстро выявить возможные помехи в работе и способствуют более плавному ходу работ в течение спринта.

Заключение

В этой главе было показано, каким образом планирование осуществляется на разных уровнях детализации в ходе выполнения проектных работ по методике Scrum. На рис. 15.7 наглядно показаны все артефакты, производимые на этих уровнях, а также их взаимосвязанный характер, за исключением уровней планирования портфеля заказов и ежедневного планирования. А в последующих главах вопросы планирования портфеля заказов, продукта, выпуска и спринта будут обсуждаться более подробно.

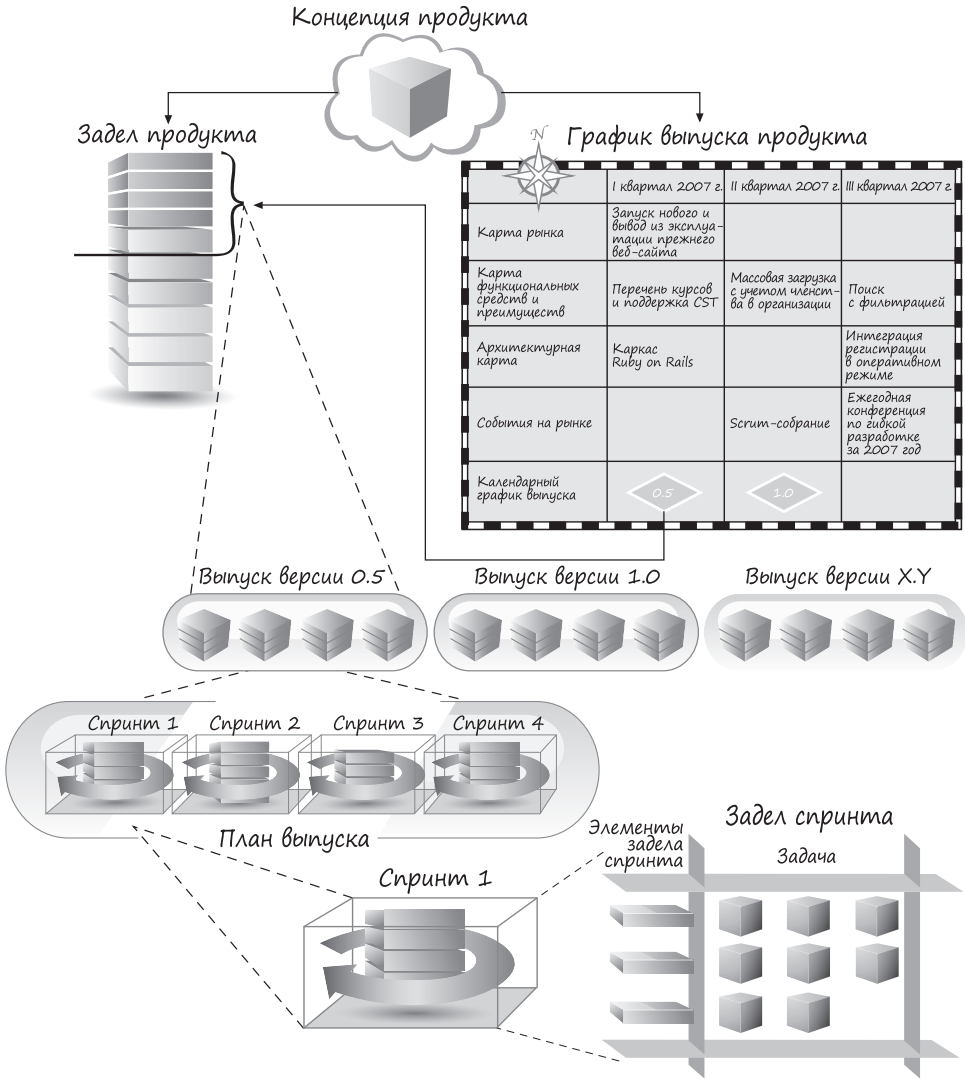


Рис. 15.7. Иерархия уровней планирования в Scrum

Глава 16
Планирование портфеля заказов

В большинстве организаций требуется или необходимо производить одновременно не один продукт. И таким организациям приходится делать экономически обоснованный выбор относительно ведения их портфеля заказов. Кроме того, им нужно согласовать ведение портфеля заказов или управление производственным процессом с основными нормами практики гибкой разработки, иначе возникнет существенное разногласие с гибким подходом, применяемым на отдельных уровнях разработки продукции. В этой главе рассматриваются 11 стратегий планирования портфеля заказов, сгруппированных по категориям календарного планирования, притока и оттока продукции. И в завершение главы поясняется, каким образом определить, следует ли вкладывать больше труда во внутрипроцессные продукты или же этого не стоит делать.

Краткий обзор

Планирование портфеля заказов (или его ведение) — это вид деятельности, направленной на определение тех элементов в заделе портфеля заказов, над которыми следует работать, а также порядка и продолжительности подобных работ. Элементом задела портфеля заказов может быть продукт, прирост продукта (один его выпуск) или проект, если в организации отдается предпочтение планированию работы над отдельными проектами. Термином *продукт* в этой главе в общем обозначаются все разновидности элементов в заделе портфеля заказов.

Как показывает мой опыт, в большинстве организаций, как практикующих, так и не практикующих гибкую разработку, очень плохо справляются с планированием портфеля заказов. Во многих организациях на уровне планирования портфеля заказов присутствуют процессы, которые полностью противоречат основным принципам гибкой разработки. И когда это происходит, решения, принимаемые на уровне портфеля заказов, нарушают быстрый и гибкий ход проектных работ. Поэтому в этой главе поясняется, как избежать подобных разногласий, выполняя планирование портфеля заказов в полном соответствии с принципами гибкой разработки.

Временны*е* рамки

Процесс планирования портфеля заказов никогда не завершается. Портфель заказов необходимо вести до тех пор, пока имеются продукты для разработки и сопровождения.

Как показано на рис. 15.1 в главе 15, планирование портфеля заказов имеет отношение к совокупности продуктов, и поэтому его рамки шире, а уровень выше, чем у планирования отдельных продуктов (т.е. выработки общего замысла). Но если планирование портфеля заказов осуществляется на более высоком уровне, то это совсем не означает, что оно предшествует планированию продукта. В действительности конечный результат планирования или выработки общего замысла нового продукта служит предпосылкой для планирования портфеля заказов. На основании данных, получаемых от выработки общего замысла, при планировании портфеля заказов принимается решение, следует ли финансировать разработку продукта и в каком порядке расположить его в *заделе портфеля заказов*. Но планирование портфеля заказов не ограничивается только новыми, предусмотренными для разработки продуктами. Оно происходит через регулярные промежутки времени для анализа внутрипроцессных продуктов, которые уже находятся в процессе разработки, эксплуатации или продажи.

Участники

Планирование портфеля заказов направлено как на новые, так и на *внутрипроцессные продукты*, и поэтому в нем задействованы соответствующие внутренние участники проектов, владельцы отдельных продуктов, а нередко и ведущие архитекторы и технические специалисты. Участники данного мероприятия должны иметь достаточно широкие деловые взгляды, чтобы правильно расставить по приоритетам элементы в заделе портфеля заказов по отношению к внутрипроцессным продуктам. В некоторых организациях участники данного мероприятия совместно образуют комитет по утверждениям, коллегиальный орган управления или какой-то другой, аналогичный субъект, надзирающий за процессом планирования портфеля заказов.

Владельцы продуктов также принимают участие в планировании портфеля заказов как ответственные за соответствующие продукты и активно выступающие за предоставление им нужных ресурсов. Нередко для того, чтобы важные технические ограничения были учтены при принятии решений, к планированию портфеля заказов привлекаются ведущие архитекторы и технические специалисты.

Как упоминалось ранее, предпосылками для планирования портфеля заказов служат сведения как о новых продуктах, предусмотренных для разработки и претендующих на включение в задел портфеля заказов, так и о внутрипроцессных продуктах. Новым продуктам сопутствуют такие данные, собранные

в процессе выработки общего замысла, как, например, затраты на разработку, ее продолжительность, риск, ценность и пр. Внутрипроцессным продуктам сопутствуют такие их собственные данные, как, например, промежуточная ответная реакция заказчиков, обновленные затраты, календарный график и оценочный объем работ, объемы технического долга, а также сведения рыночного характера, помогающие определить дальнейшее продвижение этих продуктов на рынок.

Процесс

Процесс планирования портфеля заказов наглядно демонстрируется на рис. 16.1.

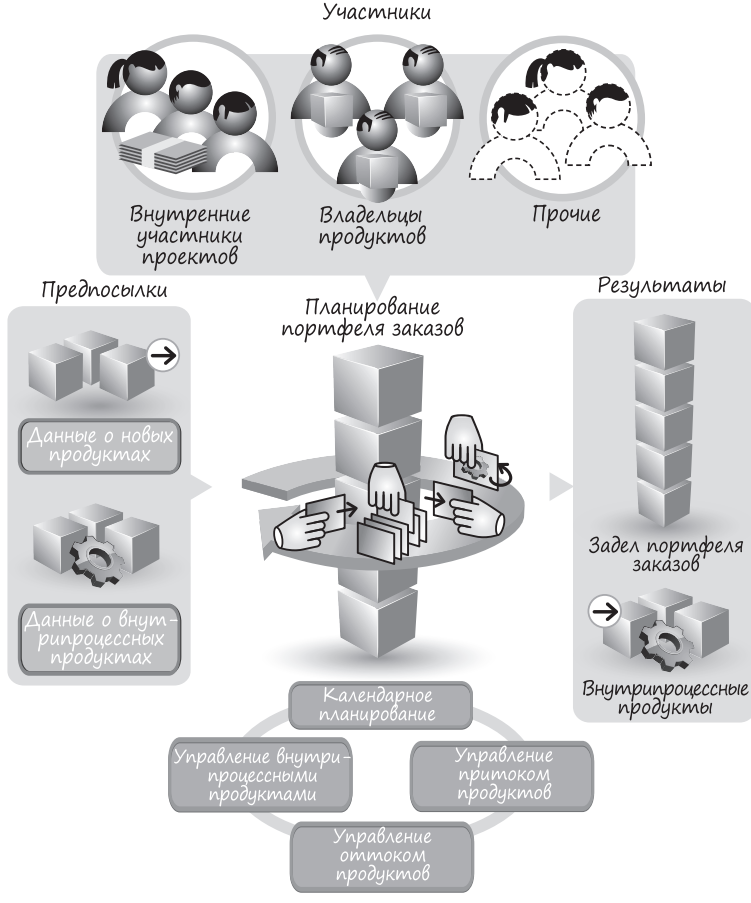


Рис. 16.1. Процесс планирования портфеля заказов

Планирование портфеля заказов приводит к двум результатам. Во-первых, это задел портфеля заказов, т.е. перечень очередности разработки перспективных продуктов, которые были утверждены, но разработка которых еще не началась. И

во-вторых, это ряд действующих продуктов, т.е. вновь утвержденных и намеченных для немедленной разработки, а также внутрипроцессных продуктов, находящихся в соответствующем процессе и утвержденных для его продолжения.

Чтобы достичь этих результатов, участники планирования портфеля заказов распределяются по следующим четырем категориям деятельности: календарное планирование, управление притоком продуктов, управление оттоком продуктов, а также управление внутрипроцессными продуктами. Отдельные стратегии планирования, связанные с этими категориями, представлены на рис. 16.2.

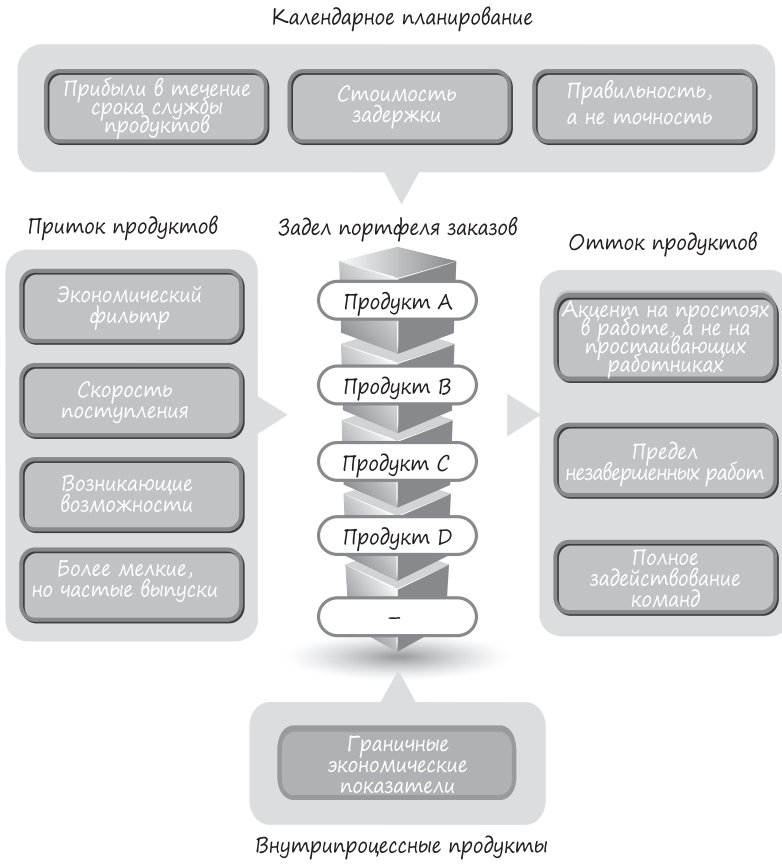


Рис. 16.2. Стратегии планирования портфеля заказов

Стратегии календарного планирования помогают определить правильный порядок расположения продуктов в заделе портфеля заказов. Стратегиями управления притоком продуктов участники руководствуются, чтобы знать, когда следует вводить элементы в задел портфеля заказов. А стратегии управления оттоком продуктов извещают участников о моменте, когда продукт следует извлечь из задела портфеля заказов. И наконец, стратегии управления внутрипроцессными

продуктами позволяют решить, когда следует сохранить, резко изменить, выпустить или прекратить работу над продуктом, находящимся в соответствующем процессе. Все одиннадцать стратегий, составляющих четыре упомянутые здесь категории, подробнее рассматриваются в остальной части этой главы.

Стратегии календарного планирования

При планировании портфеля заказов ограниченные ресурсы организации должны быть выделены для разработки продуктов экономически обоснованно. И хотя выбрать порядок разработки продуктов можно самыми разными способами, далее рассматриваются следующие три стратегии.

- Оптимизация прибылей в течение срока службы продуктов.
- Расчет стоимости задержки.
- Оценивание ради правильности, а не точности.

Оптимизация прибылей в течение срока службы продуктов

Чтобы оптимизировать порядок расположения продуктов в портфеле заказов, нужно выбрать показатель, позволяющий выяснить, насколько работоспособными оказываются наши действия по оптимизации. Райнерцен рекомендует пользоваться экономической инфраструктурой, где все решения и компромиссы рассматриваются в контексте *прибылей в течение срока эксплуатации* как стандартизированной и полезной единицы измерения [Reinertsen, Donald G. 2009a]. Исходя из этой рекомендации цель должна состоять в расположении элементов в заделе портфеля заказов, чтобы довести до максимума общие прибыли в течение срока эксплуатации.

Прибыли в течение срока эксплуатации конкретного продукта составляют общий потенциал для получения прибыли от продукта за весь срок его службы. Что же касается планирования портфеля заказов, то наибольший интерес представляет оптимизация прибылей в течение эксплуатации продуктов из всего портфеля заказов в целом, а не отдельного продукта. Поэтому можно было бы частично оптимизировать отдельные продукты, чтобы оптимизировать весь портфель заказов в целом [Poppendieck, Mary, and Tom Poppendieck. 2003]. Таким образом, цель стратегии оптимизации прибылей в течение эксплуатации состоит в том, чтобы выявить такой порядок расположения элементов в заделе портфеля заказов, который обеспечивает наибольшие прибыли в течение эксплуатации продуктов во всем портфеле заказов (соответствующий пример расчета стоимости задержки приведен в следующем разделе).

Райнерцен далее утверждает, что двумя наиболее важными показателями для оценивания того влияния, которое оказывается на прибыли в течение срока

эксплуатации, являются стоимость задержки и продолжительность работ (удобным заменителем которой служит объем работ или размер продукта). В зависимости о того, насколько эти показатели сходны (или не сходны) у продуктов в портфеле заказов, он предлагает выбрать один из трех методов календарного планирования, перечисленных в табл. 16.1.

Таблица 16.1. Разные методы календарного планирования портфеля заказов

(Если) стоимость задержки	(И) Продолжительность/ Объем/Размер	(То) Метод календарного планирования
Одинаковая у всех продуктов	Отличается у продуктов	Сначала планируется самая короткая работа
Отличается у продуктов	Одинаковая у всех продуктов	Сначала планируется работа с самой большой стоимостью задержки
Отличается у продуктов	Отличается у продуктов	Сначала планируется самая коротковзвешенная работа

Если стоимость задержки у всех продуктов одинаковая, то в качестве стратегии календарного планирования выбирается планирование сначала самой короткой работы. Если же продукты имеют одинаковые размеры (т.е. одинаковую продолжительность работ), то в качестве стратегии календарного планирования выбирается планирование работы сначала над продуктами с самой большой стоимостью задержки. А если стоимость задержки и продолжительность работ могут отличаться, что является обычным делом для разработки продукции, то экономически оптимальное расположение продуктов в портфеле заказов достигается планированием сначала самой коротковзвешенной работы (WSJF), которая рассчитывается путем деления стоимости задержки на продолжительность работ (или объем работ, требующихся для реализации продукта). Далее рассматривается расчет стоимости задержки и соотношения затрат труда и средств на разработку продуктов в портфеле заказов.

Расчет стоимости задержки

Если элементы располагаются в заделе портфеля заказов по порядку, то и работа над одними продуктами должна начинаться прежде, чем над другими. А начало работ над теми продуктами, которые не требуют принятия немедленных мер, задерживается. Следовательно, задерживается и срок их выпуска, для которого существует количественно оцениваемая стоимость.

Как пояснялось в главе 3, стоимость задержки предоставляет важную информацию для принятия обоснованных экономических решений. Тем не менее во многих организациях не готовы ответить на следующий простой вопрос: “Если разработка продукта задерживается на один месяц, то какова будет стоимость

такой задержки в исчислении прибылей в течение эксплуатации продукта?" Даже не ведая о стоимости задержки, многие организации предпочитают формировать свой портфель заказов простым (и зачастую неверным) методом, располагая в нем сначала элементы, от которых ожидается высокая прибыль.

В примере, приведенном в табл. 16.2, окупаемость инвестиций в проект А составляет 20%, а в проект В — 15%. Если применить стратегию календарного планирования сначала высокоприбыльных проектов, то работу над проектом А следует начать раньше, чем работу над проектом В, поскольку окупаемость инвестиций в него выше. На первый взгляд такая стратегия кажется благоразумной, но в ней не учитывается стоимость задержки работ над каждым продуктом, которая существенно изменяет расчет прибыли в течение срока службы продукта. Что, если, например, стоимость задержки проекта А на один месяц составит 5 тыс. долларов, а проекта В — 75 тыс. долларов, как показано в табл. 16.2? В таком случае задержка работы над проектом В для того, чтобы приступить сначала к работе над проектом А, оказывает существенное влияние на прибыльность в течение срока эксплуатации продуктов во всем портфеле заказов в целом.

Таблица 16.2. Пример применения стоимости задержки для формирования портфеля заказов

	Проект А	**Проект В**
Окупаемость инвестиций	**20%**	**15%**
Стоимость задержки на 1 месяц (в долларах США)	**5** тыс.	**75** тыс.

Стоимость задержки олицетворяет тот факт, что время может и должно оказывать влияние на большинство экономических показателей. В предыдущем примере окупаемость инвестиций в проекты А и В рассчитана, исходя из конкретных зависящих от времени предположений (например, когда начнется и завершится разработка, какие ресурсы будут доступны в это время, во что эти ресурсы обойдутся, какую цену потребители будут готовы платить за продукт со временем, какие технологические и коммерческие риски существуют, какова вероятность их появления и как это может сказаться на затратах). Стоит задержать или ускорить разработку, и величины этих показателей могут измениться, что случается довольно часто. Следовательно, стоимость задержки — это не только фактор, который следует принимать во внимание, расставляя по приоритетам элементы в портфеле заказов, но и измерение времени, которое оказывает влияние на все остальные показатели расстановки по приоритетам, в том числе на затраты, выгоды, знания и риски.

Мне чаще всего приходится слышать жалобы на то, что не совсем ясно, как же рассчитывать стоимость задержки. Чаще всего эти жалобы не обоснованы, поскольку для эффективного расчета стоимости задержки достаточно составить

две разные модели электронных таблиц: одну — без учета задержки, а другую — с учетом задержки.

Леффингуэлл предлагает следующую модель для расчета стоимости задержки, в которой учитываются три свойства продукта [Leffingwell, Dean. 2011].

- Пользовательская ценность — потенциальная ценность с точки зрения пользователя.
- Ценность времени — каким образом происходит снижение пользовательской ценности во времени.
- Реализация снижения или вероятности риска — ценность с точки зрения смягчения риска или использования его вероятности.

Чтобы рассчитать стоимость задержки в разработке продукта, каждому из перечисленных выше трех свойств следует присвоить отдельный номер стоимости задержки, используя шкалу от **1** (наименьшая стоимость) до **10** (наибольшая стоимость). Общая стоимость задержки в разработке продукта рассчитывается как сумма трех отдельных стоимостей задержки.

В качестве альтернативного (и зачастую эффективного) метода принятия обоснованных решений при календарном планировании можно охарактеризовать общий профиль стоимости задержки (рис. 16.3). Каждый из этих профилей более подробно описывается в табл. 16.3.

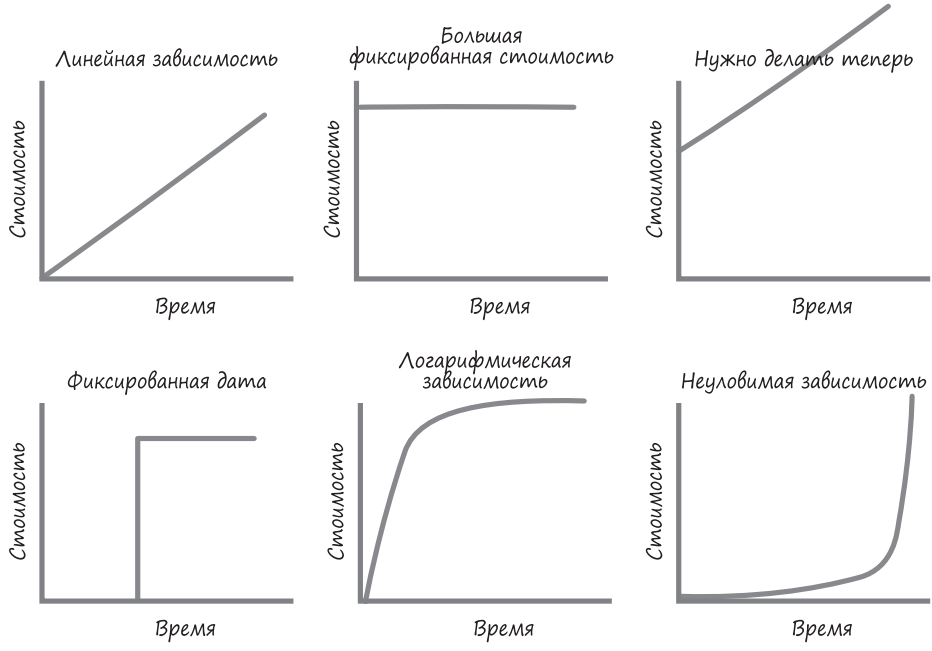


Рис. 16.3. Профили стоимости задержки

Таблица 16.3. Профили стоимости задержки

Имя профиля	Описание
Линейная зависимость	Продукт со стоимостью задержки, которая увеличивается с постоянной скоростью
Большая фиксированная стоимость	Продукт, накапливающий единовременную стоимость задержки, если он не разрабатывается немедленно. Например, значительная отдача получается только после выпуска продукта
Нужно делать теперь	Продукт нужно делать теперь, поскольку стоимость задержки сразу же резко возрастет. Это может быть, например, продукт, отсутствие которого немедленно повлечет за собой потерю доходов или экономию затрат, которая будет только возрастать со временем
Фиксированная дата	Продукт, который должен быть выпущен к фиксированной будущей дате, а следовательно, он будет иметь нулевую стоимость задержки до тех пор, пока не наступит фиксированная дата. Как только эта дата будет пройдена, начнет накапливаться полная стоимость, по крайней мере, первоначальной задержки
Логарифмическая зависимость	Продукт, очень рано накапливающий б*о*льшую часть стоимости задержки, а впоследствии эта задержка постепенно сокращается
Неуловимая зависимость	Продукт (или основной объем работ), не имеющий "очевидной" стоимости задержки в течение продолжительного периода времени, а затем неожиданно резко накапливающий большую стоимость задержки. Например, во многих организациях технический долг первоначально считается небольшим, а стоимость задержки его возмещения не учитывается. Но, как пояснялось в главе 8, технический долг может достичь переломного момента, когда стоимость задержки других работ становится заметной и очень большой

Если расчет точного номера стоимости задержки отнимает немало времени или чреват ошибками, следует выбрать соответствующий профиль стоимости задержки (или создать новый) и пользоваться им вместо конкретного номера стоимости задержки, принимая решения при календарном планировании. Но следует ли применять стоимость задержки в тех организациях, где продукты разрабатываются в таких сильно регламентированных отраслях, как производство медицинского оборудования или здравоохранение, где безопасность пациентов имеет решающее значение? Такие важные факторы, безусловно, должны приниматься во внимание, когда определяются свойства продукта. Но важные свойства этих факторов могут учитываться для выражения стоимости задержки с точки зрения прибыли в течение срока службы продуктов.

Например, в Соединенных Штатах организации, занимающиеся медицинским страхованием или обслуживанием, должны пользоваться определенными кодами для обозначения диагнозов конкретных болезней и клинических процедур в заявках, формах обращения и прочих электронных транзакциях. Для

таких кодов раньше существовал стандарт "Международная классификация болезней", 9-я редакция (ICD-9-CM). Но ему на смену пришел стандарт ICD-10-CM, вступивший в силу с 1 октября 2013 года, который должны соблюдать все организации, подпадающие под действие американского закона HIPPA (Health Insurance Portability and Accountability Act of 1996 — Закон об унификации и учете в области медицинского страхования).

У многих из подобных организаций имеется портфель заказов на продукты, которые требуют исправления аналогично проблеме 2000 года, неожиданно возникшей на переломе тысячелетий. А поскольку все продукты в портфеле заказов на исправление имеют профиль стоимости задержки по фиксированной дате (см. рис. 16.3), то для рационального определения порядка выполнения работ по исправлению этим организациям пришлось учитывать величину стоимости задержки (с точки зрения прибылей в течение срока службы) для каждого продукта, если эти работы *не* были завершены к 1 октября 2013 года. Так, если один важный продукт не отвечал упомянутому выше новому стандарту, то он мог нанести убытки до 100 млн долларов в год, тогда как другой аналогичный продукт — до 5 млн долларов. Таким образом, расчет стоимости задержки имел решающее значение для упорядочения работ по исправлению в портфеле заказов подобных организаций экономически обоснованным способом.

Оценивание ради правильности, а не точности

Для правильного календарного планирования элементов в заделе портфеля заказов нужно также ясно понимать соотношение затрат труда и средств на реализацию этих элементов, поскольку затраты оказывают влияние на прибыли в течение срока эксплуатации продуктов. Оценивая размеры элементов в заделе портфеля заказов, нужно стремиться к правильности, а не к точности, поскольку на момент, когда требуется первая оценка, имеется весьма ограниченное количество данных.

Как обсуждалось в главе 7, в некоторых организациях предпочитают оценивать элементы в заделе портфеля заказов, используя размеры футболок, а не точные цифры. При этом каждый размер футболки соответствует определенному диапазону затрат. Пример такого соответствия в одной из организаций приведен в табл. 16.4, где в приблизительный диапазон затрат включены трудозатраты, обычно составляющие *б*ольшую часть затрат организации на продукт, а также капитальные расходы и *л*юбые другие затраты, которые считаются материальными для разработки продукта. Преимущество оценивания по размерам футболок заключается в том, что оно проводится быстро и зачастую довольно точно, предоставляя практическую информацию на уровне портфеля заказов.

Таблица 16.4. Пример оценивания затрат по размерам футболок

Размер	Приблизительный диапазон затрат (в долларах США)
Очень маленький (XS)	От **10** до **25** тыс.
Малый (S)	От **25** до **50** тыс.
Средний (M)	От **50** до **125** тыс.
Большой (L)	От **125** до **350** тыс.
Очень большой (XL)	Свыше **350** тыс.

Но насколько правильное оценивание оказывается точным? Обратимся к конкретному примеру. В упомянутой выше организации проектный отдел тратил в прошлом немало времени на попытки очень точного оценивания проектных работ. Его сотрудники не были уверены в точности размеров футболок, но все соглашались с тем, что их нужно попробовать для подобного оценивания. Вскоре после этого отдел маркетинга предложил проектному отделу замысел нового проекта. В ходе обсуждения этого проекта в проектном отделе ему был присвоен средний размер (M).

На основании этой оценки в отделе маркетинга удалось выяснить, превысит ли прибыль от выполнения данного проекта затраты на проект среднего размера (от 50 до 125 тыс. долларов). Это оказалось так же полезно, как и в прошлом, когда в проектном отделе тратили много времени на точное, но неправильное оценивание затрат около 72381,27 доллара на проект среднего размера. И тогда в этой организации пришли к выводу, что диапазоны затрат по размерам футболок оказались достаточно точными, исключали напрасные расходы времени и средств, но не повышали ожидания до слишком высокого уровня и не создавали ложное ощущение надежности.

Стратегии управления притоком продуктов

Как обсуждается далее в главе 17, в процессе выработки общего замысла уточняется концепция нового продукта и собирается информация, которая требуется, чтобы принять решение о целесообразности финансирования разработки этого продукта. Стратегии управления притоком продуктов позволяют применить в организации экономический фильтр для принятия подобного решения. Кроме того, они позволяют уравновесить скорость, с которой продукты вводятся в задел портфеля заказов, со скоростью, с которой они извлекаются из задела портфеля заказов, а также предотвратить узкие места в портфеле заказов, практикуя более мелкие, но частые выпуски.

Применение экономического фильтра

В результате выработки общего замысла составляется концепция продукта и получается информация, требующаяся для выяснения порога доверия к выработке общего замысла (планированию продукта; см. главу 17). Этим результатом являются данные о новом продукте, которые служат предпосылкой для планирования портфеля заказов (см. рис. 16.1). На основании этих данных в организации принимается решение о целесообразности перехода к разработке нового продукта. Такой вид деятельности называется *применением экономического фильтра* к новому продукту с целью выяснить, насколько он удовлетворяет требованиям к финансированию, принятым в организации (рис. 16.4).

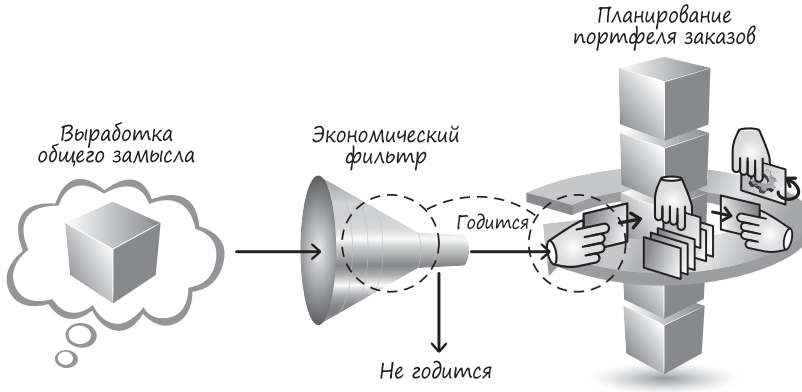


Рис. 16.4. Применение экономического фильтра

Несмотря на то что каждой организации придется самостоятельно определять экономический фильтр, который лучше всего соответствует принятым в ней правилам финансирования, хороший экономический фильтр должен быстро отобрать утверждение любой возможности, которая доставляет большую ценность, значительно превосходящую затраты на ее получение, а все остальное — отсеять. Если получаемая в итоге ценность разработки продукта значительно превосходит затраты на его разработку, то вряд ли стоит тратить дополнительное время на обсуждение такой возможности, а лишь утвердить ее и перейти к расположению данного продукта в заделе портфеля заказов. Если же незначительные отличия ценности продукта от затрат на его разработку вызывают споры перед принятием решения, то такой продукт следует отклонить, поскольку его разработка, очевидно, не даст никаких заметных экономических выгод. У большинства организаций просто имеется слишком много возможностей для разработки очень ценных продуктов, чтобы тратить зря время на обсуждение спорных возможностей.

Уравновешивание скорости поступления и отправления

На практике требуется уравновесить устойчивый поток продуктов, поступающих в задел портфеля заказов, с таким же устойчивым потоком продуктов, извлекаемых из задела портфеля заказов (рис. 16.5). Но в то же время нельзя перегружать задел портфеля заказов, вводя в него слишком много продуктов. Ведь это будет иметь эффект перегрузки системы.

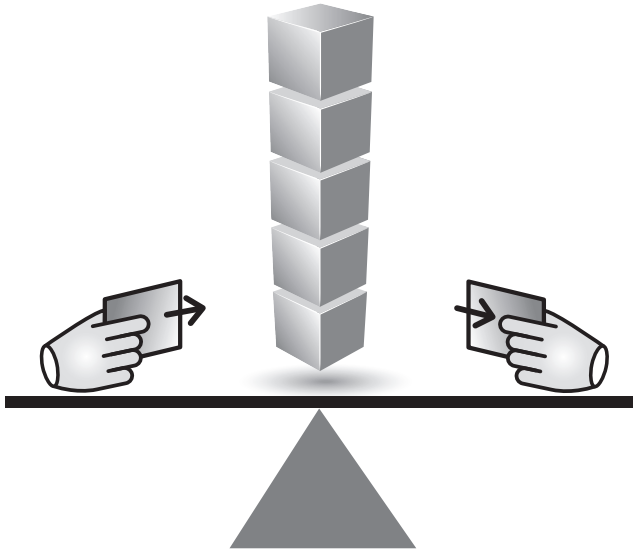


Рис. 16.5. Уравновешивание притока и оттока продуктов в заделе портфеля заказов

Чтобы стали понятнее причины, допустим, что вы решили пообедать в ресторане. Вы садитесь в свою машину и отправляетесь в путь. Подъезжая к ресторану, вы обнаруживаете рядом с ним большой автобус, из которого выходит большая группа голодных туристов и направляется в ресторан.

Что же делать? Войти в ресторан и попытаться пообедать в нем? В таком случае подумайте о последствиях высадки в ресторан целого десанта голодных туристов, жаждущих пообедать одновременно с вами. Вероятнее всего, их заказы превзойдут возможности ресторана обслужить столько посетителей сразу. И если вы все же рискнете пообедать в нем, то будете долго ждать, пока вас обслужат, и вряд ли получите удовольствие от обеда. Лучше уж вернуться в машину и отправиться в другой ресторан!

Во многих организациях проводятся ежегодные мероприятия по стратегическому планированию — как правило, в течение третьего квартала их бюджетного года. Нередко одним из результатов стратегического планирования

оказывается полный перечень продуктов, над которыми организация собирается работать в течение следующего бюджетного года. И эти продукты одновременно вводятся в задел портфеля заказов, подавляя своим количеством процесс планирования портфеля заказов.

Речь, конечно, не идет о том, чтобы организации отказались от стратегического планирования. Они непременно должны выбрать для себя стратегическое направление, но не уточняя во *всех* подробностях (на уровне отдельных продуктов), как достичь стратегической цели. Рассмотрение сначала на ежегодном совещании всех видов работ на следующий бюджетный год или еще более продолжительный период времени, а затем одновременный ввод всех этих элементов в задел портфеля заказов является очень важным (а в некоторых организациях — необратимым) решением, которое принимается в условиях значительной неопределенности. К тому же оно нарушает принцип наличия свободы выбора вариантов планирования до самого последнего момента (см. главу 14).

Решение заполнить продуктами сразу весь портфель заказов нарушает также принцип использования экономически обоснованных объемов партий, как пояснялось в главе 3. Обработка крупных партий продуктов с целью расположить их в определенном порядке в заделе портфеля заказов оказывается очень дорогим и убыточным делом, поскольку планировать приходится на целый год вперед, а то и больше. Это обходится дорого не только потому, что обрабатывать нужно много продуктов, но и потому, что большое количество элементов в заделе портфеля заказов усложняет календарное планирование, рассматривавшееся ранее в этой главе. Последовательность элементов намного проще расположить по порядку в заделе портфеля заказов, если их окажется в ней меньше. В действительности, если в заделе портфеля заказов имеется небольшое количество элементов, их достаточно расположить по порядку любым способом, кроме откровенно примитивной расстановки по приоритетам.

В качестве меры борьбы с заполнением задела портфеля заказов сразу всеми продуктами их можно вводить в него через более короткие промежутки времени, например, ежемесячно или хотя бы ежеквартально, а не один раз в год. Благодаря этому значительно сокращаются затраты труда и средств на анализ и ввод новых продуктов в портфель заказов, а также обеспечивается бо́льшая устойчивость и предсказуемость всего планирования портфеля заказов в целом.

Внимание следует уделить и более мелким продуктам, как предусматривает рассматриваемая далее стратегия более мелких, но частых выпусков. В конечном итоге это должно привести к постоянному потоку продуктов, которые завершаются, освобождая производственные ресурсы для извлечения новых продуктов из задела портфеля заказов с завидной регулярностью. Ведь частое извлечение новых продуктов из задела портфеля заказов помогает уравновесить приток и отток продуктов.

И наконец, когда размер задела портфеля заказов начинает увеличиваться, поток продуктов в задел можно отрегулировать, настроив экономический фильтр на повышение строгости критериев утверждения продуктов, чтобы через этот фильтр проходили только те продукты, которые имеют большую ценность. Благодаря этому снижается скорость поступления продуктов и устанавливается более оптимальное равновесие со скоростью их отправления.

Быстрый охват возникающих возможностей

При планировании портфеля заказов нужно быстро охватывать *возникающие возможности*. Возможность считается возникающей, если раньше она была неизвестна или маловероятна, а следовательно, тратить на нее средства теперь не стоит.

Например, одна из организаций, с которой мне пришлось однажды сотрудничать, занималась букмекерской деятельностью через Интернет. Ее деятельность строго регулировалась законодательством, которое разрешало ей предлагать свою биржу ставок. Действия регуляторных органов во всем мире в какой-то степени непредсказуемы, особенно если речь идет об игорной деятельности, и поэтому очень трудно предугадать, когда и можно ли вообще предложить продукт в рамках конкретного законодательства. Для работы в такой среде нужно быть готовым к возникающим возможностям, поскольку регуляторные нормы могут меняться вместе со сменой правящей в стране партии.

Одна из таких возможностей состояла в том, чтобы предоставить через Интернет биржу ставок на скачках в Калифорнии, где имеется немало ипподромов. Это давало очень выгодную возможность для внедрения такой биржи, если позволяли изменения в законодательстве, что данная организация и сделала, когда биржи ставок через Интернет были после многих лет запрета разрешены с мая 2012 года. Если бы эта организация имела привычку к стратегическому планированию только один раз в год в октябре, т.е. до изменений в законодательстве, то она упустила бы случай воспользоваться возникшей возможностью, если не желала рисковать созданием биржи ставок для рынка, который тогда еще не существовал и мог бы вообще не появиться.

Такие возникающие возможности нужно использовать очень быстро. В частности, выйти вторым на рынок бирж ставок через Интернет в Калифорнии означало бы завоевать лишь малую долю этого рынка или вообще ничего. Этот типичный случай иллюстрируется графиком на рис. 16.6, где экономическая ценность возникающей возможности быстро убывает во времени.

Если не действовать быстро, как только появится возможность, она может почти сразу утратить свою экономическую ценность и в конечном итоге стать неудачно с экономической точки зрения выбранным путем следования подобно очередному ежегодному мероприятию по стратегическому планированию. Если

же организация пользуется регулярно и часто корректируемым календарным планом для оценки возникающих возможностей (например, ежемесячным календарным планом), а также эффективным экономическим фильтром и мелкими, но частыми выпусками, ограничивающими объем незавершенных работ, то ей не придется долго рассматривать возникающую возможность.

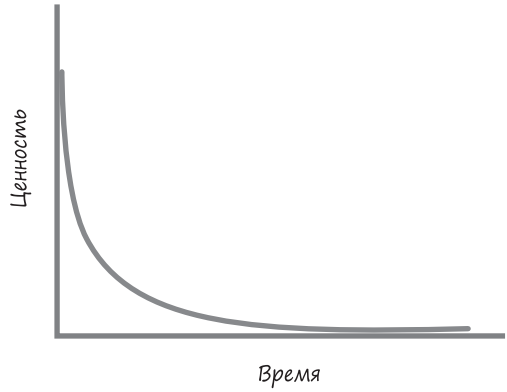


Рис. 16.6. Ценность многих возникающих возможностей быстро убывает

Планирование более мелких, но частых выпусков

Как обсуждалось в главе 14, экономические показатели более мелких, но частых выпусков весьма привлекательны. И как показано на рис. 14.4 и 14.5, а также в табл. 14.1, прибыли в течение срока службы продукта можно всегда увеличить, если разделить его на ряд более мелких постепенных выпусков.

Помимо этого значительного преимущества, имеется еще одна причина для ведения портфеля заказов с более мелкими, но частыми выпусками. Она состоит в том, чтобы избежать эффекта автоколонны (рис. 16.7).

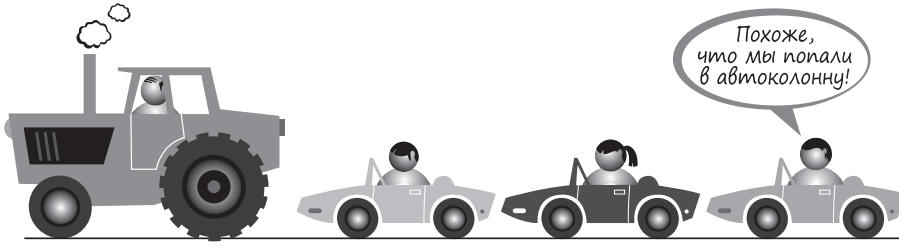


Рис. 16.7. Крупные продукты в заделе портфеля заказов создают эффект автоколонны

Что произойдет, если ехать по однополосной проселочной дороге за большим трактором, как на рис. 16.7? Скорее всего, за этим трактором соберется целая

автоколонна более мелкого автотранспорта, вынужденного долго ехать за ним на малой скорости. Причина, по которой образовалась такая автоколонна, очевидно, кроется в медленно движущемся крупном тракторе, перекрывающем всю дорогу по одной полосе (т.е. потребляющем весь общий дорожный ресурс).

Нечто подобное произойдет, если допустить появление крупных продуктов в заделе портфеля заказов. Крупные продукты потребляют немало ресурсов в течение длительного времени, а следовательно, эти ресурсы недоступны для многих других продуктов, выстроившихся в очередь за крупным продуктом. И пока они простаивают в очереди, каждый из них накапливает стоимость задержки. Если сложить стоимость задержки всех этих мелких продуктов и сравнить ее с привлекательными показателями более мелких, но частых выпусков, то станет ясно, что крупные продукты наносят значительный экономический ущерб, сокращая прибыли в течение срока эксплуатации.

В качестве меры борьбы с этим явлением в некоторых организациях устанавливаются правила для соблюдения размеров портфеля заказов на время его планирования, чтобы тем самым ограничить объемы проектных работ, предусматриваемых для каждого продукта. В моей практике, например, встречалось такое ограничение: ни одни из проектных работ не должны продолжаться больше девяти месяцев. Если же вносилось предложение для проведения более крупных проектных работ, то оно в целом отклонялось, а тем, кто его внес, предлагалось найти способ доставлять продукт более мелкими, но частыми выпусками.

Мне приходилось также работать с организациями, в которых было принято следующее правило: “Мы вообще не можем допустить второй выпуск любого продукта”. Это правило полностью противоречит стратегии более мелких, но частых выпусков. Если вообще не допускать второй выпуск, то первый выпуск, естественно, должен вместить все, что требуется от продукта, плюс все, что может однажды потребоваться. В таком случае для разработки продукта не только потребуются более крупные проектные работы, но и, вероятнее всего, возникнет задержка в выпуске очень ценных функциональных средств других продуктов на время работы над малоценными функциональными средствами крупного продукта. Такая стратегия планирования экономически ущербна. Тем организациям, которые ее применяют, нужно осознать, что последующие выпуски могут и должны делаться, исходя из их собственных экономических достоинств, и что планирования, при котором допускается только один выпуск, следует всячески избегать.

Стратегии управления оттоком продуктов

Стратегии управления оттоком продуктов помогают организациям выбрать удобный момент для извлечения продукта из задела портфеля заказов. В этом разделе описываются следующие три стратегии.

- Акцент на простоях в работе, а не на простаивающих работниках.
- Установление предела для незавершенных работ.
- Ожидание полного задействования команд.

Акцент на простоях в работе, а не на простаивающих работниках

Главная стратегия определения удобного момента для извлечения продукта из задела портфеля заказов состоит в том, чтобы не забывать о следующем принципе, упоминавшемся в главе 3: акцент следует делать на простоях в работе, а не на простаивающих работниках. Согласно этому принципу простои в работе намного более расточительны и экономически ущербны, чем простаивающие работники. И этот принцип противоречит тому, как во многих организациях ведется портфель заказов.

Типичная, но ошибочная стратегия выпуска продуктов заключается в следующем.

1. Извлечь самый верхний продукт из задела портфеля заказов и назначить людей для работы над ним.
2. Все ли люди заняты полностью, т.е. работают с полной самоотдачей? Если нет, то повторить п.1.

При такой стратегии все работники полностью заняты. Но в то же время работа над каждым продуктом продвигается медленно и чревата ошибками. Более совершенная стратегия заключается в том, чтобы начать работать над продуктом только тогда, когда можно обеспечить плавный ход работ над новым продуктом, который не прерывает ход работ над другими внутрипроцессными продуктами. Такая стратегия применяется в тесной связи со стратегией, рассматриваемой в следующем разделе.

Установление предела для незавершенных работ

Рассмотрим следующий пример. Приходилось ли вам, придя в ресторан, найти свободные столики, ни за один из которых обслуживающий персонал не захотел вас посадить? Если приходилось, то вам знакомо разочарование, которое при этом постигает посетителей ресторана. В подобной ситуации у вас может возникнуть вопрос: “Почему они не хотят меня обслуживать? Ведь у них есть свободные столики. Они не желают мной заниматься?”

Допустим, что в этот день несколько официантов заболели и не вышли на работу. В таком случае сообразительный владелец ресторана не должен вас

обслуживать. А что произойдет, если он все-таки на это пойдет? Вероятно, вам придется ждать 45 минут, прежде чем официант подойдет к вашему столику. Не знаю, как вы, но я не стал бы ждать целых 45 минут, прежде чем на меня обратят внимание! Я бы предпочел, чтобы они сразу сказали мне: "Извините, сэр, но четверо из наших официантов не вышли сегодня на работу по болезни, и поэтому вам придется ждать 45 минут, прежде чем вас обслужат". По крайней мере, эта информация даст мне возможность для выбора: ждать или пойти пообедать в другое место.

Было бы еще хуже, если бы они попытались посадить меня за обслуживаемый столик, а затем обслужить меня. Если бы они это сделали, то обслуживание всех остальных посетителей намного усложнилось бы. И если бы за столиком сидело слишком много посетителей, то обслужить их одному официанту было бы намного труднее, а следовательно, всем официантам пришлось бы работать слишком напряженно, что сказалось бы на общем настроении всех присутствующих. Именно поэтому сообразительный владелец ресторана не рассадит посетителей сверх возможностей их обслуживания официантами.

Если бы мы следовали примеру сообразительного владельца ресторана во время планирования портфеля заказов, то никогда не извлекали бы из задела портфеля заказов больше продуктов, чем мы способны завершить. Ведь это сокращает ресурсы, доступные для каждого продукта, задерживая его выпуск, а также отрицательно сказывается на качестве работы над всеми продуктами. А делать работу медленнее и с пониженным качеством вряд ли можно считать удачной стратегией.

Так как же установить подходящий предел для незавершенных работ? Как обсуждалось в главе 11, каждая команда представляет собой единицу производительности, которой следует пользоваться для установления предела для незавершенных работ. Зная, сколько имеется Scrum-команд и над какими видами продуктов они способны работать, мы можем выяснить, сколько и какие виды проектных работ над продуктами следует вести одновременно (рис. 16.8).

Как показано на рис. 16.8, *слева*, имеются три команды, способные работать над продуктами типа I, а также две команды, способные работать над продуктами типа II. Этой информации достаточно, чтобы приступить к установлению максимального количества продуктов каждого типа, над которыми организация способна работать одновременно. Только представьте, насколько труднее было бы определить подходящее количество одновременно выполняемых проектных работ, используя информацию о людях с конкретными профессиональными навыками, как показано на рис. 16.8, *справа*.

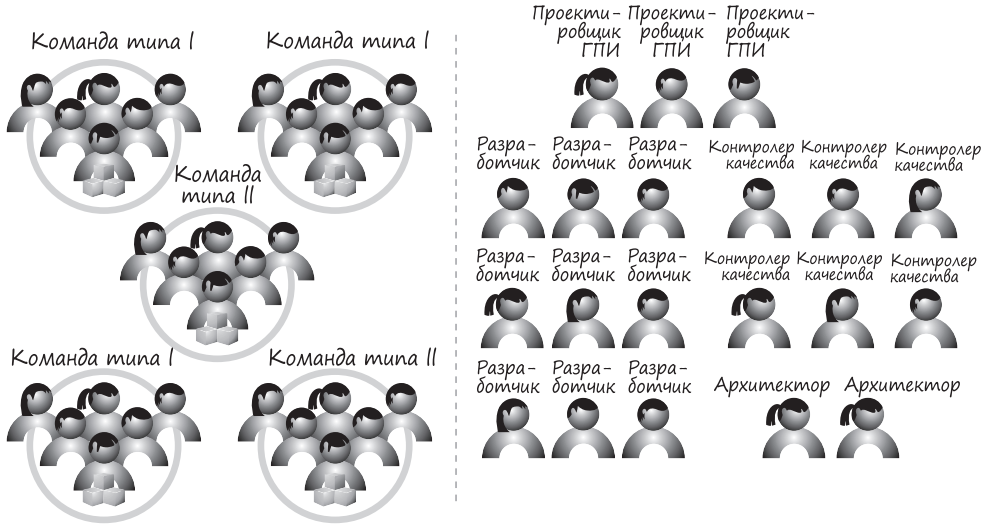


Рис. 16.8. Команды служат единицами производительности для установления предела незавершенных работ

Ожидание полного задействования команд

И последняя стратегия управления оттоком продуктов состоит в том, чтобы ожидать до тех пор, пока вся Scrum-команда не будет готова приступить к работе над продуктом. Организации, нарушающие принцип сосредоточения основного внимания на простоях в работе, а не на простаивающих работниках, нередко начинают работать над продуктом, когда для этого имеется только два свободных работника. При этом они руководствуются следующим соображением: "Двух разработчиков, пожалуй, недостаточно, но пусть они хотя бы начнут работу над этим очередным продуктом". Такая стратегия ошибочна потому, что создает еще больше препятствий в работе над другими продуктами, замедляя выпуск всех продуктов в целом и значительно увеличивая стоимость задержки.

Единицей производительности в Scrum считается команда, и поэтому работу над очередным продуктом не следует начинать, если к этому не готова вся Scrum-команда. И этого не стоит делать с точки зрения самой методики Scrum. Ведь в неполном составе Scrum-команда просто не в состоянии довести функциональные средства до состояния готовности.

Отклоняться от данной стратегии приходится в тех случаях, когда для работы над продуктами требуется несколько Scrum-команд. Если к такой работе готова только одна полная Scrum-команда, то следует решить, стоит ли начинать разработку с одной такой командой, или лучше подождать до тех пор, пока не освободятся все остальные требующиеся для этого команды.

Стратегии управления внутрипроцессными продуктами

Стратегии для управления внутрипроцессными продуктами помогают решить, когда следует сохранить, резко изменить, выпустить или прекратить работу над продуктом, находящимся в работе. Подобные решения обычно принимаются через регулярные промежутки времени (например, в конце каждого спринта), хотя из этого правила приходится иногда делать исключения, когда возникают непредвиденные обстоятельства, требующие возврата к внутрипроцессным продуктам.

К данной категории относятся самые разные стратегии, и руководство каждой организации, безусловно, установило свои нормативы для обращения с внутрипроцессным продуктами. Поэтому далее будет рассмотрена лишь одна стратегия граничных экономических показателей. Это должна быть всеохватывающая стратегия, помогающая в принятии решений и вполне согласующаяся с основными рассматриваемыми в данной книге принципами гибкой разработки вообще и методики Scrum в частности.

Применение граничных экономических показателей

С экономической точки зрения вся работа, выполненная над продуктом до момента принятия решения, считается необратимыми затратами. Нас интересуют только граничные экономические показатели, чтобы сделать следующий шаг. Делая этот шаг, мы должны решить, насколько оправданным будут очередные расходы денежных средств с точки зрения окупаемости инвестиций. Самое трудное — принять такое решение, не обременяя себя уже израсходованными денежными средствами.

Используя граничные экономические показатели, мы можем решить, что делать дальше с продуктами, разрабатываемыми в настоящее время. С этой целью мы тщательно рассматриваем каждый продукт под линзой граничных показателей, выбирая один из следующих основных вариантов.

- Сохранить — продолжить разработку продукта.
- Выпустить — завершить работу над продуктом и доставить его.
- Резко сменить стратегию — изменить направление разработки на основании полученных сведений.
- Прекратить — прервать работу над продуктом и уничтожить его.

На рис. 16.9 показан ход принятия решения при выборе одного из перечисленных выше четырех вариантов.

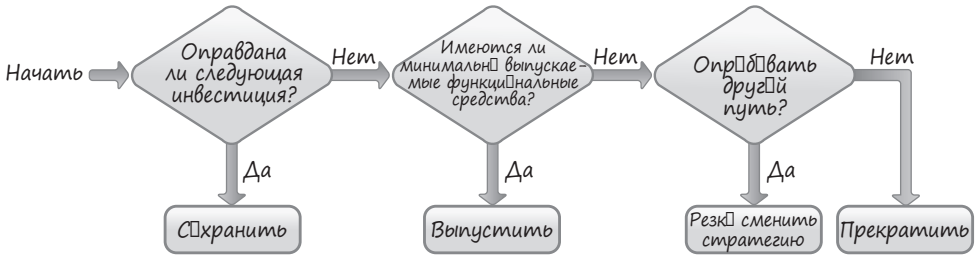


Рис. 16.9. Ход принятия решения в отношении внутрипроцессного продукта на основании граничных экономических показателей

Если очередные инвестиции в текущий продукт экономически оправданы, то, скорее всего, следует выбрать его *сохранение*. В этом случае мы анализируем внутрипроцессный продукт и приходим к выводу, что должны продолжить дальше вкладывать денежные средства в его разработку. Если же дальнейшие инвестиции в продукт экономически не оправданы, мы должны решить, что делать дальше: выпускать, резко менять стратегию или прекращать разработку продукта.

Если созданный до сих пор продукт содержит минимально выпускаемые функциональные средства (MRF), мы можем решиться на *выпуск* такого продукта. В противном случае мы идем по неверному пути и можем резко сменить стратегию, чтобы выбрать другой путь. Такой вариант, вероятнее всего, повлечет за собой возврат к стадии выработки общего замысла, чтобы рассмотреть новый путь (см. главу 17). А если дальнейшая разработка не оправдана и нас не устраивает ни сложившееся положение дел, ни перспектива резкой смены стратегии, то лучше всего выбрать *прекращение* разработки продукта.

Если же пренебречь граничными экономическими показателями, то безответственное поведение окажется просто неизбежным. В этом случае возникает следующий вопрос: “Если ваша организация уже потратила денежные средства на разработку продукта, то при каких обстоятельствах она может быть прекращена?” Просто удивительно, сколько раз мне приходилось слышать от отвечавших на этот вопрос, что их организации вообще (или крайне редко) уничтожают продукт после того, как в его разработку были вложены денежные средства. Похоже, что их стратегия — отдал пенни, придется отдать и фунт.

Меня просто поразило, как в одной организации объяснили причины, по которым они не прекращают разработку продуктов. Я спросил ее руководителей: “Допустим, что вы начинаете работать над продуктом, который считается ценным для 100% ваших потребителей, и его разработка обойдется в 1 млн долларов. Потратив эти денежные средства на разработку данного продукта, вы неожиданно узнаете, что он окажется ценным только для 10% ваших потребителей,

а общие затраты на его разработку возрастут до 10 млн долларов. Потратите ли вы еще 9 млн долларов, чтобы завершить продукт?" В ответ я услышал: "Да, потратим". Я отреагировал на это так: "Я не вижу в этом никакого смысла! Ведь соотношение издержек и прибыли от такого продукта изменится в 100 раз. Зачем же это делать?" На этот вопрос они ответили следующее: "Вы не понимаете, как ведется учет. Если мы уничтожим продукт, потратив на него 1 млн долларов до ввода системы в эксплуатацию, то бюджет отдела информационных технологий сократится на 1 млн долларов. А если мы потратим еще 9 млн долларов и введем систему в эксплуатацию, то полная стоимость системы будет перенесена в структурное подразделение, где затраты можно отнести на счет капитала". Очевидно, что в данном примере система ведения учета просто побила всякий здравый смысл.

Граничные экономические показатели являются эффективными средствами для принятия правильных решений и выявления признаков безответственного и расточительного поведения. Они должны стать вашей главной стратегией, когда вам придется решать, что делать дальше с разработкой внутрипроцессных продуктов.

Заключение

В этой главе были рассмотрены 11 важных стратегий планирования (т.е. ведения) портфеля заказов. Цель этой главы — предоставить на выбор читателя наиболее подходящие для него стратегии. Все 11 стратегий подкрепляют друг друга. Наибольшую пользу можно извлечь, применяя их все вместе. Но если по какой-нибудь причине придется воспользоваться только одной стратегией из каждой категории, то основное внимание следует уделить стоимости задержки, более мелким, но частым выпускам, установлению предела на незавершенные работы и граничным экономическим показателям.

В следующей главе обсуждаются вопросы планирования продукта, т.е. выработки общего замысла. В результате этого процесса отбираются продукты для дальнейшего рассмотрения на стадии планирования портфеля заказов.

Глава 17

Планирование продукта (выработка общего замысла)

Прежде чем начинать первый спринт для создания потребительской ценности, нужно сформировать первоначальный задел продукта. А для этого требуется составить концепцию продукта. Во многих организациях считают также полезным создавать предварительный график выпуска продукта, определяющий потенциальный ряд постепенных его выпусков. В вашей организации могут быть созданы и другие исходные артефакты. Создание таких артефактов в данной книге называется *выработкой общего замысла* или *планированием на уровне продукта*.

В этой главе описывается метод выработки общего замысла, согласующийся с принципами Scrum. И хотя этот метод очень удобен для организаций, стремящихся разрабатывать продукты по методике Scrum, тем не менее, его придется внедрять в процесс предварительного утверждения, который не является гибким.

Краткий обзор

Допустим, что имеется захватывающая мысль разработать новый продукт или следующую версию уже существующего продукта. Цель выработки общего замысла — развить эту мысль, описав назначение потенциального продукта и составить приблизительный план его создания. В конце процесса выработки общего замысла должна сложиться достаточная уверенность в том, что этот замысел пригоден для рассмотрения на стадии планирования портфеля заказов (см. главу 16), когда решается, следует ли финансировать разработку на следующем более детализированном уровне.

Выработка общего замысла или планирование продукта в Scrum не следует путать со сложной церемонией составления положения о проекте. Применяя методику Scrum, мы не считаем, что можем (или должны попытаться) знать все подробности о продукте, прежде чем приступать к работе над ним. Тем не менее мы понимаем, что финансирование разработки продукта обычно не может продвигаться вперед без предварительного составления его общей концепции, т.е. тех подробностей, которых должно быть достаточно, чтобы уяснить требования заказчиков, определить состав функциональных средств, выработать на высоком уровне решение и хотя бы приблизительно подсчитать, во что обойдется разработка продукта.

Мы не тратим много времени или труда на выработку общего замысла потому, что нам нужно быстро перейти от стадии прогнозирования, когда нам кажется, что мы знаем требования заказчика и потенциальное решение, к стадии быстрой ответной реакции, когда выполняются спринты для создания потребительской ценности. Ведь только тогда, когда мы фактически приступим к реализации решения через непрерывный цикл итераций в сложной среде, мы проходим утвержденное обучение, исходя из той реальности, в которой разрабатываемый продукт должен существовать и развиваться.

Временны*е* рамки

Выработка общего замысла, которая считается планированием на уровне продукта, является непрерывным процессом, а не единовременным событием (рис. 17.1).

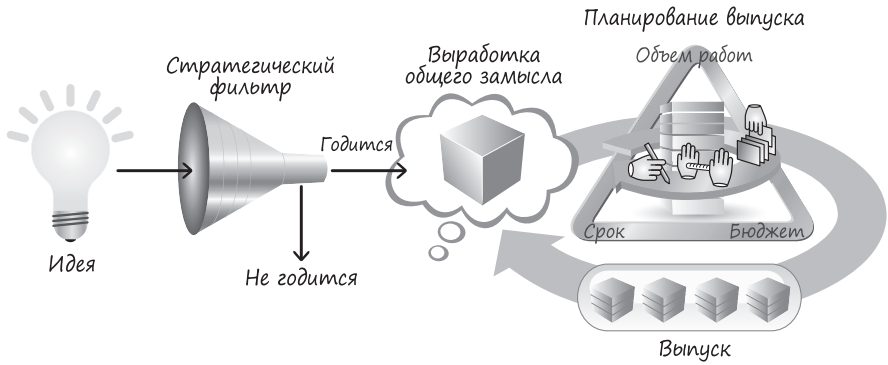


Рис. 17.1. Выработка общего замысла является непрерывным процессом

Выработка общего замысла начинается с идеи создать продукт, которая возникает у кого-нибудь одного или всей команды (этот процесс обычно называется *порождением идей*, или просто *мышлением*). Сначала возникшая идея проходит через *стратегический фильтр*, чтобы выяснить, насколько она согласуется со стратегическим направлением, выбранным в организации, а следовательно, стоит ли разрабатывать ее более углубленно и вкладывать в это дело средства.

Как только идея пройдет очищение стратегическим фильтром, можно приступать в первоначальной выработке общего замысла. В ходе этого процесса составляется общая концепция желательного перспективного продукта, чтобы определить, каким должен быть его минимальный первый выпуск. Это даст нам возможность быстро доставлять большую ценность при небольших затратах, а также передавать как можно быстрее в руки конкретных пользователей и заказчиков нечто ощутимое, оперативно получая ответную реакцию, чтобы подтвердить или опровергнуть наши предположения, сделанные в отношении целевых

потребителей, желательного набора функциональных средств или общего решения. С одной стороны, такая ответная реакция может совпасть с нашими ожиданиями, укрепив в нас желание сохранить текущую концепцию разрабатываемого продукта. А с другой стороны, она может совершенно не совпасть с нашими ожиданиями, и поэтому нам придется резко изменить свое первоначальное решение, пересмотреть свои действия и внести в наш план соответствующие коррективы.

Участники

Для первоначальной выработки общего замысла требуется участие только владельца продукта. Но, как правило, владелец продукта лишь надзирает за первоначальной выработкой общего замысла, привлекая к этому процессу одного или нескольких внутренних участников проекта, помогающих владельцу продукта, выработать общий замысел. Кроме того, в решении различных задач при выработке общего замысла нередко принимают участие специалисты в таких областях, как изучение конъюнктуры рынка, проектирование взаимодействия с пользователем и построение архитектуры систем. На рис. 17.2 приведены виды деятельности по выработке общего замысла (дополнительные участники данного процесса и артефакты обозначены пунктиром).

В идеальном случае Scrum-мастер и команда разработчиков, выполняющие спринты для создания потребительской ценности, могут также принимать участие в первоначальной выработке общего замысла, предоставляя ценную ответную реакцию на составленную концепцию продукта, а также исключая потребность передавать эту концепцию другой команде для разработки продукта. Но зачастую организация ожидает завершения процесса выработки первоначального замысла, чтобы профинансировать деятельность Scrum-команды, тем самым исключая ее участие в данном процессе. И как только разработка начнется и Scrum-команда будет в ней задействована полностью, вся Scrum-команда, включая владельца продукта, Scrum-мастера и разработчиков, должна принять участие в любых мероприятиях по пересмотру первоначально выработанного общего замысла.

Процесс

Главной предпосылкой для первоначальной выработки общего замысла служит идея, прошедшая очищение через стратегический фильтр, а главной предпосылкой для пересмотра первоначально выработанного общего замысла — резко измененная идея. Такая идея обновляется или пересматривается, исходя из ответной реакции пользователя или заказчика, изменений в финансировании проекта, непредсказуемых действий конкурентов на рынке и прочих важных перемен, которые происходят в той сложной среде, где должны существовать идеи.

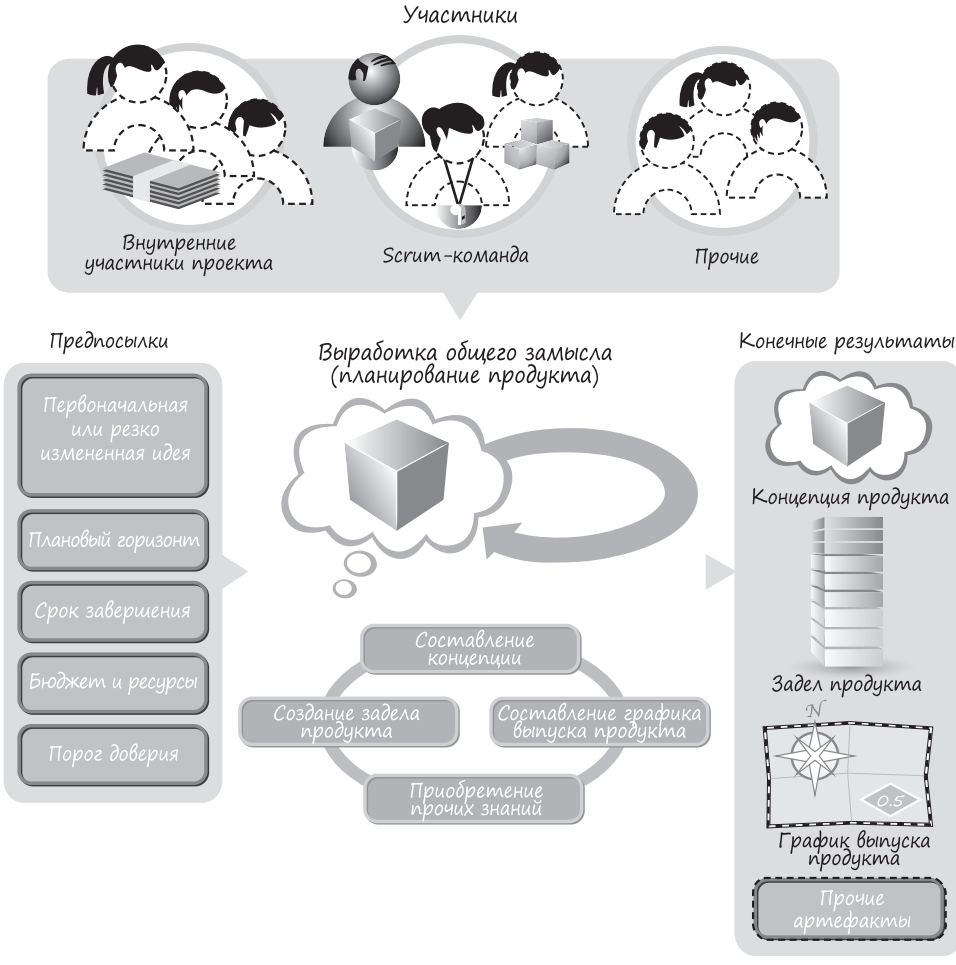


Рис. 17.2. Виды деятельности по выработке общего замысла

Для выработки общего замысла требуются и другие предпосылки. Прежде всего нужно наметить плановый горизонт — как далеко следует смотреть в перспективе, вырабатывая общий замысел. Необходимо также знать предполагаемый срок завершения проекта, если его можно вообще наметить, а также количество и виды ресурсов, доступных для выработки общего замысла. И наконец, нужно знать *порог доверия* — своего рода критерий готовности для выработки общего замысла. Такой порог устанавливается как ряд сведений, которые требуются для принятия с достаточной долей уверенности решений о целесообразности финансирования дальнейшей разработки. Подробнее о том, что составляет благоразумный порог доверия, речь пойдет далее в этой главе. И наконец, все предпосылки для выработки общего замысла, приведенные на рис. 17.2, должны быть рассмотрены одновременно, а не поочередно.

Сама выработка общего замысла состоит из самых разных видов деятельности, каждая из которых завершается важным конечным результатом или формированием первоначального задела продукта. Нередко при этом составляется также простой график выпуска продукта, наглядно показывающий постепенный характер проектных работ в виде последовательных выпусков. Вырабатывая общий замысел, мы осуществляем и другие виды деятельности, помогающие достичь намеченного порога доверия экономически оправданным способом.

Пример выработки общего замысла

В качестве примера, демонстрирующего процесс выработки общего замысла, рассмотрим идею разработки вымышленного продукта под названием **SmartReview4You** (Толковый обзор для вас), или просто SR4U. Допустим, что некая компания Review Everything, Inc. занимает лидирующее положение в области составления для Интернета обзоров потребительских продуктов и услуг. Ее главное дело — организовать форум для обмена обзорами потребительских продуктов и услуг. Доходы компании Review Everything росли за последние годы скромными темпами, хотя она и занимается выгодным делом. Но у этой компании имеется немало конкурентов, выпускающих новые функциональные средства с тревожной периодичностью. Поэтому компании Review Everything требуется новаторская служба, которая помогла бы ей обойти конкурентов.

В компании Review Everything сформировали отдельную маркетинговую команду для постоянного контроля над пространством социальных сетей, чтобы выяснить, какие потребители получают текущие услуги этой компании. Благодаря этому маркетинговой команде удалось выяснить, что многие пользователи теряют слишком много времени на веб-сайте компании Review Everything, чтобы отделить подлинные обзоры от подозрительных. Кроме того, многие пользователи жалуются на слишком большое количество обзоров некоторых продуктов (например, проигрывателей DVD) или услуг (в частности, качества обслуживания и кухни в китайском ресторане на главной улице их города), из-за чего им трудно составить общую картину по имеющимся обзорам.

В результате исследования этого сектора рынка возникла идея создать продукт SR4U как новаторское средство для выявления, фильтрации и отображения через Интернет обзоров, включающих обучающийся поисковый агент. В маркетинговой команде считают, что эту идею можно было бы воплотить в инновационной службе, предложить которую на рынке компания Review Everything уже давно искала возможность. С этой целью маркетинговая команда составила описание продукта SR4U на одну страницу, включив в него состав целевых функциональных средств высокого уровня, указав целевых потребителей и главные преимущества. Затем она отправила это описание в комитет по утверждению

новых продуктов, где оно было рассмотрено на совещании по новым идеям, которое регулярно проводится во вторую среду каждого месяца.

Высшее руководство компании, входящее в комитет по утверждению новых продуктов, согласилось с тем, что продукт SR4U дает компании Review Everything, Inc. серьезную возможность выделиться на рынке. И тогда этот комитет поручает Роджеру, представителю отдела стратегического маркетинга, стать владельцем продукта SR4U.

Руководство компании дало две недели на выработку общего замысла. За это время члены комитета по утверждению новых идей должны были проанализировать результаты выработки общего замысла и принять решение о целесообразности финансирования первоначальной разработки продукта SR4U. Помимо Роджера, руководство компании поручило двум экспертам в области фильтрации информации, исследователю рынка и ряду других заинтересованных лиц принять участие в выработке общего замысла. Но они не были уполномочены расходовать больше трудовых ресурсов Scrum-команды во время выработки общего замысла.

Роджеру предложили воспользоваться доступными ему ресурсами, чтобы получить следующие результаты.

- Первоначальная концепция продукта, задел и график выпуска продукта.
- Проверка достоверности исходного предположения о том, что пользователи предпочтут результаты, отфильтрованные средствами SR4U, неотфильтрованным результатам (далее в этой главе будет показано, как Роджер и его коллеги обеспечат такое утвержденное обучение).
- Описание других важных предположений (гипотез) о потенциальных пользователях и функциональных средствах, которые предполагается проверить в первом выпуске продукта.
- Ряд главных, намечаемых мероприятий с целью проверить другие предположения и выяснить, насколько первоначальный выпуск продукта SR4U оправдывает возлагаемые на него надежды.
- Перечень вопросов (известных неизвестных), которые требуется решить.

Без этой информации у высшего руководства компании не будет достаточного порога доверия, чтобы принять осознанное решение, следует ли продолжать первоначальную разработку.

Составление концепции продукта

Роджер и остальные участники проекта должны прежде всего составить общую убедительную концепцию продукта SR4U. Такую концепцию в Scrum не принято составлять в виде сложного документа на несколько сотен страниц. Если столько места требуется для описания общей концепции продукта, то его

назначение вряд ли станет от этого более понятным. Концепция даже сложного продукта должна быть сформулирована просто и задавать ясное направление тем, кто должен его реализовывать. Обратимся в качестве примера к представлению президента Джона Кеннеди о полете на Луну: “Я считаю, что эта нация должна посвятить себя достижению данной цели еще до конца нынешнего десятилетия, высадив человека на Луну с благополучным возвратом на Землю” [Kennedy, John Fitzgerald. 1961]. Кеннеди понадобилось 25 слов, чтобы выразить решительное и ясное представление о том, что для достижения намеченной цели в конечном итоге потребуются совместные усилия многих тысяч людей, создающих сложные системы из сотен тысяч связанных вместе компонентов.

При разработке продукции или услуг концепция нередко выражается исходя из того, как участники проекта собираются получить ценность. Примерами тому могут служить участки достижения ценности из категорий, приведенных на рис. 17.3.

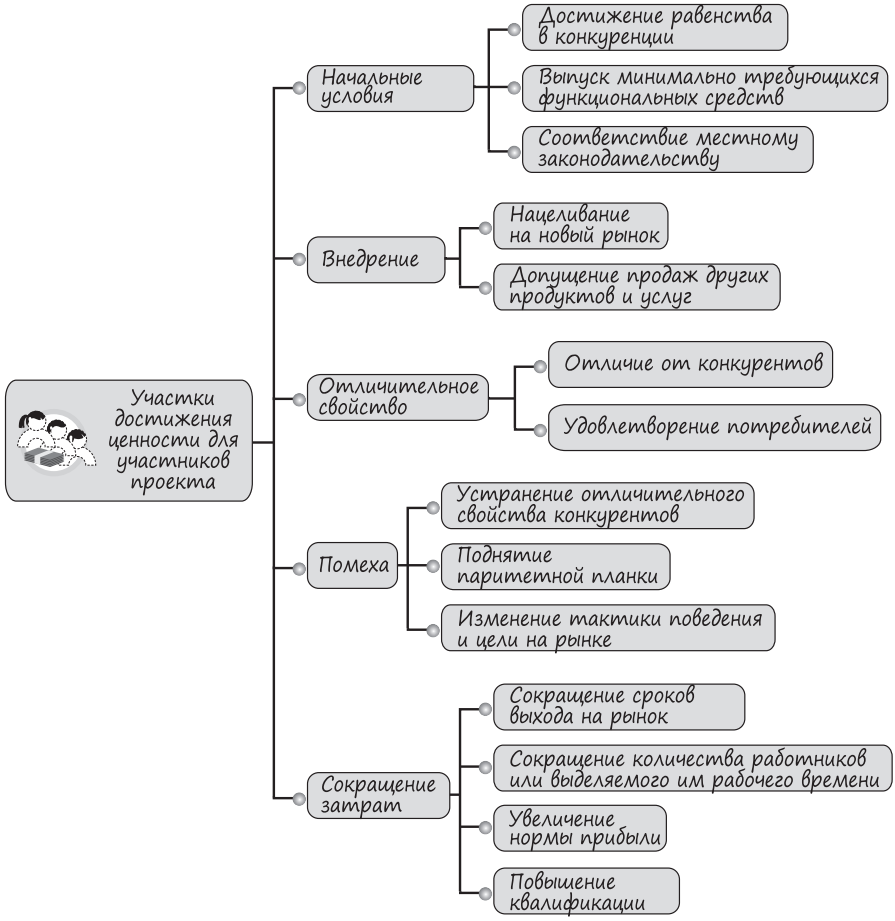


Рис. 17.3. Участки достижения ценности для участников проекта

Форма концепции может быть самой разной: от заявления в духе Кеннеди до вымышленного журнального обозрения. Некоторые примеры распространенных форм концепций продукции или услуг приведены в табл. 17.1, составленной по материалам, взятым из [Highsmith, Jim. 2009]. Можете выбрать ту форму, которая больше всего подходит вашей организации, группе для выработки общего замысла или идее.

Таблица 17.1. Наиболее распространенные формы концепций

Форма	Описание
Презентация в лифте	Напишите краткое (на полминуты или минуту) представление концепции продукта. Представьте, что вы поднимаетесь в лифте с предпринимателем, готовым пойти на риск, и должны увлечь его концепцией своего продукта. Сумеете ли вы сделать это в течение краткой поездки в лифте?
Таблица данных о продукте	Составьте первоначальную таблицу данных о продукте. Постарайтесь вместить ее на лицевой стороне одностраничного маркетингового листка
Упаковка концепции продукта	Нарисуйте коробку, в которую хотели бы упаковать продукт для его поставки. Сумеете ли вы найти три или четыре точки излома на коробке для иллюстрации? (Начертить 15 точек проще, чем 3 или 4.)
Слайды для пользовательской конференции	Создайте два или три презентационных слайда для представления своего продукта на пользовательской (или аналогичной) конференции. Постарайтесь избежать выделения жирными точками отдельных пунктов на своих слайдах
Информационный бюллетень	Составьте информационный бюллетень, чтобы опубликовать его, как только продукт будет готов. Грамотно составленные информационные бюллетени ясно сообщают все самое важное на одной странице, а то и меньше
Журнальное обозрение	Напишите вымышленное журнальное обозрение от имени обозревателя программных решений в самом популярном в вашей отрасли специализированном издании

Роджер и другие участники проекта из компании Review Everything решили выбрать форму информационного бюллетеня для описания концепции продукта SR4U. И начали они с обозначения участков достижения той ценности для участников проекта, которую должен доставлять продукт (см. рис. 17.3). Соответствующие участки описаны в табл. 17.2.

Таблица 17.2. Потенциальные участки достижения ценности продукта SR4U

Участок	Описание
Сокращение затрат/ Экономия времени	Продукт SR4U должен существенно экономить время, затрачиваемое пользователями на поиск обзоров
Отличительное свойство/ Удовлетворенность потребителей	Продукт SR4U должен произвести неизгладимое впечатление на пользователей. У них должно возникнуть ощущение, что данная служба замечательно справляется со своей задачей, помогая им делать покупки обоснованно
Помеха/Поднятие паритетной планки	Продукт SR4U должен вызвать смятение в рядах конкурентов. Их текущие программные решения должны сразу же стать устаревшими в сравнении с ним. Продукт SR4U должен установить для обзорных служб в Интернете новый исходный уровень, на который остальным придется подняться, чтобы соответствовать

Исходя из этих участков достижения ценности для участников проекта Роджер и другие участники проекта составили следующий информационный бюллетень (формулировку общей концепции).

- Сегодня компания Review Everything, Inc. объявила об успешном внедрении своей службы SmartReview4You. Эта служба предоставляет всем пользователям свой обучающийся агент для поиска в Интернете непредвзятой, подходящей информации о продукте или услуге.
- Дорис Джонсон как активный пользователь обзоров в Интернете так прокомментировала возможности данной службы: “Теперь у меня имеется свой личный помощник, имитирующий мой метод поиска и отбора обзоров в Интернете. Просто замечательно, что я сначала обучаю службу SmartReview4U тому, что мне нравится и что не нравится в обзорах, а затем она находит в Интернете объективные обзоры продуктов или услуг, автоматически отсеивая предвзятые или фиктивные. И то, что мне приходилось искать бесконечно долго, теперь находится с молниеносной скоростью. Эта служба экономит массу времени!”
- Председатель правления компании Review Everything К. Дж. Роллингз заявил: “Мы рады предложить первую в мире по-настоящему интеллектуальную службу. С момента зарождения Интернета люди стали эффективно пользоваться коллективной мудростью всей подключенной к Сети толпы. Но толпа иногда ведет себя слишком шумно, что затрудняет отделять зерна от плевел. Наша суперинтеллектуальная служба берет на себя всю неблагодарную работу по просеиванию огромных объемов данных обзоров в Интернете, отделяя подозрительные обзоры и возвращая только

подходящие. В итоге вы читаете только отобранные вами обзоры, на самостоятельные поиски которых вам раньше требовался не один час".

- Новая служба SmartReview4You свободно доступна на веб-сайте по адресу **`www.smartreview4you.com`**.

Создание задела продукта на высоком уровне

Как только будет составлена концепция продукта, можно приступать к созданию элементов задела продукта на высоком уровне. Имеется немало способов представить элементы задела продукта, но я лично предпочитаю пользовательские истории, подробно рассматривавшиеся в главе 5. Представляя элементы задела продукта в виде пользовательских историй, на стадии выработки общего замысла нужно написать эпические, т.е. очень крупные истории, согласующиеся с планированием на уровне продукта. Такие эпические истории должны соответствовать составленной концепции продукта и обеспечивать следующий уровень его детализации для высшего руководства и Scrum-команды.

Подобные истории обычно пишут те, кто составлял концепцию продукта, т.е. его владелец, участники проекта, а желательно — Scrum-мастер и команда разработчиков. Как правило, я привлекаю к написанию подобных историй всех членов своей Scrum-команды. Но, как упоминалось ранее, если разработка продукта еще не утверждена и не финансируется, то на стадии выработки общего замысла может быть задействована не вся Scrum-команда. В подобных случаях владелец продукта может обратиться к некоторым техническим работникам, проявляющим интерес к предметной области разрабатываемого продукта, за помощью в написании историй.

Продукт SR4U еще не утвержден в компании Review Everything, и команда разработчиков для него не назначена. Поэтому Роджер и участники проекта попросили опытного архитектора Иветт присоединиться к ним для написания историй в режиме мозгового штурма. В ходе этого мероприятия они составили первоначальные эпические истории, в том числе следующие.

- Мне как типичному пользователю требуется обучить службу SR4U тем видам обзоров, которые она должна отсеивать, чтобы знать критерии для отсеивания обзоров от моего имени.
- Мне как типичному пользователю требуется простой, как в Google, интерфейс для запросов на поиск обзоров, чтобы не тратить много времени на описание того, что мне требуется.
- Мне как типичному пользователю требуется, чтобы служба SR4U контролировала появление в Интернете новых обзоров интересующих меня

продуктов или услуг, автоматически отбирала и сообщала о них мне, чтобы я не просил ее постоянно делать это.

- Мне как опытному пользователю требуется возможность сообщать службе SR4U об источниках для поиска от моего имени, чтобы не получать обзоры с тех сайтов, которые мне не по душе или которым я не доверяю.
- Мне как поставщику продукта требуется возможность демонстрировать на своем веб-сайте сводку обзоров, специально отобранных службой SR4U, чтобы потенциальные потребители могли сразу видеть реакцию рынка на данный продукт из такого доверенного источника, как служба SR4U.

Составление графика выпуска продукта

Итак, имея в своем распоряжении исходную первоначальную концепцию и задел продукта на высоком уровне, можно приступить к составлению первоначального графика выпуска продукта в виде последовательного ряда выпусков для реализации данной концепции частично или полностью. Можно также попытаться развертывать продукт постепенно, если это целесообразно, т.е. уделить внимание более мелким, но частым выпускам, предоставляя в них общее решение по частям до тех пор, пока оно не будет реализовано полностью. График выпуска продукта должен давать первоначальное общее представление о таком постепенном развертывании продукта. Безусловно, если планируется только один небольшой выпуск, то составлять график выпуска продукта не нужно.

Частые выпуски совсем не означают, что установлены слишком жесткие сроки, поскольку такие сроки нередко не соблюдаются. Напротив, каждый выпуск сосредоточен на небольшом ряде *минимально выпускаемых функциональных средств* (MRF), в отношении которых все участники проекта приходят к общему строго коллективному согласию. Минимально выпускаемые функциональные средства представляют наименьший набор обязательных функциональных средств, которые должны присутствовать в выпуске, чтобы удовлетворить ожидания потребителей в отношении ценности и качества продукта. Такой набор иногда еще называют *минимально жизнеспособным продуктом* (MVP) или *минимально пригодными для продажи функциональными средствами* (MMF). И хотя можно выбрать доставку не только минимально выпускаемых функциональных средств в отдельном выпуске, тем не менее заказчики могут не воспринять в достаточной степени ценность продукта, если ее будет доставлено еще меньше. Поэтому следует всегда определять минимальный набор функциональных средств.

В дополнение к минимально выпускаемым функциональным средствам в некоторых организациях применяется стратегия фиксированных периодических выпусков. Например, выпуски происходят ежеквартально, чтобы упростить график выпуска продукта (рис. 17.4).

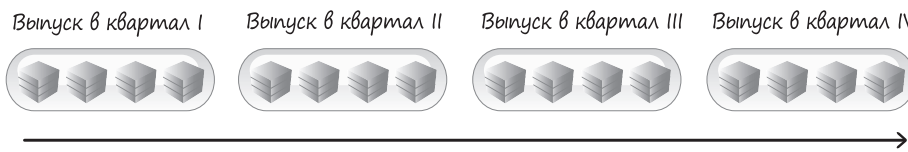


Рис. 17.4. Фиксированные периодические выпуски

У такого подхода имеется ряд преимуществ. Прежде всего, он хорошо понятен и предоставляет всем участникам проекта (как внутренним, так и внешним) предсказуемые выпуски продукта. Кроме того, он устанавливает размеренный темп выпусков, помогающий распределять ресурсы предсказуемым способом и позволяющий разрозненным группам согласовывать их планы.

Тем не менее, применяя такую стратегию, нужно все равно определять минимально выпускаемые функциональные средства для каждого выпуска. Если для разработки этих средств потребуется меньше времени, чем допускает фиксированное время выпуска, то могут быть созданы дополнительные ценные средства. И хотя фиксированные периодические выпуски могут быть не всегда приемлемы, если они зависят от внешних событий вроде конференции или фиксированного срока запуска совместно разрабатываемого продукта, преимущества данной стратегии не следует сбрасывать со счетов.

У каждого выпуска в графике выпуска продукта должна быть ясно намеченная цель, сообщающая назначение и желательные результаты выпуска. *Цель выпуска* намечается с учетом многих факторов, включая целевых потребителей, вопросы архитектуры, решаемые на высоком уровне, важные события на рынке и пр.

При составлении графика выпуска продукта нужно принимать во внимание круг потребителей и их разделение на разные категории по рынкам. График выпуска продукта должен выражать, как и когда следует удовлетворить эти категории потребителей. Если обратиться к примеру продукта SR4U, то его первоначальным рынком потребителей являются индивидуальные пользователи, которые заинтересованы в чтении полезных обзоров, прежде чем приобретать продукты или услуги. Поэтому команда, выработавшая общий замысел продукта SR4U, разделила далее этот сегмент рынка на категории типичных и опытных пользователей, которым требуется менее или более подробный контроль над работой службы SR4U соответственно. Эта команда решила, что первоначально данный продукт должен быть нацелен на типичных пользователей.

Кроме того, команда, выработавшая общий замысел продукта SR4U, может предусмотреть на будущее потребительскую базу поставщиков продуктов и услуг, которые будут пользоваться службой SR4U для предоставления на своих веб-сайтах предыстории непредвзятых обзоров их предложений. Но, прежде чем поставщики увидят ценность в продукте SR4U, чтобы платить за него, компании Review Everything, Inc. нужно упрочить его как надежное фирменное средство для накопления и фильтрации обзоров.

Составляя график выпуска продукта, следует также принимать во внимание архитектурные и технологические вопросы, решаемые на высоком уровне. Например, главным технологическим вопросом для разработки продукта SR4U является определение форм доступа к службе поиска обзоров. Поэтому команда решила предоставить сначала доступ к этой службе через веб-браузер. Но в дальнейшем она может также предусмотреть приложения для доступа к службе SR4U из мобильных устройств, работающих под iOS, Android и других операционных систем. А в долгосрочной перспективе команда намерена предоставить партнерам открытый прикладной программный интерфейс API для доступа к основным функциям SR4U.

Кроме того, при составлении графика выпуска продукта, возможно, придется допустить влияние на временны*е* рамки выпуска функциональных средств любых значительных событий на рынке. Например, компания Review Everything всегда принимает участие в ежегодной конференции Social Media Expo, посвященной социальным сетям. Роджер и участники проекта согласны, что было бы неплохо приурочить выпуск к этой ежегодной конференции, которая должна состояться через три месяца, чтобы получить ответную реакцию на данную службу.

Главная цель, преследуемая при составлении графика выпуска продукта, — принять во внимание любые факторы, способные оказать помощь в определении намечаемого ряда выпусков разрабатываемого продукта. Не следует, однако, забывать, что этот график лишь приближенно намечает один или больше ближайших по срокам выпусков. Поэтому у составителей графика выпуска продукта должно быть право обновлять его по мере поступления уточняющей информации.

Необходимо также рассмотреть дальнейшие перспективы развития графика выпуска продукта. Несмотря на то что концепция продукта может быть настолько крупной и смелой, что для ее полной реализации потребуется не один год, вряд ли стоит составлять подробный график выпуска продукта, полностью охватывающий такую концепцию. Применяя методику Scrum, мы составляем график выпуска продукта в перспективе настолько, насколько это благоразумно и желательно. Временны*е* рамки этого графика зависят от конкретных обстоятельств. Но он должен, как минимум, охватывать период времени, на который запрашивается финансирование проекта.

Роджер и участники проекта считают, что для полной реализации их концепции продукта SR4U потребуется несколько лет. Но в то же время Роджер считает, что было бы непрактично пытаться простирать график выпуска продукта так далеко, принимая во внимание низкий уровень утвержденного обучения и частоту перемен на рынке обзоров в Интернете. Поэтому он и участники проекта останавливают свой выбор на девятимесячном графике выпуска продукта, как показано на рис. 17.5.

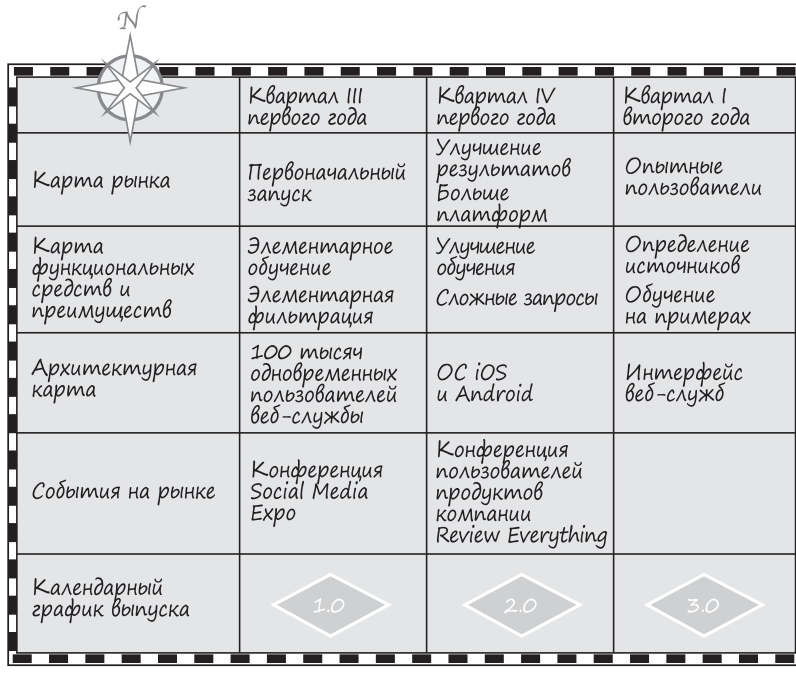


Рис. 17.5. График выпуска продукта SmartReview4You

Другие виды деятельности

Выработка общего замысла может включать в себя любые другие виды работ, которые, по мнению участников данного процесса, пригодны для достижения целевого порога доверия. Для этого, возможно, придется провести минимальное исследование рынка и намеченного круга потребителей и пользователей; быстрый конкурентный анализ предлагаемого продукта, сравнив его с другими предложениями на рынке; или же создать приблизительную коммерческую модель, которая должна помочь выяснить, проходит ли предлагаемый продукт экономический фильтр, установленный в организации.

В некоторых организациях могут даже решить провести стадию выработки общего замысла в течение нескольких спринтов. В подобных случаях для выработки общего замысла назначается отдельная Scrum-команда, ведущая задел работ, связанных с выработкой общего замысла, расставленных по приоритетам и выполняемых в течение коротких спринтов, длящихся не больше одной недели. Некоторые из этих спринтов могут также включать в себя виды деятельности по приобретению знаний, как пояснялось в главе 5. К примерам спринтов, посвященных приобретению знаний, можно отнести создание прототипа или опытного образца продукта или же архитектурного средства, имеющего решающее значение.

Что же касается продукта SR4U, то Роджер и его команда, включая и экспертов в предметной области, решили выполнить на стадии выработки общего замысла один спринт для приобретения знаний. Прежде чем вкладывать средства в разработку автоматизированной системы, Роджеру нужно провести простой сравнительный тест, чтобы подтвердить предположение, что обзоры, фильтруемые службой SR4U, действительно оказываются намного более полезными для пользователей, чем неотфильтрованные обзоры (рис. 17.6).

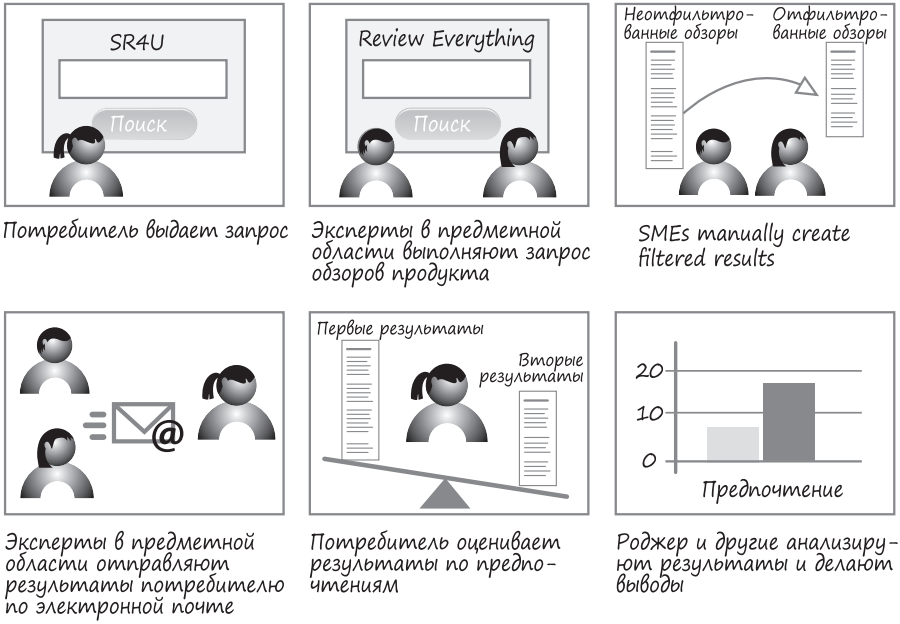


Рис. 17.6. Раскадровка спринта для приобретения знаний о продукте SR4U

В течение спринта по выработке общего замысла команда имитирует одну веб-страницу (простую HTML-страницу поиска в Google для службы SR4U), где небольшая образцовая группа пользователей может выдавать запросы на обзоры избранных продуктов или услуг и получать в ответ два ряда результатов. Первые результаты должны содержать неотфильтрованные обзоры, которые, как правило, возвращаются по запросу, а вторые результаты — отфильтрованные обзоры, среди которых отсутствуют подозрительные обзоры. При этом пользователям не сообщается, какие из этих обзоров отфильтрованы, а какие — нет.

Образцовой группе пользователей заранее сообщается, что результаты их запросов будут готовы на следующий день и отправлены электронной почтой, поскольку без их ведома Роджер не намерен на данном этапе разрабатывать технологию для автоматизации формирования отфильтрованных результатов по запросу. Вместо этого он предлагает двум экспертам в предметной области

выполнять фильтрацию обзоров вручную и предоставлять пользователям как отфильтрованные, так и неотфильтрованные результаты. Роджер и его команда собираются затем опросить всех членов образцовой группы пользователей, чтобы выяснить, какие именно результаты они предпочитают получать и почему.

Цель данного раннего теста — получить элементарное подтверждение ценности основного предположения, что пользователи будут удовлетворены обзорами, отфильтрованными службой SR4U. Если эксперты в предметной области не в состоянии сформировать вручную убедительные отфильтрованные результаты, то способность компании Review Everything создать продукт типа экспертной системы, доставляющий ценность на рынок, будет поставлена под серьезное сомнение.

Высшее руководство компании просит также Роджера описать другие основные предположения или гипотезы, которые еще не проверены на предмет потенциальных пользователей и функциональных средств, а также основные меры для проверки этих предположений. Он будет сотрудничать с представителями отдела маркетинга и другими заинтересованными лицами, чтобы завершить эту работу. Вместо обширного исследования рынка, отнимающего немало времени, Роджер планирует воспользоваться разработкой первого выпуска экспериментального инструментального средства, чтобы выяснить, что же люди на самом деле думают о продукте SR4U и какие функциональные средства они хотели бы получить в свое распоряжение.

Экономически обоснованная выработка общего замысла

Выработка общего замысла должна проводиться экономически обоснованно. Ее следует рассматривать в качестве инвестиций в приобретение сведений, необходимых руководству организации для принятия взвешенного решения, следует ли финансировать разработку продукта по его концепции. Если выработка общего замысла проведена недостаточно, то разработчики не будут готовы к первому спринту по созданию потребительской ценности. А если выработка общего замысла проведена чрезмерно, то возникнут большие запасы артефактов планирования продукта, которые, возможно, придется переделать или вообще отвергнуть, чтобы приступить к утвержденному обучению.

Во многих организациях процесс выработки общего замысла называется *составлением положения о проекте*, *зарождением проекта* или *основанием проекта*. А в некоторых организациях этот процесс является частью всесторонней модели управления закрытием этапов проектирования. Нередко составление положения о проекте в данном контексте превращается в процесс со сложными церемониями и обширным планированием на основании ряда прогнозируемых данных. Такие

формы неопределенных планов, основанных на подробных, но еще не проверенных данных, создают иллюзию полной определенности, когда приходится принимать решение о целесообразности финансирования проектных работ.

Кроме того, сложный метод проектирования по восходящей плохо согласуется с нисходящим процессом гибкой разработки по методике Scrum. Такая потеря соответствия сродни следующему заявлению: "Вы можете вести разработку по методике Scrum, но, прежде чем мы утвердим ее, нам все же потребуются такие же артефакты, как и всегда: обширные, заранее составленные требования, полностью сформированный бюджет и точный календарный план". При таком расхождении организации будет очень трудно извлечь в долгосрочной перспективе немалые выгоды из методики Scrum.

Применяя методику Scrum, мы сохраняем процесс выработки общего замысла как можно более простым. Для этого мы выполняем лишь достаточное предварительное планирование и приобретаем знания, исходя из характера продукта, масштабов и степени риска его разработки. Кроме того, мы допускаем подробную разработку ряда других артефактов в оперативном режиме. Наша цель — принять наилучшее из возможных решений теперь на основании пригодной информации, своевременно полученной из финансовых источников. В то же время мы признаем, что наши знания о продукте могут и должны измениться, как только мы создадим что-нибудь и представим заказчику или пользователю на тщательное рассмотрение. Для экономически обоснованной выработки общего замысла имеются полезные рекомендации, представленные на рис. 17.7.

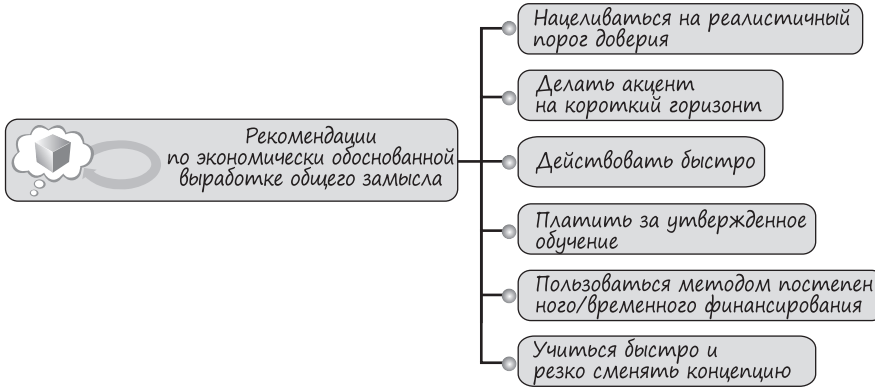


Рис. 17.7. Рекомендации по экономически обоснованной выработке общего замысла

Нацеливаться на реалистичный порог доверия

Порог доверия определяет минимальный уровень и тип информации, запрашиваемой теми, кто принимает решение, чтобы они с достаточной уверенностью принимали решение о целесообразности финансирования проектных работ

на следующем уровне. Порог доверия можно рассматривать в виде планки, которую нужно преодолеть, прежде чем завершить процесс выработки общего замысла и передать продукт для тщательного рассмотрения на стадии планирования портфеля заказов, где продукт пропускается через экономический фильтр с целью выяснить, насколько он соответствует критериям финансирования, принятым в организации. И если продукт им соответствует, то можно перейти к проверке достоверности основных предположений и разработке продукта. Высота такой планки имеет реальные экономические последствия (рис. 17.8).

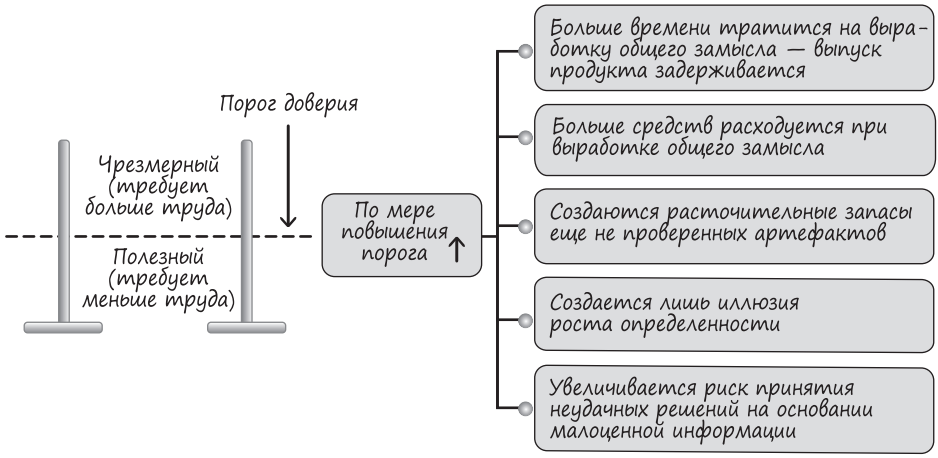


Рис. 17.8. Последствия установки слишком высокой планки порога доверия

Чем выше планка, тем труднее ее преодолеть. Дополнительное время, затрачиваемое на выработку общего замысла, скорее всего, приведет к задержке поставки продукта, а такая задержка имеет свою стоимость (см. главу 3). Кроме того, время, затрачиваемое на выработку общего замысла, должно окупиться, и поэтому чем выше планка, тем дороже будет ее преодолеть. К тому же более прогнозируемая работа создает дополнительные запасы незавершенных работ, которые могут легко стать расточительными, если изменятся обстоятельства. А ведь большая часть незавершенных работ еще не проверена (например, артефакты планирования, прогнозирующие то, что может произойти в будущем), и поэтому поднятие планки еще выше не вселяет больше уверенности в затрачиваемых усилиях. И наконец, дополнительные затраты труда могут увеличить риск принятия неудачного решения продолжить работу дальше из-за иллюзии определенности, которую может дать постоянно растущий ряд получаемых артефактов планирования. Увеличение количества артефактов совсем не означает большую определенность или лучшее решение о финансировании проектных работ.

Как упоминалось в главе 14, предварительное планирование должно быть полезным, если оно не чрезмерно, и поэтому нужно установить некоторый порог

на уровне его полезности, но не чрезмерности. Такой порог устанавливается исходя из особенностей конкретной организации. Так, в одних организациях решения удобнее принимать при весьма неопределенных условиях, тогда как в других для дальнейших действий требуется высокая степень определенности. По мере увеличения потребности в определенности требуется больше усилий для сбора данных и утвержденного обучения. Существует практический предел для утвержденного обучения до тех пор, пока не начнется разработка, построение чего-то конкретного и фактическая его проверка с участием намеченного круга пользователей. Поэтому, устанавливая такой предел, следует руководствоваться здравым смыслом, реалистично глядя на вещи. Кроме того, порог для выработки общего замысла следующего выпуска долговечной базовой системы, скорее всего, окажется ниже, чем для выработки общего замысла нового, в большой степени новаторского и потенциально дорогого продукта.

Компания Review Everything, Inc. отказалась от сложной процедуры предварительного планирования на уровне продукта. Комитет по утверждениям согласился с тем, чтобы порог доверия должен быть установлен "вполне приемлемым" и "явно достаточным", чтобы продолжить первоначальную разработку, в ходе которой компания сможет проверить достоверность предположений с помощью намеченного круга пользователей. Комитету по утверждениям не требуется полный план проекта, разработанный до уровня отдельных задач с указанием, кто и когда их должен выполнять. Напротив, ему требуется выяснить цель следующих проектных работ и каким образом Роджер собирается оценивать результаты, чтобы выбрать следующий наилучший план действий.

Делать акцент на короткий горизонт

Не следует пытаться сразу вырабатывать общий замысел слишком подробно, а прежде всего делать акцент на обязательных функциональных средствах как первых претендентах на выпуск. Если же мы собираемся делать акцент на очень широкий горизонт, то расточаем свое время, планируя мероприятия, которые могут вообще не состояться. А если мы разрабатываем новый, передовой продукт, то б*о*льшая часть наших предположений еще не проверена на достоверность. И, скорее всего, когда продукт будет передан для эксплуатации в неопределенной потребительской среде, мы узн*а*ем нечто важное, побуждающее нас адаптировать концепцию продукта и планы его разработки.

График выпуска продукта SR4U, составленный Роджером, спланирован на девять месяцев вперед, но на самом деле основной акцент в нем сделан на первом выпуске. Всем заинтересованным лицам известно, что до тех пор, пока они не получат службу поиска обзоров, которой потребители смогут пользоваться и комментировать ее работу, они будут только догадываться, какой именно

набор функциональных средств требуется этой службе. Таким образом, для того чтобы заглянуть слишком далеко в будущее, им придется основывать одни предположения на других, еще бо́льших предположениях, нарушая принцип выработки меньшего количества кратковременных, но важных предположений.

Действовать быстро

Процесс выработки общего замысла должен быть не долгим и растянутым, а быстрым и эффективным. Чем быстрее он завершится, тем скорее мы приступим к построению чего-нибудь ощутимого, чтобы проверить на нем правильность наших представлений и предположений.

Время, затраченное на выработку общего замысла, должно быть включено в расчет времени, требующегося для выпуска продукта. Рыночные часы начинают отсчет времени с момента, когда становится известной коммерческая возможность (т.е. возникает идея), и не останавливаются до тех пор, пока продукт не будет выпущен на рынок. Поэтому излишне долгая выработка общего замысла задерживает выпуск продукта, а такая задержка может обойтись очень дорого. Экономические показатели быстрых действий во время выработки общего замысла впечатляют. Как отмечают Смит и Райнерцен, это "дешевая" возможность сократить продолжительность цикла разработки [Smith, Preston G., and Donald G. Reinertsen. 1998].

Действовать быстро означает также содействовать неотложности принятия важного решения в отношении продукта. Такая неотложность помогает выявить нужные ресурсы и обязывает своевременно завершить процесс выработки общего замысла.

Чтобы дело продвигалось быстро, можно, в частности, сообщить команде, вырабатывающей общий замысел, срок завершения данного процесса как одну из его предпосылок. Далеко не каждая идея требует одинакового времени для выработки общего замысла. Как упоминалось ранее, идея разработать новый, передовой продукт может потребовать больше времени для выработки общего замысла, чем идея усовершенствовать или обновить уже давно существующий продукт. Но в любом случае требуется установить обоснованные временны́е рамки для процесса выработки общего замысла, чтобы быстро перейти к стадии проверки достоверности предположений по ответной реакции.

Что же касается продукта SR4U, то Роджеру и другим было отведено две недели на выработку общего замысла. Чтобы поспеть к этому сроку, Роджеру пришлось посвятить данной работе все свое время; экспертам в предметной области — половину своего рабочего времени в течение второй недели, когда они выполняли спринт по приобретению знаний о продукте, а специалисту по исследованию рынка — два дня на первой неделе.

Платить за утвержденное обучение

Результаты выработки общего замысла следует оценивать с экономической точки зрения, исходя из того, как они способствуют утвержденному обучению в отношении целевого потребителя, целевого набора функциональных средств или программного решения. Следует, однако, проявлять особую осторожность, выполняя прогнозирующие действия, дающие информацию с большой степенью неопределенности, т.е. такую информацию, которая считается достоверной, но на самом деле еще не проверена на достоверность с помощью намеченного круга потребителей или пользователей. Такие действия опираются на малоценную информацию и не только не окупаются, но и могут оказаться расточительными, если в результате утвержденного обучения придется пересмотреть или вообще отвергнуть весьма неопределенную информацию, которая оказывается неверной.

Кроме того, получение в большом количестве малоценной и весьма неопределенной информации может помешать правильному суждению и заставить нас считать, что мы понимаем сложившуюся ситуацию лучше, чем это есть на самом деле. В итоге мы принимаем важные решения под иллюзорным впечатлением полной определенности (рис. 17.9).

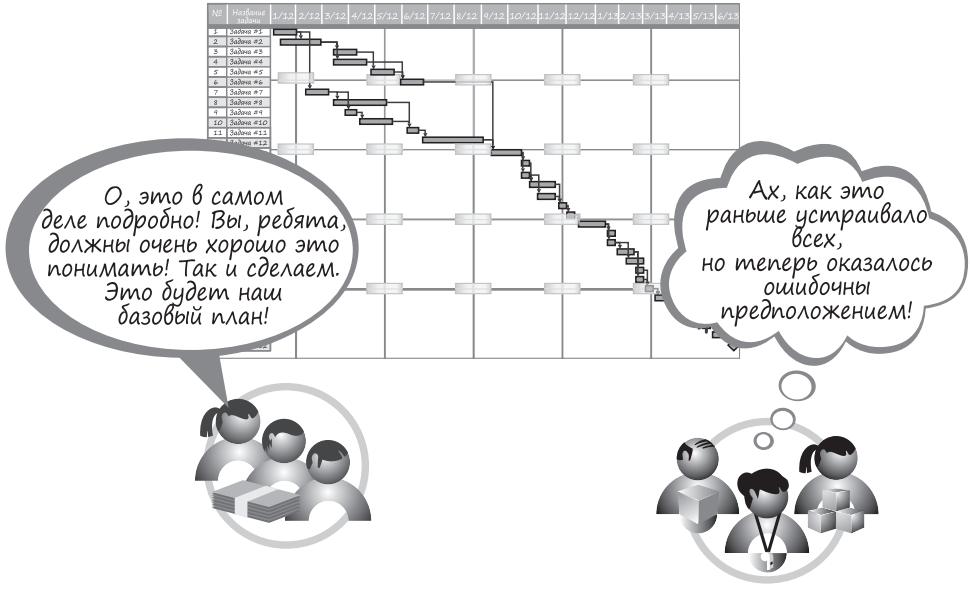


Рис. 17.9. Принятие решений под иллюзорным впечатлением полной определенности

В рассматриваемом здесь примере разработки продукта SR4U содержимое задела продукта и графика его выпуска представляет весьма неопределенную информацию. Роджер считает, что полученные им результаты служат лишь хорошей догадкой о том, что же действительно требуется пользователям, и только

отдаленно напоминает то, что они в конечном итоге получат. Но в то же время это содержимое подвержено изменениям по мере утвержденного обучения команды во время разработки. Поэтому Роджеру нужно быть очень осторожным в отношении степени детализации информации, получаемой на данном этапе.

В процессе выработки общего замысла продукта SR4U высшее руководство компании готово платить за утвержденное обучение в отношении основного предположения, что пользователи предпочитают отфильтрованные результаты поиска обзоров неотфильтрованным. Высшее руководство компании считает экономически обоснованным платить за подобную информацию в процессе выработки общего замысла, прежде чем они вложат намного больше средств в приобретение этой же информации впоследствии. Было бы намного менее обоснованно потратить сначала значительные средства на создание первой версии продукта SR4U, а затем обнаружить, что пользователи не особенно предпочитают отфильтрованные результаты поиска обзоров неотфильтрованным.

Пользоваться методом постепенного/временного финансирования

К финансированию разработки продукта следует всегда применять постепенный или временный подход (рис. 17.10).

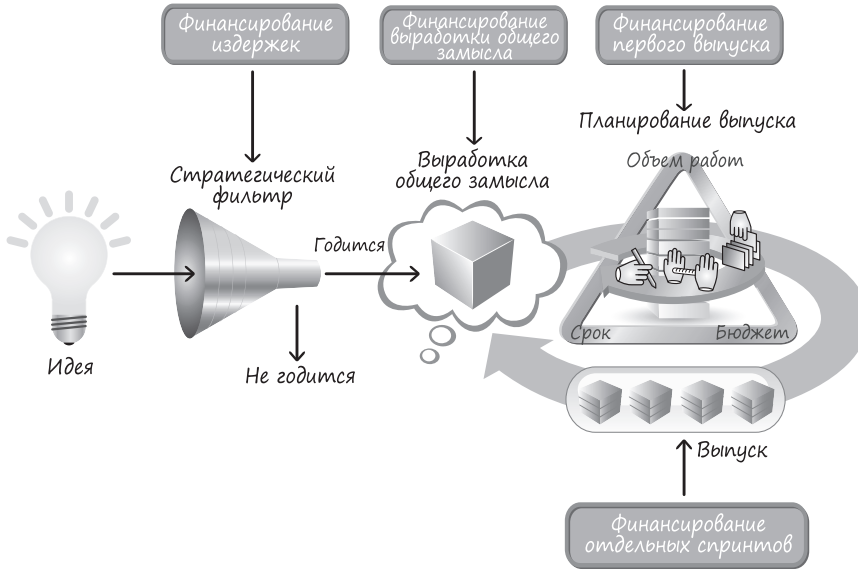


Рис. 17.10. Постепенное/временное финансирование

Решения о финансировании постоянно принимаются и пересматриваются по мере поступления более полной информации — по крайней мере, так должно

быть. При первой попытке выработать общий замысел не следует стараться добыть столько информации, сколько потребуется, чтобы утвердить и финансировать всю дальнейшую разработку продукта. Вместо этого лучше добыть столько информации, сколько должно быть достаточно, чтобы профинансировать часть проектных работ для получения следующих важных, а самое главное — истинных знаний и отзывов, касающихся потенциальных потребителей, функциональных средств или метода разработки программного решения.

Применяя метод постепенного финансирования, мы сначала выделяем средства лишь на небольшую часть проектных работ, а затем пересматриваем решение, когда оплачено утвержденное обучение, предоставляющее важную информацию для принятия такого решения. При постепенном финансировании мы можем сократить объем работ по выработке общего замысла и время на их завершение.

Не следует также забывать, что выделение средств совсем не означает, что мы собираемся их потратить. По мере поступления ответной реакции из отдельных спринтов мы можем резко сменить первоначальную концепцию продукта на новую или просто прекратить проектные работы над продуктом (подробнее об этом см. в главе 16).

Что же касается продукта SR4U, то в компании Review Everything принят гибкий подход к финансированию не крупными порциями, а в количестве, достаточном для проверки достоверности следующего важного предположения. Опираясь на ответную реакцию и утвержденное обучение, высшее руководство компании может продолжить тратить уже выделенные средства, выделить дополнительные средства или прекратить финансирование дальнейшей разработки.

Учиться быстро и резко сменять концепцию

Выработка общего замысла является частью цикла быстрого обучения и резкой смены концепции. Такой метод иногда еще называют вычурно *быстрым провалом*. Проще говоря, мы благоразумно и эффективно управляем своими ресурсами, чтобы быстро и дешево выработать общий замысел. Затем мы также быстро и дешево проверяем достоверность наших знаний и предположений о потенциальных потребителях, функциональных средствах и продукте, чтобы выяснить, насколько концепция и планы разработки продукта отвечают нашим коммерческим ожиданиям. Если мы узнаем, что они не отвечают им, то резко и быстро меняем концепцию продукта на более подходящую или просто уничтожаем продукт и прекращаем дальнейшее расходование средств на него.

Мы можем существенно сократить финансовые расходы, если готовы сначала принять взвешенное решение на основании доступной информации, а затем изменить направление поисков нужного решения или прекратить работу над продукта, если его концепция окажется неверной. Как правило, быстро начинать

работу над продуктом и столь же быстро узнавать, что мы не правы, оказывается намного дешевле, чем тратить поначалу немало времени и средств на принятие "верного" решения лишь для того, чтобы в конечном итоге выяснить, что мы все-таки ошиблись. Такой метод быстрого провала приносит пользу лишь в том случае, если мы готовы уничтожить продукт, как только начнем тратить средства на его разработку (см. обсуждение граничных экономических показателей в главе 16).

Что же касается рассматриваемого здесь продукта SR4U, то в данном случае преследуется цель очень быстро (и экономно) получить ответную реакцию, создав первоначальную версию возможностей службы для обучения и фильтрации обзоров, чтобы люди смогли начать пользоваться данной службой. Если после скоро полученной ответной реакции команда узнает, что отфильтрованные результаты поиска обзоров не считаются целевым кругом пользователей намного лучшими, чем неотфильтрованные результаты, то компания может выделить больше времени на попытку усовершенствовать алгоритм фильтрации. Но если после вложения достаточных средств компания по-прежнему не в состоянии получить ощутимо лучшие результаты фильтрации обзоров, то, возможно, настает переломный момент, когда нужно решить, прекратить ли дальнейшую работу над продуктом или выбрать другое направление, чтобы продолжить в нем эффективное и быстрое обучение.

Заключение

В этой главе подробно описан процесс выработки общего замысла (планирования на уровне продукта). Этот процесс продемонстрирован на примере того, как в вымышленной компании может быть составлена концепция нового продукта, сформирован задел и построен график выпуска продукта SmartReview4You, реализующего службу поиска и фильтрации обзоров. В этой главе также показано, насколько полезным может оказаться спринт по приобретению знаний в процессе выработки общего замысла для достижения порога доверия, необходимого для завершения выработки общего замысла. Далее в главе были рассмотрены рекомендации по выработке общего замысла экономически обоснованным способом, чтобы лучше согласовать работы по предварительному планированию продукта с последующими работами по созданию потребительской ценности в соответствии с методикой Scrum. А в следующей главе будет описано, как воспользоваться результатами выработки общего замысла на стадии планирования выпуска.

Глава 18

Планирование выпуска (долгосрочное планирование)

Планирование выпуска является долгосрочным планированием, позволяющим найти ответы на следующие вопросы: “Когда мы это сделаем?”, “Какие функциональные средства можно получить к концу года?” или “Во что это обойдется?” Планирование выпуска должно уравновешивать потребительскую ценность и общее качество с ограничениями на объем работ, сроки и бюджет. В этой главе поясняется, каким образом планирование выпуска вписывается в инфраструктуру Scrum и как оно выполняется в отношении фиксированных сроков и объемов работ над выпусками.

Краткий обзор

Каждая организация должна определить надлежащий размеренный темп выпуска функциональных средств для ее заказчиков (рис. 18.1).

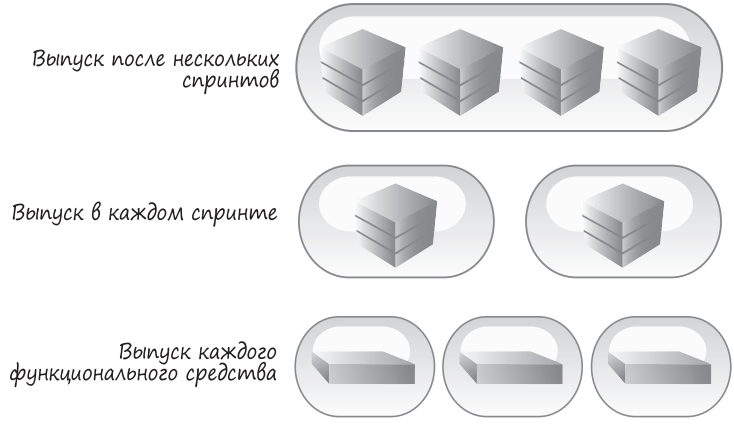


Рис. 18.1. Разные размеренные темпы выпуска

Несмотря на то что результат отдельного спринта потенциально готов к поставке, во многих организациях предпочитают не выпускать новые функциональные средства после каждого спринта. Вместо этого они объединяют результаты нескольких спринтов в один выпуск.

В других организациях размеренный темп выпуска согласуют с ритмом проведения спринта. В подобных случаях организация выпускает в конце каждого спринта прирост потенциально готового к поставке продукта, создаваемый во время данного спринта.

В некоторых организациях даже не ждут окончания спринта, а выпускают каждое функциональное средство по мере его готовности. Такая норма практики называется *непрерывной разработкой* или *непрерывной доставкой*. В подобных организациях функциональные средства или изменения в них для некоторых или всех потребителей выпускаются по мере их готовности, что может происходить с частотой до нескольких раз в день.

Свои намерения развертывать выпуски через каждый спринт, несколько спринтов или непрерывно большинство организаций воплощают в долгосрочные планы, которые они находят полезным составлять на более высоком уровне и в продолжительные сроки. Если вас не устраивает термин *планирование выпуска*, замените его более подходящим для вас термином. Ниже перечислены его синонимы, обычно употребляемые во многих организациях. Но как бы ни называть такое планирование, оно нацелено на будущее состояние проекта, когда нужно уравновесить важные показатели, в том числе сроки, объем работ и бюджет.

- **Долгосрочное планирование.** Преследует цель расширить горизонт дальше одного спринта.
- **Поэтапное планирование.** Называется так потому, что выпуски обычно приурочивают к важным промежуточным этапам вроде пользовательской конференции, завершения минимального набора функциональных средств или пригодного для продажи выпуска.

Временны*е* рамки

Планирование выпуска является не единовременным, а скорее частым событием, происходящим буквально в каждом спринте (рис. 18.2). Первоначальное планирование выпуска логически следует после планирования на уровне продукта, т.е. выработки общего замысла.

Назначение планирования продукта — выработать представление о том, каким должен быть продукт, т.е. его концепцию. А назначение планирования выпуска — определить следующий логический шаг к достижению конечной цели выпуска продукта.

Прежде чем начать выпуск, во многих организациях, практикующих методику Scrum, проводится первоначальное планирование выпуска с целью составить предварительный план выпуска. Как правило, это мероприятие длится в течение одного или двух дней, хотя его продолжительность зависит от масштабов, степени риска выпуска и знакомства участников проекта с предметом разработки.

Рис. 18.2. Когда происходит планирование спринта

Если разрабатывается новый продукт, то первоначальный план выпуска составляется не полностью и не точно. Вместо этого план выпуска обновляется по мере утвержденного обучения в течение выпуска. Планы выпуска могут пересматриваться при подведении итогов спринта или в ходе обычной подготовки к каждому последующему спринту или его проведения.

Участники

Планирование выпуска предполагает сотрудничество участников проекта и всей Scrum-команды. В какой-то момент эти лица привлекаются к планированию выпуска, поскольку им приходится идти на коммерческие и технические компромиссы, чтобы достичь золотой середины между ценностью и качеством. Степень участия каждого из этих лиц в планировании выпуска может со временем меняться.

Процесс

Процесс планирования выпуска приведен на рис. 18.3.

Предпосылками для планирования выпуска служат результаты планирования продукта, в том числе концепция продукта, задел продукта на высоком уровне, а также график выпуска продукта. Необходимо также знать скорость работы одной или нескольких команд, задействованных в выпуске. Для существующей команды таким показателем служит известная скорость ее работы, а иначе она прогнозируется во время планирования выпуска (см. главу 7).

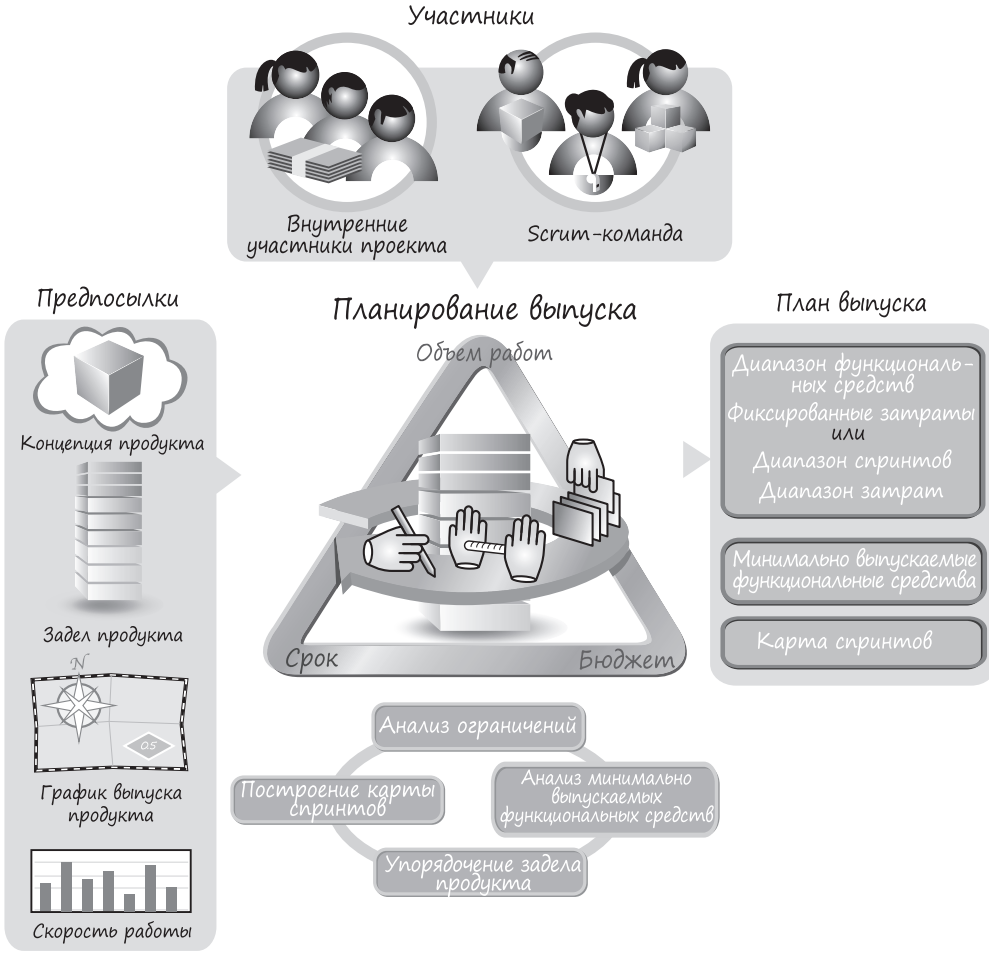


Рис. 18.3. Процесс планирования выпуска

Один вид деятельности, повторяющейся при планировании выпуска, состоит в утверждении ограничений, накладываемых на объем работ, сроки и бюджет, а также в их анализе с целью выявить любые изменения, которые следует внести со временем и на основании того, что известно о продукте и данном выпуске. А другой вид деятельности при планировании выпуска состоит в упорядочении задела продукта, которое включает в себя создание, оценивание и расстановку по приоритетам элементов более детализированного задела продукта на основании элементов высокоуровневого задела продукта. Эти виды деятельности могут происходить в разные моменты времени, в том числе следующие.

- После планирования продукта, но перед первоначальным планированием выпуска.
- В ходе первоначального планирования выпуска.
- В течение каждого спринта по мере необходимости (подробнее об упорядочении задела продукта см. в главе 6).

Каждый выпуск должен иметь вполне определенный набор минимально выпускаемых функциональных средств (MRF). Первоначальный набор минимально выпускаемых функциональных средств может быть определен в процессе выработки общего замысла. Несмотря на это, при планировании выпуска минимально выпускаемые функциональные средства всегда анализируются, чтобы они действительно представляли минимально жизнеспособный продукт (MVP) с точки зрения заказчиков.

При планировании выпуска во многих организациях составляется также карта спринтов, обозначающая спринты, в которых могут быть созданы некоторые или многие элементы задела продукта. Карта спринтов не составляется на слишком отдаленную перспективу, но служит для наглядного преставления того, что должно произойти в ближайшем будущем. Следовательно, она помогает команде лучше справляться с собственными зависимостями и ограничениями на ресурсы, а также координировать усилия многих сотрудничающих команд.

Результаты планирования выпуска совместно называют *планом выпуска*. План выпуска сообщает с той степенью точности, которая считается приемлемой на данной стадии проектных работ, когда будет завершен выпуск, какие функциональные средства будут получены и во что все это обойдется. Этот план дает также ясное представление о желательном наборе минимально выпускаемых функциональных средств. И наконец, он нередко показывает, каким образом элементы задела продукта распределяются по отдельным спринтам в течение выпуска.

Ограничения на выпуск

Назначение планирования выпуска — определить наиболее ценное содержимое следующего выпуска, а также желательный уровень его качества. Ограничения, накладываемые на объем работ, сроки и бюджет, являются важными показателями, оказывающими влияние на достижение конечной цели.

На основании результатов планирования продукта на выпуск накладывается одно ограничение или более. В главе 17 был рассмотрен пример планирования продукта в вымышленной компании Review Everything, Inc. В этой главе был прослежен весь процесс выработки общего замысла нового продукта SR4U — обучающегося поискового агента для поиска и фильтрации обзоров

в Интернете. В графике выпуска этого продукта Роджер и его команда указали, что было бы желательно выпустить первую его версию к предстоящей конференции Social Media Expo. Таким образом, на выпуск версии 1.0 продукта SR4U наложено следующее жесткое ограничение в виде жестко заданных сроков: этот выпуск должен быть готов к дате начала конференции Social Media Expo. А остальные ограничения (на объем работ и бюджет) являются гибкими.

В табл. 18.1 перечислены различные сочетания жестких и гибких ограничений на выпуск. А далее эти сочетания ограничений рассматриваются в свете их влияния на планирование выпуска.

Таблица 18.1. Сочетания жестких и гибких ограничений на выпуск

Тип проекта	Объем работ	Сроки	Бюджет
Все задано жестко (не рекомендуется)	Жесткий	Жесткие	Жесткий
Жестко заданные сроки и объем работ (не рекомендуется)	Жесткий	Жесткие	Гибкий
Жестко заданный объем работ	Жесткий	Гибкие	Жесткий (не совсем)
Жестко заданные сроки	Гибкий	Жесткие	Жесткий

Все задано жестко

Как пояснялось в главе 3, в традиционной, плановой, прогнозируемой методике разработки допускается, что требования известны и могут быть спрогнозированы заранее и что объем работ не изменяется. Исходя из этих представлений можно составить полный план и оценить затраты и график выполнения проекта. Такой метод наложения ограничений обозначается в табл. 18.1 как “Все задано жестко”.

При гибкой разработке по методике Scrum не считается возможным выяснить все особенности проекта заранее. Следовательно, если все параметры проекта задаются жестко, то такой метод вряд ли пригоден. Для планирования выпуска при гибкой разработке по методике Scrum требуется, чтобы хотя бы один из этих параметров был задан гибко.

Жестко заданные сроки и объем работ

Еще один метод состоит в том, чтобы задать жестко сроки и объем работ, а бюджет сделать гибким. Но такой метод страдает рядом недостатков. Во-первых, во многих организациях увеличить бюджет после начала проектных работ нелегко или вообще невозможно. Во-вторых, как показывает мой опыт, такой метод жестко фиксирует два параметра проекта, которые очень трудно определить заранее. И на практике оказывается, что, даже если посчитать

поначалу жестко заданными сроки и объем работ в выпуске, один из этих параметров все равно придется пересматривать.

Рассмотрим в качестве примера проблему 2000 года. Многие организации, работавшие над смягчением последствий этой проблемы, имели в своем распоряжении ряд приложений, которые требовалось обновить не позднее 31 декабря 1999 года. И многие из них жестко задали срок и объем работ, оставив свой бюджет гибким. Но в конечном итоге оказалось, что как бы эти организации ни увеличивали свой бюджет, они все равно не могли завершить все работы к крайнему сроку, жестко заданному на 31 декабря. Ведь эту дату нельзя было передвинуть, а следовательно, пришлось пересматривать объем работ. В каком-то смысле параметры сроков и объема работ постоянно играют друг с другом в игру "кто первым сдастся" (рис. 18.4)!

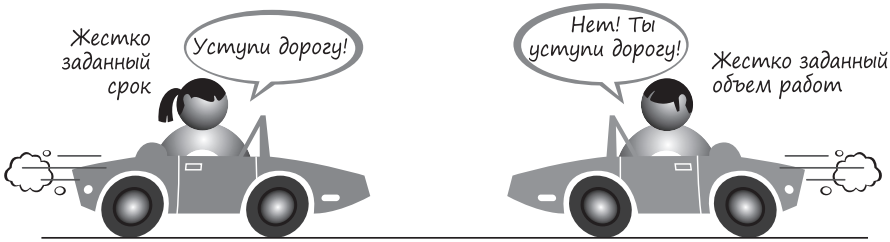


Рис. 18.4. Жестко заданные сроки и объем работ постоянно играют друг с другом в игру "кто первым сдастся"

В какой-то момент, когда время уже на исходе, приходится пересматривать либо сроки, либо объем работ. Если не удается сделать ни то ни другое, то в конечном итоге накапливается большой технический долг.

Когда сроки и объем работ задаются жестко, а бюджет оставляется гибким, предполагается, что чем больше ресурсов расходуется на разрешение затруднения, тем больше работы приходится выполнять и/или сокращать время на ее выполнение. Безусловно, такое иногда действительно случается во время разработки продукции. Например, можно потратить дополнительные средства, чтобы ускорить выполнение работ (возможно, нанять субподрядчика для выполнения части работ). В таком случае, тратя дополнительные средства, мы выигрываем время.

Но выиграть можно только во времени или объеме работ. Зачастую объем работ при разработке продукции нельзя сократить, а это означает, что расходование дополнительных ресурсов или средств вряд ли поможет, а возможно, и нанесет ущерб. Это положение Фред Брукс наглядно поясняет на следующем примере: "Девять женщин не родят ребенка за один месяц" [Brooks, Frederick P. 1995].

Во время разработки продукции понятие "гибкий бюджет" нередко означает "привлечь к работе больше людей". Но, как предупреждает Брукс и многие из нас знают по своему опыту, "привлечение рабочей силы в запоздалый проект

делает его еще более запоздалым" [Brooks, Frederick P. 1995]. Иногда привлечение дополнительных работников подходящей квалификации на ранней стадии выпуска может помочь делу. Но решение бросить рабочую силу на решение проблемы на поздней стадии редко помогает добиться успешного выпуска при жестко заданных сроках и объеме работ.

На практике гибкий бюджет многих организаций редко означает привлечение к работе над проектом большего числа людей. Как правило, такой бюджет означает, что тем же самым людям придется работать больше времени, особенно если это работники на твердом окладе. Чрезмерные сверхурочные работы с целью удовлетворить ограничениям на жестко заданные сроки и объем работ истощают трудовые ресурсы и нарушают принцип Scrum, требующий вести разработку в размеренном темпе.

Если приходится работать над выпуском, который первоначально определен с жестко заданными сроками и объемом работ, то итеративный и постепенный подход к разработке по методике Scrum позволяет раньше понять, что дела обстоят неважно. Это оставляет больше времени, чтобы заново уравновесить ограничения, накладываемые на сроки, объем работ и бюджет, и тем самым добиться успешных результатов.

До сих пор обсуждалось, что если зафиксировать жестко все или только сроки и объем работы, то такие ограничения на выпуск оказываются чрезмерными для разработки продукции. И тогда остаются лишь два других реалистичных варианта наложения ограничений на выпуск: задать жестко объем работ или сроки.

Жестко заданный объем работ

Модель с жестко заданным объемом работ оказывается пригодной там, где объем работ действительно важнее, чем сроки их выполнения. Если мы, придерживаясь этой модели, исчерпаем время и не завершим все функциональные средства, то можем перенести сроки выпуска, чтобы сделать все необходимое для удовлетворения критериям минимально выпускаемых функциональных средств. Эту модель нельзя назвать с жестко заданным объемом работ *и* бюджетом, поскольку бюджет на самом деле нельзя задать жестко. Ведь если предоставить команде больше времени для завершения проектных работ, то ее члены вправе ожидать, что им заплатят за их труд! Иными словами, если предоставить больше времени для завершения жестко заданного объема работ, то придется также выделить больший бюджет для оплаты труда людей в течение этого дополнительного времени.

Нередко объем работ для выпуска задается жестко, потому что общий объем работ слишком велик. Вместо этого лучше рассмотреть возможность более мелких, но частых выпусков с жестко заданными сроками. Кроме того, в тех организациях, где многие отделы (например, разработки, маркетинга и технической

поддержки) должны координировать свои действия, перенос сроков в одних отделах может сильно нарушить планы других отделов. Тем не менее далее в этой главе будет показано, как спланировать выпуск с жестко заданным объемом работ, применяя методику Scrum, когда объем работ важнее, чем сроки их выполнения.

Жестко заданные сроки

Метод жесткого задания сроков выпуска приведен в табл. 18.1 последним. Многие, в том числе и я, считают этот метод наиболее точно соответствующим принципам Scrum. Проще говоря, он позволяет жестко задать сроки и бюджет, но оставить гибким объем работ.

Принцип создания в первую очередь высокоприоритетных функциональных средств по методике Scrum должен смягчить любые огорчения от потери других функциональных средств. Когда истекает время выпуска с жестко заданными сроками, то все, что еще не создано, должно иметь более низкую ценность, чем все, что уже создано. Ведь намного проще принять решение о поставке, если недостающие функциональные средства имеют малую ценность. А если в выпуске отсутствуют функциональные средства, имеющие большую ценность, то, вероятнее всего, сроки выпуска придется перенести, если это, конечно, возможно.

Но это можно сделать лишь в том случае, если высокоприоритетные функциональные средства действительно считаются готовыми по согласованному критерию готовности. Вряд ли нас устроит ситуация, когда обязательные высокоприоритетные функциональные средства готовы лишь на 75–90% и одно из них придется исключить из выпуска, чтобы сделать полностью готовыми остальные средства в жестко заданные сроки.

Пользоваться моделью с жестко заданными сроками становится легче, если определить лишь небольшой набор минимально выпускаемых функциональных средств. Если такой набор удобно выпустить в жестко заданные сроки, то все в порядке, поскольку любые другие функциональные средства по определению являются лишь желательными, но не обязательными.

Выпуски с жестко заданными сроками вполне согласуются и с акцентом, который делается в Scrum на ограничении по времени. Устанавливая фиксированное время для выпуска, мы ограничиваем объем выполняемых работ и вынуждаем людей принимать трудные решения о приоритетности этих работ.

Переменное качество

При общем ограничении объема работ, сроков и бюджета качество становится “гибким”. С одной стороны, это может привести к выпуску продукта, не удовлетворяющего ожиданиям заказчика. А с другой стороны, как обсуждалось в главе 8, “гибкое” (т.е. переменное) качество может привести к накоплению

технического долга, что затруднит в дальнейшем ввод функциональных средств в продукт или его адаптацию.

Коррекция ограничений

Важной частью непрерывного планирования выпуска является приобретение текущих знаний и коррекция ограничений с целью уравновесить их заново. Например, что делать Роджеру и его команде из компании Review Everything, если они приближаются к крайнему сроку выпуска 1.0 продукта SR4U и совершенно ясно, что они не успеют завершить минимально выпускаемые функциональные средства? Данный выпуск имеет жестко заданные сроки, и поэтому лучше всего исключить из этого выпуска малоценные функциональные средства. Но допустим, что в данном случае им придется исключить обязательные функциональные средства, которые частично являются минимально выпускаемыми, чтобы удовлетворить ограничениям на сроки выпуска.

Вероятно, правильное решение состоит в том, чтобы определить меньший набор функциональных средств, включаемых в состав минимально выпускаемых функциональных средств. Например, в первоначальной версии продукта SR4U основное внимание можно уделить фильтрации ресторанных рецензий, получаемых лишь из небольшого числа фиксированных источников. Роджеру и его команде придется оценить, понизит ли сокращение объема работ воспринимаемую потребителем ценность выпуска до неприемлемого уровня. И если будет решено, что компания Review Everything не может исключить функциональные средства из данного выпуска без значительного ущерба для его ценности, то руководству компании останется одно из двух: рассмотреть возможность привлечь больше людей к данному проекту, изменив его бюджет, или отказаться от надежды запустить службу к началу конференции Social Media Expo, перенеся сроки выпуска. Подобные решения приходится постоянно принимать, пересматривать и принимать снова во время выполнения проектных работ.

Упорядочение задела продукта

Главным видом деятельности при планировании выпуска является упорядочение задела продукта для достижения целей качества и ценности. В процессе выработки общего замысла (т.е. планирования продукта) сначала создается задел продукта на высоком уровне (возможно, на уровне эпических историй), а затем он используется при определении набора минимально выпускаемых функциональных средств для каждого выпуска. Многие из этих элементов задела продукта слишком велики, чтобы оказаться пригодными для планирования выпуска.

Например, в процессе выработки общего замысла продукта SR4U Роджер предложил общее представление о том, какие именно высокоуровневые функциональные средства должны быть доступны к началу конференции Social Media Expo. Допустим, что в предложенном Роджером графике выпуска продукта указано, что в выпуске 1.0 основное внимание будет сосредоточено на функциональных средствах элементарного обучения и фильтрации, которые соответствуют следующим элементам задела продукта.

- Мне как типичному пользователю требуется обучить службу SR4U тем видам обзоров, которые она должна отсеивать, чтобы знать критерии для отсеивания обзоров от моего имени.
- Мне как типичному пользователю требуется простой, как в Google, интерфейс для запросов на поиск обзоров, чтобы не тратить много времени на описание того, что мне требуется.

Эти элементы слишком велики, чтобы справиться с ними в течение планирования выпуска. Поэтому для их уточнения Роджер и его команда должны провести совещание по написанию пользовательских историй (см. главу 5) в рамках совещания по планированию выпуска, а возможно, и как отдельное мероприятие перед совещанием по планированию выпуска. В результате такого совещания по написанию пользовательских историй в заделе продукта появится немало более детализированных элементов, в том числе следующие.

- Мне как типичному пользователю требуется указывать службе SR4U игнорировать обзоры, содержащие отдельные ключевые слова, которые, на мой взгляд, обозначают предвзятость обзора, чтобы я не просматривал ни один из обзоров, содержащих эти ключевые слова.
- Мне как типичному пользователю требуется выбирать категорию продуктов или услуг, чтобы помочь службе SR4U сосредоточиться только на подходящих обзорах.

Как только истории окажутся достаточно мелкими, команда сможет их оценить (см. главу 7), чтобы составить общее представление о затратах. (Такое оценивание отчасти требуется для первоначального планирования выпуска. И как только новые истории появятся в течение выпуска, их нужно будет также оценить для непрерывного хода планирования выпуска.) Участники планирования выпуска затем расставят по приоритетам оцененные истории, исходя из цели выпуска и накладываемых на него ограничений. После перестановки приоритетов в заделе продукта участники планирования выпуска должны быть начеку, чтобы набор минимально выпускаемых функциональных средств был всегда определен и согласован.

Уточнение состава минимально выпускаемых функциональных средств

Как пояснялось в главе 17, минимально выпускаемые функциональные средства представляют наименьший набор обязательных функциональных средств, которые должны быть непременно включены в выпуск, если требуется удовлетворить ожидания заказчика в отношении ценности и качества продукта. Важная часть планирования выпуска состоит в прилежном переоценивании и уточнении состава минимально выпускаемых функциональных средств для выпуска. По мере получения скорой ответной реакции на спринты и утвержденного обучения постоянно корректируется состав минимально выпускаемых функциональных средств.

Во многих организациях, с которыми я сотрудничал, мне нередко приходилось сталкиваться с неспособностью согласовать состав минимально выпускаемых функциональных средств, поскольку многие участники проекта отстаивали свою точку зрения и не могли прийти к общему согласию. Но ведь неудачно определенный или пассивно-агрессивно согласованный состав минимально выпускаемых функциональных средств мешает принять правильное решение при планировании выпуска. Если, например, время на исходе, то какие именно функциональные средства следует исключить из выпуска? Недостаточная ясность в отношении состава минимально выпускаемых функциональных средств может усложнить принятие такого решения.

При гибкой разработке по методике Scrum владелец продукта несет окончательную ответственность за определение состава минимально выпускаемых функциональных средств. Разумеется, он может и должен сделать это в тесном сотрудничестве с соответствующими участниками проекта и Scrum-командой.

Некоторым понятие минимально выпускаемых функциональных средств может показаться нелогичным: почему бы не попытаться выпустить самый крупный, а не самый мелкий набор функциональных средств? На этот вопрос проще всего ответить следующим образом: самый крупный набор выпускаемых функциональных средств, вероятно, обойдется дороже, отнимет больше времени и наиболее рискован. С другой стороны, минимально возможный набор выпускаемых функциональных средств должен обойтись дешевле всего, отнять меньше всего времени и наименее рискован. Следовательно, определение минимального состава выпускаемых функциональных средств в большей степени соответствует принципу более мелких, но частых выпусков, как пояснялось в главе 14.

Состав минимально выпускаемых функциональных средств должен быть определен со знанием их масштабов, определяемых во время упорядочения задела продукта, хотя с этим согласны не все. Некоторые считают, что состав минимально выпускаемых функциональных средств должен быть определен независимо от затрат. Это означает, что минимально выпускаемыми следует считать

такие функциональные средства, которые соответствуют порогу пользовательской ценности для данного выпуска (и такое определение их состава должно осуществляться независимо от данных о затратах). Представление о первоначальном наборе минимально выпускаемых функциональных средств можно составить и без данных о затратах, но поскольку все решения при планировании выпуска должны приниматься в обоснованных экономических рамках, то, зная затраты на разработку функциональных средств, можно проверить экономическую целесообразность выпуска минимального набора функциональных средств. Если же выяснится, что выпускать минимальный набор функциональных средств нецелесообрано, то, вероятно, настало время резко сменить направление разработки.

Составление карты спринтов

В каждом спринте команда работает над рядом элементов из задела продукта. Команда и владелец продукта не решают, над какими элементами задела продукта им следует работать в конкретном спринте, до тех пор, пока они не приступят к планированию спринта. Означает ли это, что перед совещаниями по планированию спринтов у них нет ни малейшего представления о том, как распределить элементы задела продукта по отдельным спринтам?

Совсем нет! Некоторые команды считают даже полезным быстрое и раннее распределение (или разделение) элементов задела продукта по отдельным спринтам, называемое иначе *составлением карты спринтов*. Например, распределение по нескольким спринтам в многокомандной среде может помочь командам лучше скоординировать свои действия.

Для такого распределения требуется надлежащим образом детализированный, оцененный и расставленный по приоритетам задел продукта. Используя показатель скорости работы команды, можно приблизительно определить состав элементов задела продукта для каждого спринта, сгруппировав вместе те элементы, совокупный размер которых примерно соответствует средней скорости работы команды. Полученный результат может выглядеть так, как показано на рис. 18.5, *слева*. Некоторые предпочитают отображать карту спринтов по горизонтали, как показано на рис. 18.5, *справа*, чтобы она в большей степени была похожа на временную шкалу. Мне встречались и такие Scrum-команды, которые размещали горизонтально ориентированную карту спринтов над стандартным планом проекта (например, диаграммой Гантта), описывавшим работу команд, не практиковавших методику Scrum, чтобы нагляднее представить согласование и точки соприкосновения этих разных категорий команд в работе над проектом.

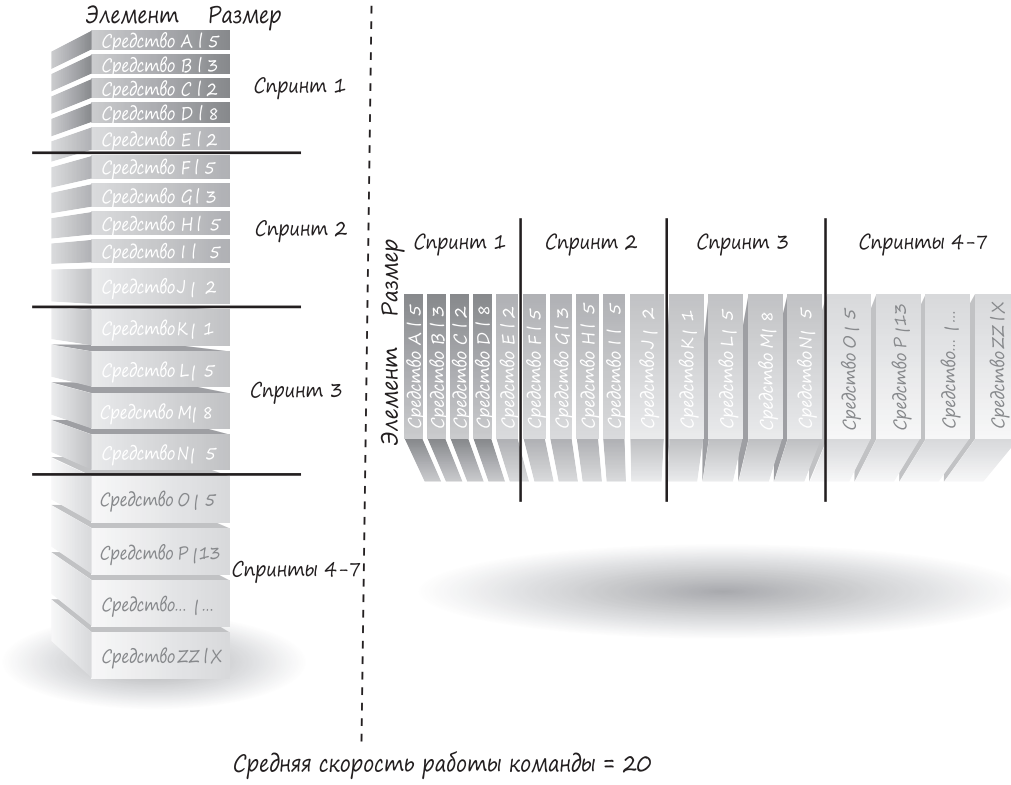


Рис. 18.5. Распределение элементов задела продукта по отдельным спринтам, иначе называемое составлением карты спринтов

Если разработка ведется силами одной Scrum-команды, такое распределение элементов из задела продукта можно было бы выполнить при первоначальном планировании выпуска, чтобы получить общее представление о том, когда именно будут созданы отдельные функциональные средства в течение выпуска. Такое распределение может также побудить к реорганизации задела продукта, чтобы сгруппировать его элементы более естественным или эффективным образом. Реорганизовать можно и работу, чтобы результатов, достигнутых в конце спринта, оказалось достаточно для утвержденного обучения и оперативной ответной реакции.

Если же разработка ведется силами многих команд, то, возможно, придется выполнить некоторое предварительное распределение элементов задела продукта по отдельным спринтам, чтобы упростить управление зависимостями, возникающими между командами. С этой целью можно, в частности, воспользоваться формой скользящего перспективного планирования [Cohn, Mike. 2006], где каждая команда рассматривает нужные элементы задела продукта не для одного,

а по крайней мере для двух и даже большего числа предстоящих спринтов. Таким образом, Scrum-команды, задействованные в проектных работах, будут знать, какая из них и когда приблизительно собирается работать над конкретными элементами.

Допустим, что в работе над проектом SR4U задействованы три Scrum-команды. Первая команда занимается сквозной обработкой пользовательских запросов, разрабатывает функциональные средства, дающие пользователям возможность делать запросы на обзоры, выполнять запросы и получать результаты их обработки. Вторая занимается созданием механизма с искусственным интеллектом, обладающим развитой логикой для анализа и различения обзоров. А третья команда занимается установлением соединения с разными источниками информации в Интернете для извлечения обзоров, претендующих на просмотр пользователем.

Эти три команды должны координировать свои действия, чтобы разработать минимально выпускаемые функциональные средства к моменту начала конференции Social Media Expo. В связи с этим целесообразно, чтобы все три команды приняли участие в совместном планировании выпуска.

В ходе совещания по первоначальному планированию выпуска каждая команда предлагает свое представление о том, когда она будет готова приступить к работе над своими элементами задела продукта. При последующем обсуждении первая команда может заявить: “Мы считаем, что будем готовы создать функциональное средство, отсеивающее обзоры с конкретными ключевыми словами, во втором спринте. Но нам потребуется помощь третьей команды, чтобы извлечь данные хотя бы из одного источника в Интернете до начала второго спринта или в самом его начале”. Члены третьей команды могут тогда справиться по своей карте спринтов, планируют ли они в настоящий момент сделать доступным ко второму спринту хотя бы один источник в Интернете. Если они этого не планируют, то обе команды могут обсудить возникшую зависимость и совместно выработать те коррективы, которые необходимо внести в свои планы одной или обеим командам.

Если принимается решение выполнить раннее распределение элементов задела по отдельным спринтам, то нужно понимать, что такое распределение может и должно получить дальнейшее развитие в процессе создания выпуска. В конечном счете решение о том, какой из команд работать над конкретными функциональными средствами в отдельном спринте, принимается в последний удобный момент, когда этот спринт планируется.

С другой стороны, во многих организациях (особенно тех, где разработка обычно ведется силами одной команды) предпочитают выполнять лишь незначительное раннее распределение элементов задела продукта по отдельным спринтам или совсем не делать этого. Команды в таких организациях считают,

что усилия, затраченные на подобное распределение, не оправдываются той ценностью, которую оно доставляет. Они полагают, что на совещании по предварительному планированию выпуска вопросы распределения элементов задела продукта по отдельным спринтам не должны рассматриваться — по крайней мере, они не существенны.

Планирование выпуска с жестко заданными сроками

Как упоминалось ранее, во многих организациях, практикующих методику Scrum, предпочитают метод выпуска с жестко заданными сроками. В табл. 18.2 перечислены этапы планирования выпуска с жестко заданными сроками.

Таблица 18.2. Этапы планирования выпуска с жестко заданными сроками

Этап	Описание	Комментарии
1	Определить количество спринтов в данном выпуске	Если у всех спринтов одинаковая продолжительность, то их нетрудно определить по календарю, поскольку заранее известно начало первого спринта и срока выпуска
2	Упорядочить задел продукта на достаточную глубину путем создания, оценивания размеров и расстановки по приоритетам элементов в заделе продукта	Мы пытаемся выяснить, какие именно элементы задела продукта можно получить к жестко заданному сроку, и поэтому их количество должно быть таким, чтобы запланировать этот срок
3	Измерить или оценить скорость работы команды в виде диапазона	Определяется диапазон средних скоростей работы от низкой до высокой (см. главу 7)
4	Умножить низкую скорость работы на количество спринтов. Отсчитать количество полученных очков от вершины задела продукта и провести разделительную линию	Эта разделительная линия обозначает все, что *будет* включено в выпуск
5	Умножить высокую скорость работы на количество спринтов. Отсчитать количество полученных очков от вершины задела продукта и провести вторую разделительную линию	Эта разделительная линия обозначает все, что *может* быть включено в выпуск

Обратимся снова к примеру разработки продукта SR4U. Его выпуск версии 1.0 привязан к конференции Social Media Expo, которая должна начать свою работу в понедельник 26 сентября. В компании Review Everything решают, что на этой конференции было бы неплохо продемонстрировать первоначальную версию данного продукта. И это стало бы отличным промежуточным этапом на пути к реализации концепции данного продукта.

И хотя в данном примере мы допустили, что над продуктом SR4U могут работать три команды, вернемся все же к первоначальному предположению поручить разработку данного продукта одной группе. Роджер и его команда хотели бы начать первый спринт с первой недели июля и завершить выпуск к 23 сентября, т.е. в пятницу накануне конференции. Поэтому им нетрудно было определить по календарю, сколько спринтов потребуется для выполнения данного выпуска. Если учесть, что продолжительность каждого спринта в данном выпуске одинакова (это обычное явление в Scrum), то для выпуска версии 1.0 продукта SR4U потребуется шесть спринтов по две недели (десять рабочих дней) каждый. Распределение этих спринтов по календарю приведено на рис. 18.6.

Рис. 18.6. Календарь спринтов для выпуска версии 1.0 продукта SR4U

Затем они определили объем работ, который команда должна выполнить в течение шести спринтов. Воспользовавшись методом, описанным в главе 7, они рассчитали скорость работы команды в пределах от 18 до 22 очков за историю в течение каждого спринта. Следовательно, Роджер и его команда должны были суметь выполнить работы объемом от 108 до 132 очков на историю в течение данного выпуска.

Далее они определили состав функциональных средств в данных пределах очков за историю. По завершении планирования продукта Роджер и его команда получили в свое распоряжение задел продукта, составленный на высоком уровне эпических и тематических пользовательских историй. Как пояснялось ранее, эта команда затем провела совещание по написанию пользовательских историй, чтобы создать более детализированные элементы задела продукта. Затем она оценила их, а владелец продукта с участием команды разработчиков и прочих заинтересованных лиц расставил их по приоритетам.

В ходе данного процесса Роджер и его команда должны были определить ряд обязательных функциональных средств, составляющих минимально выпускаемые функциональные средства. На практике для работы над минимально выпускаемыми функциональными средствами в выпуске с жестко заданными сроками требуется меньше, чем 100% всего времени, выделяемого на выпуск. Я лично предпочитаю выделять для них от 60 до 70% этого времени по следующим причинам.

- Если время выпуска на исходе, а все функциональные средства, намеченные для этого выпуска, являются обязательными, то какое из этих средств следует исключить из выпуска? Ведь если все функциональные средства являются обязательными, то по определению ни одно из них нельзя исключать из выпуска. Если же определить, что выпуск должен состоять на 60–70% из обязательных функциональных средств, а остальной объем работ направить на желательные функциональные средства, то последние можно исключить из выпуска, чтобы сократить общий объем работ и уложиться в срок.
- Если составить выпуск полностью из обязательных функциональных средств, то что произойдет, когда во время выпуска неожиданно появится еще одно обязательное функциональное средство? Иными словами, это средство, о котором раньше ничего не было известно, может возникнуть как совершенно необходимое для включения в выпуск, чтобы сделать последний вполне жизнеспособным. Как выйти из этого затруднительного положения? Если бы определить выпуск таким образом, чтобы он включал в себя желательные функциональные средства, то одно или больше таких средств можно было бы исключить из выпуска, а вместо них включить в него неожиданно возникшее обязательное функциональное средство.

В конечном итоге получается задел продукта, который структурно выглядит так, как показано на рис. 18.7 и как его понимает Роджер и его команда в данный момент, включая тематические и эпические истории, не запланированные для данного выпуска.

Роджер и его команда могут далее применить результаты сделанных ранее расчетов скорости работы, в ходе которых команда оценила ее в пределах от 108 до 132 очков для выполнения проектных работ в течение шести спринтов. Команда может наглядно представить, до какого места в заделе продукта она может добраться, отсчитав от его вершины сначала 108 очков, а затем 132 очка (рис. 18.8).

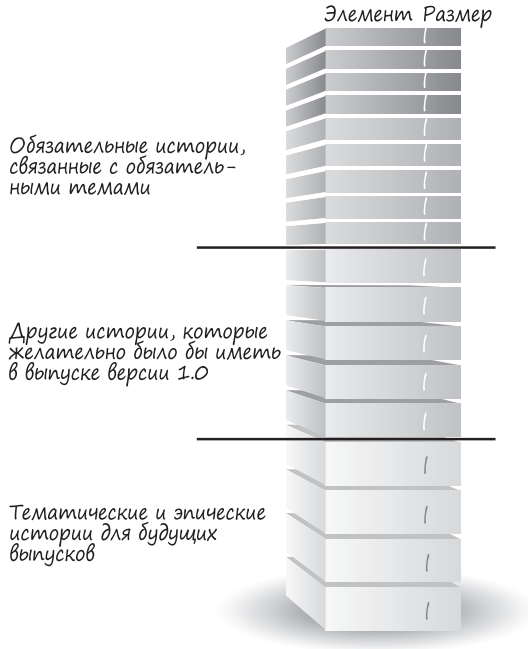


Рис. 18.7. Задел продукта, подготовленный к планированию выпуска

Рис. 18.8. Определение пределов для включения функциональных средств в выпуск с жестко заданными сроками

Как видите, двумя линиями на рис. 18.8 задел продукта разделяется на три части: включительно, возможно, исключительно. Подобным способом наглядно показывается, как ответить на следующий вопрос: “Что будет получено к намеченному сроку выпуска?” На ранней стадии выпуска очень трудно дать точный ответ на этот вопрос. Но в то же время можно дать приблизительный в некоторых пределах ответ, который, впрочем, содержит в себе неопределенность. И чем шире эти пределы, тем больше неопределенность.

Чтобы понять, насколько верно составлен такой план выпуска, Роджеру и его команде достаточно провести через задел продукта, показанный на рис. 18.8, разделительную линию, обозначающую то, что должно быть обязательно включено в выпуск. Некоторые из возможных результатов такой операции приведены на рис. 18.9. Обратите внимание на то, что линия, обозначающая то, что должно быть обязательно включено в выпуск, отделяет минимально выпускаемые функциональные средства, которые оказываются выше этой линии, от остальной части задела продукта.

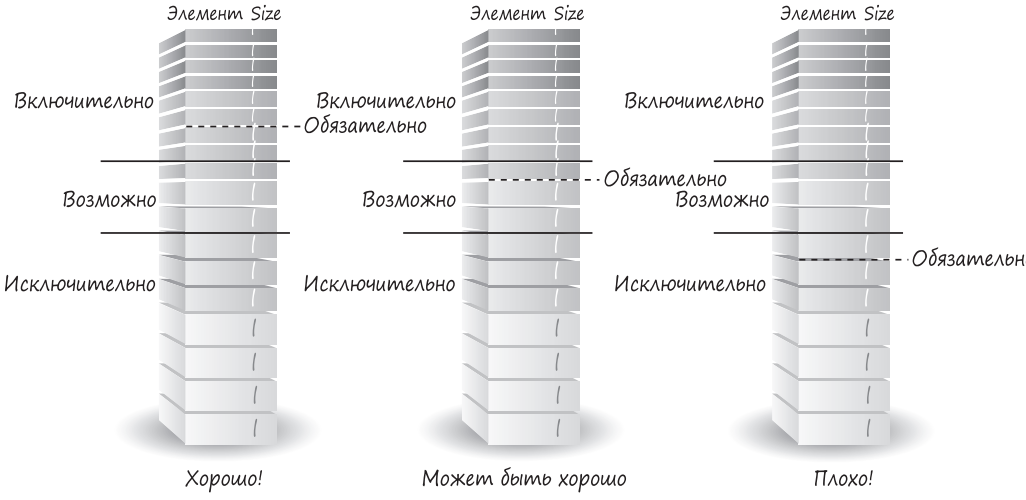


Рис. 18.9. Расположение обязательных функциональных средств относительно пределов выпускаемых функциональных средств в заделе продукта

Крайний слева задел продукта на рис. 18.9 обозначает весьма положительную ситуацию. Ее можно трактовать следующим образом: мы получим все обязательные функциональные средства, а следовательно, должны продолжить работу над выпуском.

Средний задел продукта на рис. 18.9 обозначает следующее: мы получим бо́льшую часть обязательных функциональных средств, но, возможно, не все. Очевидно, что в данном случае риск больше, чем в предыдущем. Если пойти

на такой риск, мы получим не все обязательные функциональные средства, но сможем продолжить работу дальше. А поскольку мы планируем учиться быстро, то могли бы решиться начать работу над данным выпуском и выполнить несколько спринтов. После этого мы могли бы еще раз оценить положение дел и принять решение продолжить работу над выпуском или уничтожить его, как пояснялось в предыдущих главах. Кроме того, ответная реакция на уже завершенные работы могла бы подсказать, что некоторые функциональные средства, первоначально включенные в состав минимально выпускаемых функциональных средств, на самом деле не являются обязательными, а следовательно, все в порядке.

С другой стороны, можно рассмотреть возможность установить новый, более поздний срок, который нетрудно рассчитать, или же привлечь больше людей к проектным работам, чтобы увеличить скорость работы, если это, конечно, поможет. На данном этапе многие организации могут выбрать накопление технического долга, чтобы команда смогла выпустить обязательные функциональные средства к намеченному сроку, хотя и за счет снижения уровня качества. Но если технический долг берется в текущем выпуске, то он должен быть возмещен в следующем выпуске, что приведет к сокращению величины доставляемой ценности.

Крайний справа задел продукта на рис. 18.9 обозначает следующее: мы *не* получим обязательные функциональные средства. В таком случае вряд ли стоит продолжать работу над данным выпуском, а, возможно, следует изменить срок выпуска или задействовать больше ресурсов. Если же выбрать в подобной ситуации накопление технического долга, то его объем, скорее всего, окажется немалым. Разумеется, если будет решено продолжить работу над выпуском, план выпуска придется пересматривать в каждом спринте, чтобы обновлять его, опираясь на текущие знания о продукте.

Например, в конце каждого спринта появляется дополнительная контрольная точка замера скорости работы, которая для новой команды, не накопившей особенно много данных о скорости работы, или для команды, выполнявшей совершенно другую работу, будет означать необходимость перерасчета среднего количества очков, которые команда может реализовать за спринт. А ведь элементы в заделе продукта могли, как и следовало ожидать, измениться. В частности, могли появиться новые элементы, тогда как другие элементы — перенесены из данного выпуска или вообще удалены, как только стало известно, что они не нужны теперь или вообще. Чтобы наглядно представить пересмотренный план выпуска, придется снова нарисовать картину, аналогичную приведенной на рис. 18.8.

Планирование выпуска с жестко заданным объемом работ

Несмотря на то что выпуски с жестко заданными сроками весьма распространены в гибкой разработке по методике Scrum, вполне возможно, что объем работ над продуктом окажется важнее, чем сроки его сдачи. Что, если, например, имеется большой набор обязательных функциональных средств, не вписывающийся в состав минимально выпускаемых функциональных средств, и поэтому требуется переместить сроки выпуска, чтобы завершить работу над всеми этими средствами?

Можно ли в таком случае отобрать абсолютно минимальный состав обязательных функциональных средств? Мне иногда приходится слышать такой ответ на данный вопрос: "Но ведь мы реализуем стандарт, а выпустить половину стандарта нельзя". И хотя это может быть и так, в большинстве стандартов вполне возможны такие части, которые совсем не обязательно реализовывать теперь (например, в веб-браузере могут поддерживаться изменяющиеся или появляющиеся стандарты HTML или CSS). Вполне возможны и такие случаи, когда приходится выходить на рынок с реализованным не полностью стандартом, извещая потребителей, какие именно части стандарта поддерживаются, а какие нет.

На мой взгляд, если рассуждать постепенно и решительно нацелиться на действительно минимальный состав выпускаемых функциональных средств, то зачастую выпуск с жестко заданным объемом работ можно превратить в ряд более мелких выпусков с жестко заданными сроками. Когда состав минимально реализуемых функциональных средств становится мелким, то преобладающим, как правило, становится другое ограничение (например, время).

Допустим, что мы отобрали обязательные функциональные средства в самый минимальный набор, а главное ограничение на выпуск по-прежнему накладывается по дате, когда эти средства будут готовы. В таком случае планирование выпуска осуществляется так, как показано в табл. 18.3.

Если выпуск осуществляется с жестко заданным объемом работ, то нужно знать, какие именно функциональные средства следует реализовывать в начале выпуска. Мы могли бы знать эти функциональные средства, если бы разрабатывали простой или знакомый продукт. Но при разработке новаторских продуктов многие функциональные средства будут возникать и развиваться в течение проектных работ. И хотя у нас, безусловно, имеется некоторое предварительное представление о желательных функциональных средствах, тем не менее, мы должны быть готовы к постоянному пересмотру плана выпуска по мере прояснения нашего представления о требующихся изменениях в функциональных средствах.

Таблица 18.3. Этапы планирования выпуска с жестко заданным объемом работ

Этап	Описание	Комментарии
1	Упорядочить задел продукта путем создания, оценивания размеров и расстановки по приоритетам хотя бы тех элементов в заделе продукта, которые требуется включить в данный выпуск	Нужно точно знать, какие именно элементы из задела продукта входят в планируемый объем работ, поскольку речь идет о выпуске с жестко заданным объемом работ
2	Определить общий объем работ над элементами задела продукта, доставляемыми в данном выпуске	Если имеется задел продукта, состоящий из оцененных элементов, то достаточно просуммировать оцененные размеры всех элементов, которые должны войти в данный выпуск
3	Измерить или оценить скорость работы команды в виде диапазона	Определяется диапазон средних скоростей работы от низкой до высокой
4	Разделить общий размер элементов в заделе продукта на высокую скорость работы и округлить результат до следующего целого числа	В итоге получится наименьшее количество спринтов, требующихся для выпуска функциональных средств
5	Разделить общий размер элементов в заделе продукта на низкую скорость работы и округлить результат до следующего целого числа	В итоге получится наибольшее количество спринтов, требующихся для выпуска функциональных средств

Если мы проводим совещание по планированию выпуска в его начале, то сначала должны упорядочить задел продукта, как это делалось при планировании выпуска с жестко заданными сроками. Отличие заключается в том, что при планировании выпуска с жестко заданными сроками мы пытаемся включить в выпуск не все обязательные элементы, чтобы каким-то образом защититься от неопределенности. А при планировании выпуска с жестко заданным объемом работ нам требуется определить весь объем работ для выпуска обязательных функциональных средств. При выпусках с жестко заданным объемом преследуется цель своевременно завершить все обязательные функциональные средства. Если же по ходу дела возникают дополнительные функциональные средства, мы просто включаем их в объем работ для данного выпуска и смещаем его дату.

При планировании выпуска с жестко заданными сроками мы точно знаем, сколько спринтов нам предстоит провести. А при планировании выпуска с жестко заданным объемом работ мы должны рассчитать количество спринтов, требующихся для выпуска жестко заданного набора функциональных средств.

Чтобы произвести необходимые расчеты, нам потребуется диапазон скоростей работы команды, как и при планировании выпуска с жестко заданными сроками. Допустим, что скорость работы команды в двухнедельном спринте

находится в пределах от 18 до 22 очков за историю. Чтобы ответить на вопрос, когда будет готов жестко заданный набор функциональных средств, достаточно просуммировать сначала размеры всех этих функциональных средств как элементов задела продукта, а затем разделить полученную сумму на высокую и низкую скорости работы команды. В итоге получится диапазон спринтов, в котором будет происходить выпуск.

Допустим, что в следующем выпуске для реализации функциональных средств потребуется 150 очков за историю. Если разделить 150 очков на 18 (т.е. низкую скорость работы команды) и округлить результат, то в итоге получатся девять спринтов. А если разделить 150 очков на 22 (т.е. высокую скорость работы команды) и округлить результат, то в итоге получатся семь спринтов. Эти итоги расчетов можно наглядно продемонстрировать, как показано 18.10.

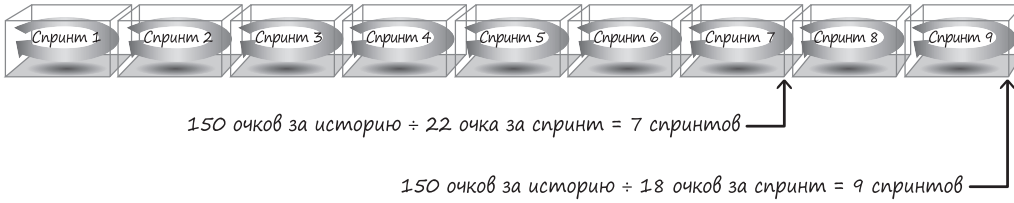


Рис. 18.10. Результаты планирования выпуска с жестко заданным объемом работ

Следует иметь в виду, что и в этом случае приходится давать приблизительный в некоторых пределах ответ, но теперь на следующий вопрос: "Сколько спринтов потребуется для завершения выпуска с объемом работ, оцениваемым в 150 очков?" Ответ в данном случае будет находиться в пределах от семи до девяти спринтов. А поскольку эти спринты должны продолжаться не больше двух недель, то ответ будет находиться в пределах от 14 до 18 недель.

Расчет затрат на выпуск

Рассчитать затраты на весь выпуск с жестко заданными сроками или объемом работ совсем не трудно, как следует их табл. 18.4.

Допустим, что состав команды, которой поручены проектные работы, остается достаточно устойчивым. Иными словами, людей не перемещают из одной команды в другую. Но если это все-таки делается, то изменения в составе команды должны быть незначительными, причем труд перемещаемых работников должен оплачиваться приблизительно одинаково.

Исходя из подобных предположений мы можем легко определить затраты на каждый спринт, поскольку знаем состав команды и продолжительность спринтов. Если исключить из рассмотрения остальные затраты (например, капитальные), что нередко оказывается оправданным, поскольку бо́льшая часть

расходов на разработку программного обеспечения зачастую приходится на зарплату работников, то затраты на зарплату будут довольно точно отражать общие затраты на каждый спринт.

Таблица 18.4. Расчет затрат на выпуск

Этап	Описание	Комментарии
1	Определить состав команды	Допустим, что состав команды по существу не меняется в течение спринта или от одного спринта к другому
2	Определить продолжительность спринта	Допустим, что все спринты имеют одинаковую продолжительность
3	Определить затраты на зарплату при выполнении спринта, исходя из состава команды и продолжительности спринта	Это делается просто, если предыдущие предположения верны, но дело немного усложняется, если состав команды или продолжительность спринта колеблется
4a	Умножить количество спринтов в выпуске на затраты, приходящиеся на каждый спринт, если речь идет о выпуске с жестко заданными сроками	В результате получаются затраты на зарплату для данного выпуска
4b	Умножить большое и малое количество спринтов на затраты, приходящиеся на каждый спринт, если речь идет о выпуске с жестко заданным объемом работ	В результате получаются пределы затрат на зарплату для данного выпуска. Один предел обозначает меньшие затраты на выпуск, а другой предел — б*о*льшие

Чтобы завершить расчеты, мы должны знать количество спринтов в выпуске. Для выпуска с жестко заданными сроками количество спринтов точно известно, и поэтому нам остается лишь умножить это количество на затраты, приходящиеся на каждый спринт, чтобы определить стоимость выпуска.

А в выпуске с жестко заданным объемом работ имеется диапазон спринтов. В предыдущем примере мы рассчитали диапазон от семи до девяти спринтов, и поэтому затраты на данный выпуск будут в семь–девять раз больше, чем затраты на каждый спринт. В большинстве организаций бюджет составляется исходя из верхнего предела в этом диапазоне затрат на выпуск, поскольку для завершения выпуска на самом деле может потребоваться девять спринтов.

Другой метод расчета затрат может быть применен в том случае, если известны исторические затраты на каждое очко за историю. Так, если имеются данные, обозначающие, на сколько очков была выполнена работа в течение предыдущего периода времени (например, года), эти данные следует разделить на затраты, приходящиеся на оплату труда членов команды, чтобы выяснить затраты на каждое очко. Если предположить, что одни и те же затраты на каждое очко распространяются на весь текущий выпуск, то можно приблизительно оценить затраты на 150-очковый

выпуск, умножив 150 очков на исторически сложившиеся затраты, приходящиеся на каждое очко, еще до первоначального планирования выпуска.

Сообщение о достигнутом прогрессе

Важной особенностью планирования выпуска является сообщение о достигнутом прогрессе. И хотя для такого сообщения можно воспользоваться любыми наглядными средствами, большинство команд употребляют для этой цели некоторую форму диаграммы сгорания или выгорания, наглядно отражающей состояние выпуска. Рассмотрим, каким образом сообщение о достигнутом прогрессе происходит в выпусках с жестко заданными сроками и объемом работ.

Сообщение о достигнутом прогрессе в выпуске с жестко заданным объемом работ

В выпуске с жестко заданным объемом работ у нас имеется представление об общем объеме работ, которые должны быть выполнены. В данном случае преследуется цель известить о том, насколько эти работы продвинулись к завершению.

Диаграмма сгорания выпуска с жестко заданным объемом работ

Диаграмма сгорания выпуска с жестко заданным объемом работ наглядно показывает общий объем незавершенных работ, который остается в каждом спринте, чтобы достичь цели текущего выпуска. По вертикальной оси на данном виде диаграммы в цифрах откладываются размеры элементов из задела продукта (как правило, в очках за историю или идеальных днях), а по горизонтальной оси — спринты (рис. 18.11).

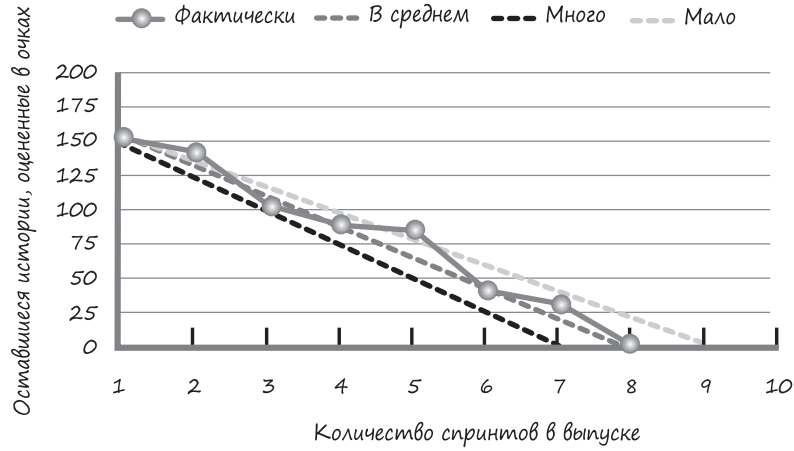


Рис. 18.11. Диаграмма сгорания выпуска с жестко заданным объемом работ

Если обратиться к примеру, рассмотренному ранее в этой главе, то в начале разработки (т.е. по завершении первоначального планирования выпуска или в начале первого спринта) имеется объем работ, оцениваемый в 150 очков. А в конце каждого спринта эта диаграмма обновляется, чтобы отражать объем оставшихся работ в текущем выпуске. Разность между объемом оставшихся работ в начале и в конце спринта определяет скорость выполнения спринта и обозначена на рис. 18.11 линией "Фактически".

На диаграмме сгорания можно также показать предполагаемые результаты. Так, все три линии на рис. 18.11 обозначают прогнозирование возможных сроков завершения выпуска, причем каждая из них соответствует прогнозируемой скорости работы команды. Если команда способна работать с высокой скоростью 22 очка за спринт, то она завершит выпуск за семь спринтов. Если же команда работает с низкой скоростью 18 очков за спринт, то для завершения выпуска ей потребуется девять спринтов. А если команда работает со средней скоростью 20 очков за спринт, то для завершения выпуска ей понадобится восемь спринтов. Но имеется несколько отклонений от основной диаграммы сгорания. Все они аналогичны тем, что отражают совокупный объем оставшихся работ для достижения цели выпуска.

Диаграмма выгорания выпуска с жестко заданными сроками

Диаграмма выгорания выпуска с жестко заданными сроками наглядно показывает общий объем работ в выпуске как рубеж прогресса, достигнутого в каждом спринте для достижения цели выпуска (рис. 18.12). По вертикальной и горизонтальной осям на диаграмме выгорания откладываются те же самые величины, что и на диаграмме сгорания выпуска.

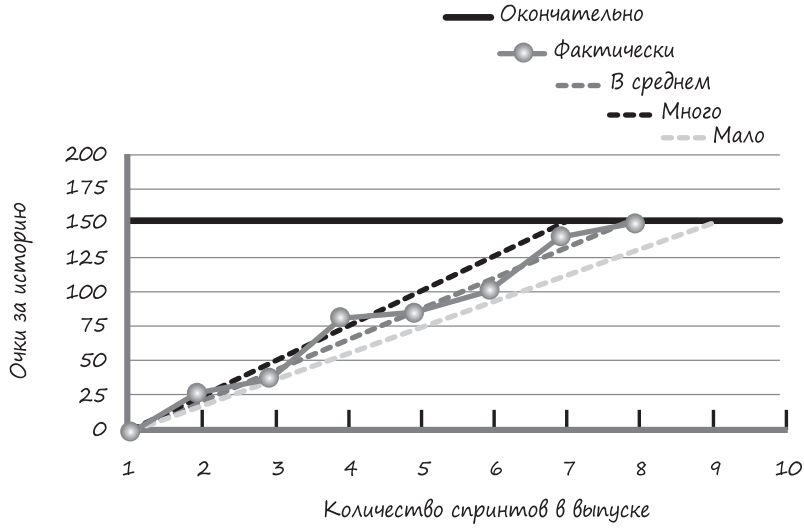


Рис. 18.12. Диаграмма выгорания выпуска с жестко заданными сроками

Как следует из диаграммы выгорания, приведенной на рис. 18.12, в конце каждого спринта постепенно увеличивается совокупное количество реализованных очков за историю из всего количества реализованных очков за историю. Назначение диаграммы выгорания — показать, как достичь окончательного количества очков за историю в текущем выпуске. Как и на диаграмме сгорания выпуска, на диаграмме выгорания приведены те же самые линии прогнозирования, показывающие вероятное количество спринтов, требующееся для достижения конечной цели выпуска.

Некоторые предпочитают пользоваться формой диаграммы выгорания, поскольку она может легко показать изменения в объеме работ над выпуском. Так, если увеличить объем работ в текущем выпуске, чтобы он действительно не имел жестко заданный объем работ, то в том спринте, где вводится новый объем работ, следует просто переместить вверх линию, обозначающую рубеж достигнутого прогресса (рис. 18.13). Изменения в объеме работ можно также показать на диаграмме сгорания выпуска [Cohn, Mike. 2006].

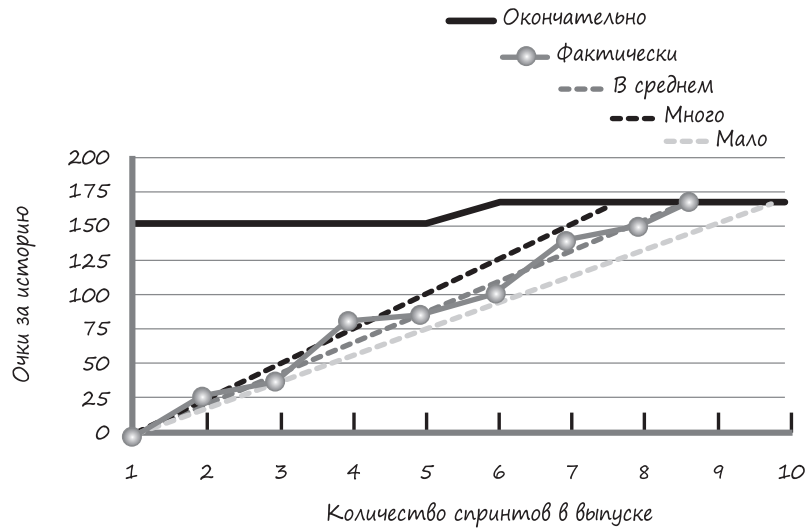


Рис. 18.13. Диаграмма выгорания выпуска с переменным объемом работ

Сообщение о достигнутом прогрессе в выпуске с жестко заданными сроками

При выпуске с жестко заданными сроками нам известно количество спринтов в нем, и поэтому наша цель — извещать о диапазоне функциональных средств, которые мы предполагаем завершить, а также о достигнутом прогрессе после каждого спринта в данном диапазоне. Традиционные диаграммы сгорания и выгорания

непригодны для планирования выпуска с жестко заданными сроками, потому что они предполагают, что нам известен общий объем сгорания или выгорания работ. А поскольку это планирование выпуска с жестко заданными сроками, то мы пытаемся рассчитывать и извещать со временем о сужении пределов объема работ, которые должны быть выполнены к жестко заданному сроку.

На рис. 18.9 показано, как наглядно представить диапазон функциональных средств, которые предполагается реализовать в выпуске с жестко заданными сроками. Если обновлять диаграммы, приведенные на рис. 18.9, после каждого спринта, то тем самым можно довольно эффективно сообщать о предполагаемом диапазоне функциональных средств, которые должны быть завершены к жестко заданным срокам выпуска. Это даст также возможность лучше понять, каким образом можно получить готовыми обязательные функциональные средства к намеченному сроку выпуска.

Если же требуется сохранить одну из диаграмм, отражающую исторический прогресс в достижении конечного объема работ, то с этой целью можно создать специальную диаграмму выгорания в форме, приведенной на рис. 18.14.

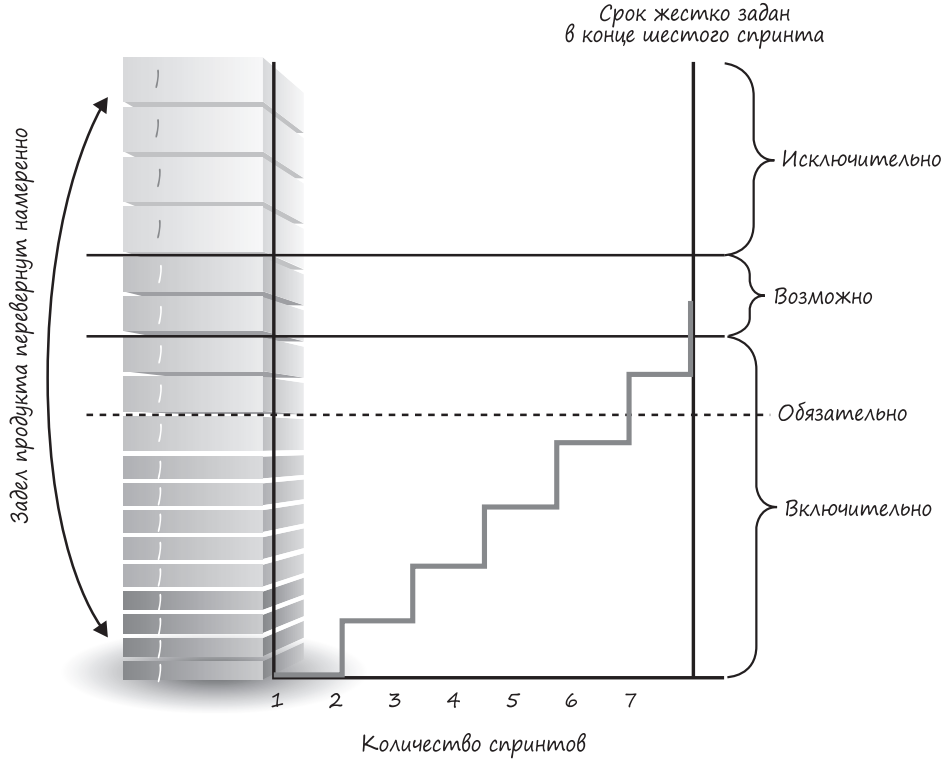


Рис. 18.14. Диаграмма выгорания выпуска с жестко заданными сроками и перевернутым заделом продукта

На этой диаграмме имеются те же самые элементы, что и на диаграммах, приведенных на рис. 18.9. Но на рис. 8.14 задел продукта намеренно перевернут вверх дном, чтобы элемент с наивысшим приоритетом, физически располагающийся на вершине задела продукта, оказался на его дне. Теперь элементы с более низким приоритетом находятся в верхней части задела продукта. Такое переворачивание задела продукта избавляет от необходимости знать объем работ над элементами задела продукта в текущем выпуске, что, как правило, требуется для построения традиционных диаграмм сгорания и выгорания выпуска.

На рассматриваемой здесь диаграмме показан диапазон функциональных средств, которые предполагается получить к концу шестого спринта (или началу седьмого). После выгорания каждого спринта на этой диаграмме показаны функциональные средства, завершенные в данном спринте. Таким образом, в конце первого спринта (начале второго спринта) на диаграмме выгорания появляется вертикальная линия, длина которой обозначает количество функциональных средств, завершенных в первом спринте. Такой метод извещения позволяет ясно видеть достигнутый прогресс в достижении целевого диапазона функциональных средств, а также в завершении обязательных функциональных средств. Ради простоты на этой диаграмме не показаны линии тенденций, наметившихся в ходе проектных работ, но их нетрудно ввести путем экстраполяции из предыдущих спринтов того объема работ, который в конечном итоге будет выполнен.

Заключение

В этой главе были подробно рассмотрены особенности планирования выпуска, в том числе время его проведения, участники и мероприятия, а также элементы составляемого в итоге плана выпуска. В ней также обсуждались подробности планирования выпуска с жестко заданными сроками и объемом работ и способы сообщения о прогрессе, достигнутом в ходе выпуска.

Этой главой завершается часть III данной книги. А в следующей главе обсуждение будет продолжено на следующем уровне планирования спринта. С нее начинается часть IV данной книги, посвященная видам деятельности, характерным для спринтов.

Часть IV

Проведение спринтов

Глава 19
Планирование спринта

Как правило, выпуск состоит из нескольких спринтов, в каждом из которых доставляется потребительская или пользовательская ценность. Каждый спринт начинается с его планирования, когда Scrum-команда собирается вместе для согласования цели спринта и определения того, что она способна выпустить в течение предстоящего спринта. В этой главе поясняется, каким образом планирование спринта вписывается в инфраструктуру Scrum и как оно выполняется.

Краткий обзор

Для формирования задела продукта могут потребоваться недели, а то и месяцы, чего вряд ли можно было бы добиться в течение одного короткого спринта. Чтобы определить наиболее важное подмножество элементов в заделе продукта, которые требуется реализовать в следующем спринте, Scrum-команда выполняет планирование спринта. В течение планирования спринта Scrum-команда согласовывает цель спринта, а команда разработчиков определяет конкретные элементы в заделе продукта, которые соответствуют данной цели и могут быть действительно выпущены в конце спринта. Чтобы приобрести уверенность в том, что именно можно выпустить, команда разработчиков составляет план реализации элементов из задела продукта. Совместно элементы задела продукта и план их реализации образуют задел спринта.

Временны*е* рамки

Планирование спринта выполняется неоднократно, оперативно и в начале каждого спринта, когда можно наилучшим образом воспользоваться доступной информацией, чтобы решить, над чем следует работать в предстоящем спринте (рис. 19.1). Планирование спринта, длящегося от двух недель до месяца, должно происходить не дольше четырех–восьми часов.

Рис. 19.1. Когда происходит планирование спринта

Участники

Все члены Scrum-команды сотрудничают вместе в течение планирования спринта. Владелец продукта делится с ними первоначальной целью спринта, представляет расставленный по приоритетам задел продукта и отвечает на любые вопросы, которые могут возникнуть у членов команды по поводу элементов задела спринта. Затем команда разработчиков прилежно работает над тем, чтобы определить те обязательства, которые она действительно готова взять на себя в конце планирования спринта. Scrum-мастер, выполняя роль наставника, надзирает за ходом планирования спринта, задает наводящие вопросы и оказывает всяческую помощь в достижения успешного результата. Но поскольку Scrum-мастер не руководит командой разработчиков, то он не может принять от ее имени решение о взятии обязательств на предстоящий спринт. Тем не менее Scrum-мастер может способствовать тому, чтобы эти обязательства были взяты реалистично и надлежащим образом.

Процесс

Процесс планирования спринта наглядно показан на рис. 19.2.

Планирование спринта опирается на ряд предпосылок, которыми команда разработчиков руководствуется, определяя ценность, которую она действительно готова доставить к концу планируемого спринта. Эти предпосылки описаны в табл. 19.1.

Первой и самой главной предпосылкой к планированию спринта является задел продукта, который был упорядочен перед планированием спринта таким образом, чтобы самые верхние элементы в заделе продукта соответствовали критерию готовности, установленному командой (см. главу 6). Как правило, это означает, что самые верхние элементы в заделе продукта обладают вполне определенными критериями соответствия и надлежащим образом нормированы по размеру, оценены и расставлены по приоритетам.

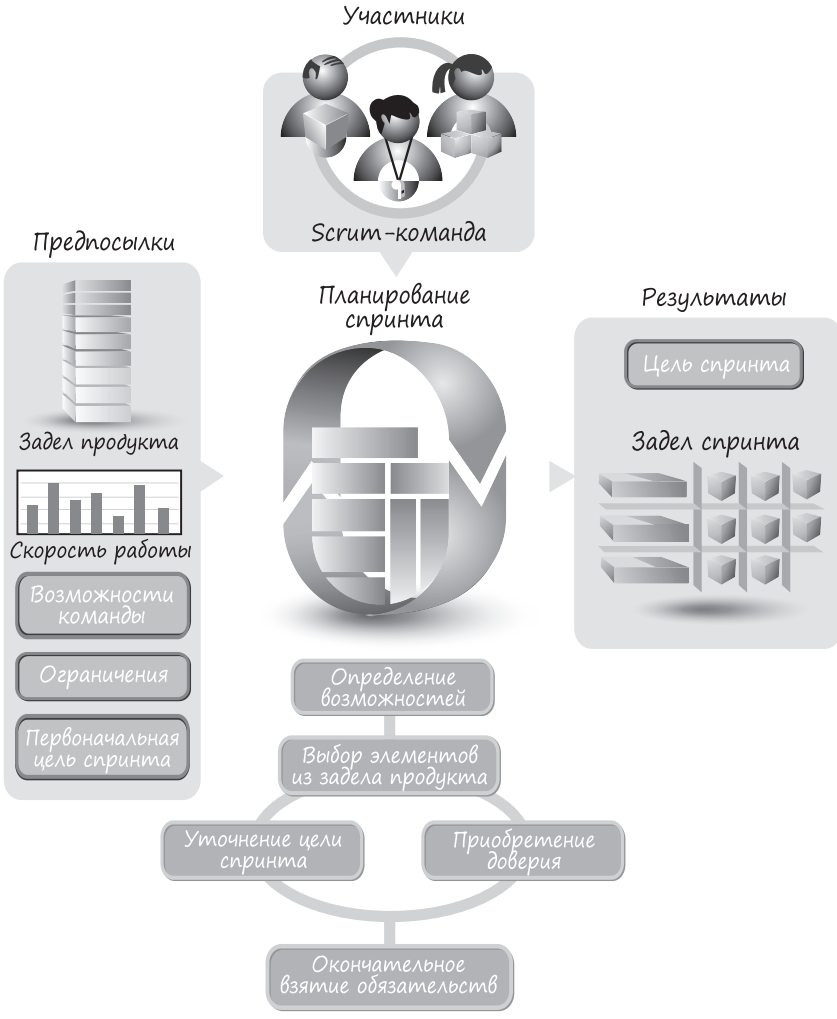


Рис. 19.2. Процесс планирования спринта

Таблица 19.1. Предпосылки к планированию спринта

Предпосылка	Описание
Задел продукта	Перед планированием спринта самые верхние элементы в заделе продукта упорядочены и приведены в *готовое* состояние
Скорость работы группы	Исторически сложившаяся скорость работы является показателем того объема работы, который команда способна выполнить в течение спринта
Ограничения	Это те коммерческие или технические ограничения, которые способны существенно повлиять на доставляемую командой ценность
Возможности команды	Возможности принимают во внимание состав команды, квалификацию каждого члена команды, а также степень их участия в предстоящем спринте
Первоначальная цель спринта	Это та коммерческая цель, которой владельцу продукта хотелось бы достичь в течение спринта

Владельцы продукта, привлекаемые к планированию спринта, вносят свои предложения по поводу того, что команда должна выпустить в конце данного спринта. При этом они могут иметь в виду конкретные высокоприоритетные элементы из задела продукта (например, предложение реализовать пять верхних элементов из задела продукта) или более общие пожелания (например, реализацию в конце спринта возможности для типичного пользователя выдавать простой запрос на ключевые слова). Зная цель спринта, команда может уравновесить противоборствующие приоритеты. Владелец продукта должен ясно изложить преследуемую им первоначальную цель спринта таким образом, чтобы не оказывать чрезмерного влияния на взятие командой разработчиков б*о*льших обязательств, чем она способна выполнить на самом деле.

Но даже если владелец и знает точно, чего он хочет, это еще не означает, что команда разработчиков способна достичь намеченной им цели в планируемом спринте. Реалистичные обязательства вырабатываются лишь в результате тесного сотрудничества, а иногда и переговоров, владельца продукта с членами команды разработчиков. Участники планирования спринта должны иметь возможность рассматривать и обсуждать альтернативные варианты формирования ценности и принимать решение о том, чего можно практически добиться в планируемом спринте, исходя из возможностей команды, прогнозируемой скорости ее работы или любых известных ограничений.

Чтобы приобрести уверенность в том, чего можно достичь в планируемом спринте, команда разработчиков составляет план достижения цели спринта. Совместно выбранные элементы задела продукта и план образуют задел спринта, как показано на рис. 19.2. Большинство команд разбивают намеченный

элемент задела продукта на ряд оцениваемых задач, которые совместно образуют план. Команды, которые предпочитают именно такой метод, как правило, следуют полезному правилу разбиения задач таким образом, чтобы ни одна из задач не требовала для своего выполнения больше восьми часов рабочего времени, хотя некоторые задачи могут быть и покрупнее. На данном уровне детализации у команды должно быть ясное представление о том, какие именно задачи действительно требуется и можно ли вообще выполнить в отведенное время. В конце планирования спринта команда разработчиков берет на себя обязательства в виде намеченной конечной цели спринта и его задела.

Методы планирования спринта

В этом разделе описываются два метода планирования спринта: одно- и двухэтапное планирование.

Двухэтапное планирование спринта

Один из методов планирования спринта состоит в том, чтобы разделить этот процесс на два этапа (рис. 19.3). На первом этапе, обозначающем “что делать”, команда разработчиков сначала определяет свои возможности выполнить работу, а затем, исходя из этих возможностей, прогнозирует выпуск отдельных элементов задела продукта в конце планируемого спринта. Так, если команда считает, что она способна выполнить работу объемом 40 очков за историю, она выбирает работу объемом около 40 очков за историю.

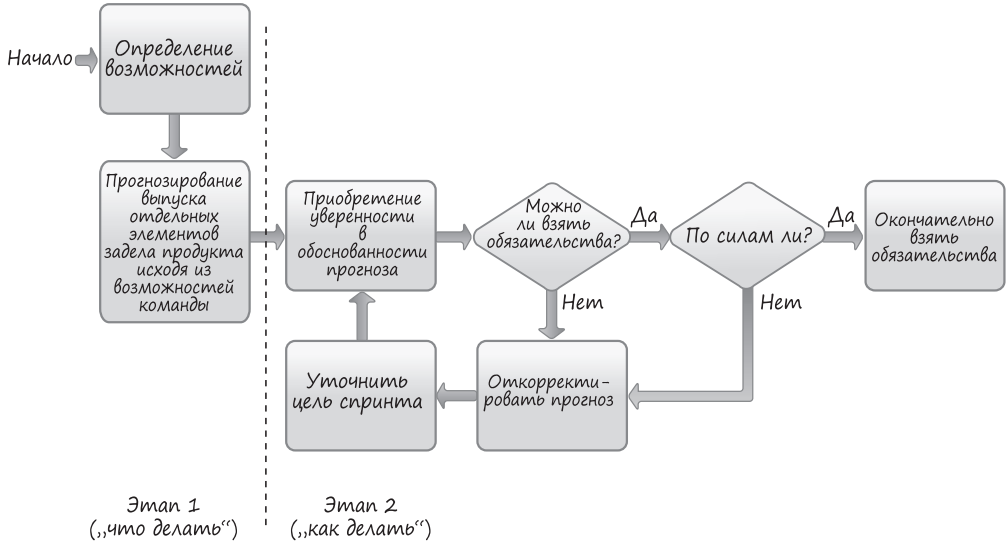


Рис. 19.3. Двухэтапный метод планирования спринта

На втором этапе, обозначающем “как делать”, команда приобретает уверенность в своих силах реализовать те элементы из задела продукта, которые она спрогнозировала на первом этапе, составляя план действий. Большинство команд составляют этот план, разбивая сначала элементы задела продукта на ряд задач, а затем оценивая (в рабочих часах) трудозатраты на выполнение каждой задачи. После этого команда сравнивает оценку трудозатрат на выполнение задач со своей производительностью в рабочих часах, чтобы выяснить, насколько реалистичными оказались обязательства, первоначально взятые командой.

Если команда обнаружит, что взяла на себя слишком много или, наоборот, слишком мало обязательств или же выбрала элементы, которые она не в состоянии на самом деле реализовать в одном и том же спринте при накладываемых на него ограничениях, она может откорректировать прогноз, а возможно, и уточнить цель спринта, чтобы согласовать ее со своими возможностями и имеющимися ограничениями. Как только прогноз команды уложится в рамки возможностей и ограничений, команда окончательно берет на себя обязательства, и на этом планирование спринта завершается.

Одноэтапное планирование спринта

Другой, чаще применяемый метод состоит в одноэтапном планировании спринта, в течение которого выбор элементов из задела продукта перемежается с приобретением уверенности в их выпуске. Этот метод наглядно показан на рис. 19.4.

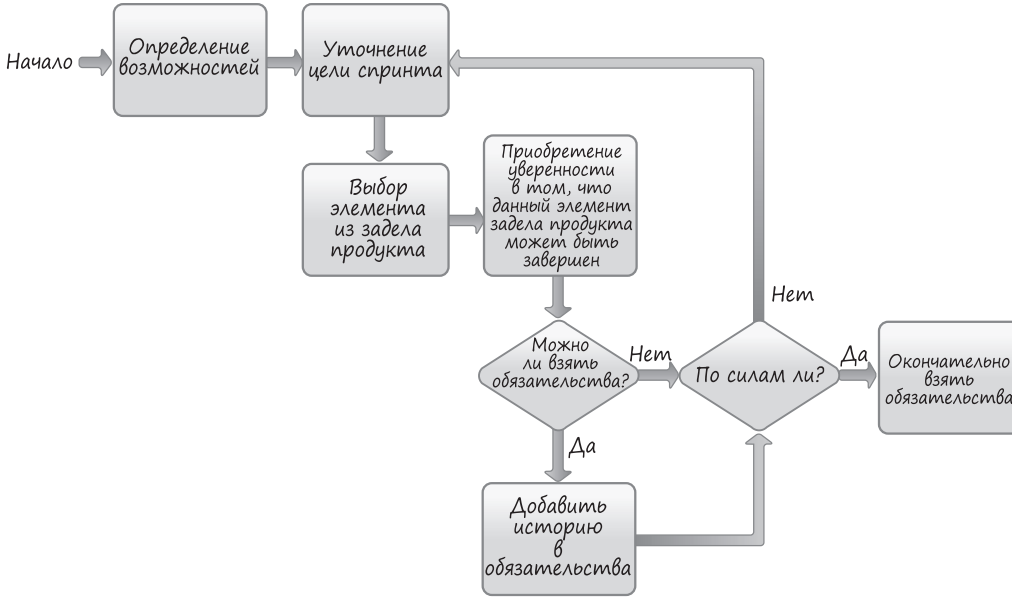


Рис. 19.4. Одноэтапный метод планирования спринта

Применяя данный метод, команда разработчиков начинает планирование спринта с определения своих возможностей выполнить работу. Исходя из имеющихся возможностей, команде, вероятно, придется уточнить цель спринта. Далее команда выбирает элемент из задела продукта и приобретает уверенность в том, что выбранный элемент обоснованно вписывается в рамки планируемого спринта, принимая во внимание другие элементы, уже включенные в обязательства, формируемые командой. Этот цикл повторяется до тех пор, пока команда не исчерпает свои возможности выполнить любую дополнительную работу. Тогда команда берет на себя окончательное обязательство, и на этом планирование спринта завершается.

Определение возможностей команды

Важным видом деятельности при планировании спринта является определение имеющихся возможностей команды выполнить работу в течение спринта. Зная свои возможности, Scrum-команда может определить то, что она способна выпустить в планируемом спринте.

Что означают возможности команды

На рис. 19.5 наглядно показаны различные факторы, оказывающие влияние на возможности команды работать над элементами задела продукта в течение предстоящего спринта. К числу этих факторов относится время, требующееся для других видов деятельности Scrum-команды, обязательства, не связанные с планируемым спринтом, личное отсутствие на работе и потребность в резерве.

Допустим, что команда выполняет двухнедельный спринт (в течение десяти рабочих дней). Следует сразу признать, что у команды на самом деле нет десяти рабочих дней, чтобы посвятить их полностью выполнению спринта. Как известно, в течение двухнедельного спринта требуется зарезервировать около дня для совместного планирования, подведения итогов и ретроспективы спринта. Известно также, что команда должна зарезервировать до 10% своего рабочего времени, чтобы помочь владельцу продукта упорядочить задел продукта, создавая и уточняя, оценивая и расставляя по приоритетам элементы в заделе продукта, чтобы подготовить их к работе.

Команда должна также определить, сколько времени следует зарезервировать для работы за пределами спринта, включая техническую поддержку текущего продукта, сопровождение другого продукта или иную работу, не связанную с текущим спринтом. Кроме того, команда должна также помнить, что в течение восьмичасового рабочего дня ее члены не заняты все время работой над элементами задела продукта. Ведь им приходится посещать совещания, отвечать на сообщения по электронной почте, делать перерывы в работе и т.д. Далее команде

нужно учитывать и личное отсутствие на работе членов команды разработчиков в течение спринта, поскольку это также сокращает общие возможности команды выполнить работу, которую они обязуются завершить в планируемом спринте.

Рис. 19.5. Возможности команды разработчиков в спринте

После того, как будет исключено время, посвящаемое другим видам деятельности в Scrum, работе за пределами спринта, личному отсутствию на работе, останется то, что и составляет возможности команды работать над элементами задела продукта в данном спринте. Но даже в этих общих возможностях следует сделать некоторый резерв на незапланированные действия. Например, любое оценивание не является совершенным, и поэтому элементы задела продукта могут оказаться более крупными, чем они казались поначалу. К тому же в течение спринта не все и не всегда идет гладко. Поэтому благоразумно сделать некоторый резерв на преодоление непредвиденных затруднений.

Команда может воспользоваться разными методами, чтобы определить практическую величину резерва (соответствующие примеры приведены в [Cohn, Mike. 2006]). На практике этот резерв может быть установлен эмпирически после того, как команда выполнит несколько спринтов и сможет лучше понять, какой именно резерв на преодоление неопределенности в разработке ей требуется установить. Как только такой резерв будет определен, команда может

окончательно выяснить свои возможности выполнить запланированную работу в течение спринта.

Оценивание возможностей команды в очках за историю

Какой единицей следует измерять возможности команды? Очевидным ответом на этот вопрос является выбор тех же самых единиц измерения, что и для элементов задела продукта (как правило, очков за историю или идеальных дней), или тех же самых единиц измерения, что и для задач в заделе спринта (обычно человеко-часов).

Скорость работы команды выражается в единицах измерения элементов из задела продукта (например, в очках за историю). Так, если выразить возможности команды в очках за историю, то возможности команды определяются таким же образом, как и прогнозирование целевой скорости работы команды на предстоящий спринт.

Чтобы сделать такое определение возможностей команды на предстоящий спринт, следует начать с долгосрочного оценивания средней скорости работы или предыдущей скорости работы в спринте (такой метод иногда еще называют "прогнозом погоды на сегодня"). Затем нужно рассмотреть, чем предстоящий спринт может отличаться от типичных предыдущих спринтов. В итоге должны быть определены благоразумно откорректированные возможности команды на предстоящий спринт.

Допустим, что средняя скорость работы команды составляет 40 очков за историю в течение двухнедельного спринта. Но планируемый спринт приходится на две последние недели декабря в Соединенных Штатах, а это означает, что многие члены команды разработчиков будут отсутствовать на работе в связи с рождественскими праздниками. Если команда примет 40 очков за среднюю скорость своей работы, то она возьмет на себя слишком много работы в данном спринте. Поэтому лучше принять 20 (или около того) очков в качестве более реалистичных возможностей работы команды в предстоящем спринте.

Оценивание возможностей команды в человеко-часах

Возможности команды можно выразить и в человеко-часах. В табл. 19.2 показано, каким образом возможности команды определяются в человеко-часах для выполнения работы на уровне отдельных задач в течение двухнедельного или десятидневного спринта.

Результаты расчетов возможностей команды, приведенные в табл. 19.2, получают следующим образом. Сначала члены команды разработчиков определяют, сколько дней у них имеется для работы в предстоящем спринте (время, недоступное для работы, равнозначно сегменту "Личное отсутствие на работе"

на рис. 19.5). Бетти и Райеш планируют посетить двухдневные учебные курсы, и поэтому каждый из них может быть занят в спринте только восемь рабочих дней. А Саймон собирается взять отгул на выходные дни, и в связи с этим он будет занят в спринте девять рабочих дней.

Таблица 19.2. Определение возможностей команды в человеко-часах

Работник	Количество имеющихся дней, кроме личного времени	Количество дней на другие виды деятельности в Scrum	Количество рабочих часов в день	Количество имеющихся человеко-часов
Хорхе	10	2	4–7	32–56
Бетти	8	2	5–6	30–36
Райеш	8	2	4–6	24–36
Саймон	9	2	2–3	14–21
Хейди	10	2	5–6	40–48
Итого:				140–197

Далее члены команды разработчиков определяют, сколько времени требуется зарезервировать для других видов деятельности в Scrum. Они резервируют целый день рабочего времени на проведение таких мероприятий, как планирование, подведение итогов и ретроспектива спринта. Кроме того, они выделяют время, необходимое на оказание владельцу продукта помощи в проведении мероприятий по упорядочению задела продукта. Вместе это составляет меньше двух дней, требующихся каждому работнику для выполнения работы на уровне отдельных задач.

Члены команды разработчиков определяют, сколько часов в день они могли бы посвятить работе в данном спринте. Каждый член команды выделяет часть времени с учетом дополнительной работы, не связанной с реализацией элементов задела спринта (эти работы соответствуют сегменту “Другие виды деятельности” на рис. 19.5). Например, Саймон лишь половину времени занят в работе над данным проектом, и поэтому он оценивает свое участие в нем два-три часа в день на протяжении планируемого спринта.

Как только будет учтено личное отсутствие членов команды на работе, другие виды деятельности в Scrum, а также обязательства, не связанные с планируемым спринтом, члены команды, перечисленные в табл. 19.2, оценивают свои возможности в пределах 140–197 человеко-часов для работы над задачами из задела планируемого спринта.

Такой команде не следует выбирать оценку своих возможностей в 197 человеко-часов, поскольку это не оставит ей никакого резерва в планируемом спринте. Ей лучше выбрать оценку своих возможностей чуть больше 140 человеко-часов, но, безусловно, меньше 197 человеко-часов, чтобы взять на себя реалистичные обязательства выполнить намеченный объем работ в течение данного спринта.

Отбор элементов из задела продукта

Любой из рассмотренных выше методов планирования спринта требует, чтобы из задела продукта были отобраны элементы, претендующие на включение в обязательства команды на спринт. Такой отбор может быть сделан несколькими способами. Если команда наметила цель спринта, то она должна отобрать из задела продукта те элементы, которые соответствуют данной цели. Если формальная цель спринта отсутствует, то по умолчанию элементы извлекаются из вершины задела продукта, начиная с самого верхнего элемента и далее вниз. И если команда не обязуется реализовать следующий после самого верхнего элемент в заделе продукта (возможно, из-за недостатка имеющихся у нее навыков), то она должна выбрать из задела продукта следующий далее подходящий элемент, который может быть реализован в пределах имеющихся ограничений.

Отбирая элементы из задела продукта для планируемого спринта я обычно руководствуюсь следующим эмпирическим правилом: не приступать к той работе, которую нельзя завершить. Так, если следующий элемент в заделе продукта слишком велик, чтобы завершить его в планируемом спринте, принимая во внимание другие элементы, реализация которых уже согласована, следует попытаться разбить этот элемент на два или более мелких элемента, которые могут представлять ценность для заказчиков, или рассмотреть возможность работы над другим элементом, который команда в состоянии реализовать. Кроме того, наличие *критерия готовности* препятствует выбору из задела продукта тех элементов, которые неясно определены, имеют неудовлетворенные ограничения на ресурсы или зависимости, мешающие завершению этих элементов в планируемом спринте.

Упомянутое выше правило опирается на следующие принципы: запасы незавершенных работ должны быть ограничены, а если работы начинаются, но не завершаются, то они приводят к самым разным формам расточительства. Оба эти принципа обсуждались в главе 4 при рассмотрении правила, не разрешающего изменять цель спринта в течение его выполнения. А если допустить перенос незавершенных элементов из одного спринта в другой, то тем самым не достигается цель получить прирост потенциально готового к поставке продукта в конце каждого без исключения спринта.

Приобретение уверенности

Приобрести уверенность в способности работать с прогнозируемой скоростью можно, в частности, оценив реалистичность взятых на себя обязательств. Так, если прогнозируемая скорость работы составляет 25 очков за историю, а команда посчитала, что сможет работать со скоростью 45 очков за историю, то такое решение команды должно настораживать. По крайней мере, членам

команды нужно хотя бы задаться вопросом: почему они считают, что взятые ими обязательства вполне достижимы? Во всяком случае, скорость работы служит отличным средством для сдерживания и уравновешивания обязательств.

Но употребление скорости работы в качестве единственного средства для установления доверия связано с риском не выполнить взятые на себя обязательства, несмотря на то, что цифры выглядят правильно. Так, если прогнозируемая скорость работы составляет 25 очков за историю, а команда обязуется в итоге работать со скоростью 21 очко за историю, то такое обязательство выглядит вполне обоснованным. Но лишь углубившись на уровень отдельных задач, можно точно узнать, реально ли завершить набор элементов задела продукта с итоговой скоростью 21 очко за историю. Ведь этому могут помешать имеющиеся зависимости, навыки, а также целый ряд других факторов, из-за которых команда не сможет завершить все намеченные на спринт элементы из задела продукта.

Большинство Scrum-команд приобретают необходимую степень уверенности, разбивая элементы задела продукта на отдельные задачи, чтобы завершить их по согласованному Scrum-командой критерию готовности. Эти задачи затем оцениваются (обычно в человеко-часах) и вычитаются из оценки возможностей команды (в тех же единицах измерения). Разбиение элементов из задела продукта на отдельные задачи является формой оперативного планирования мер, позволяющих достичь состояния готовности элементов при выполнении проектных работ.

В результате всех описанных выше действий возникает задел спринта. На рис. 19.6 приведен наглядный пример задела спринта, состоящего из четырех элементов, которые в целом оцениваются в 21 очко и могут быть завершены в конце спринта по общему мнению команды. Задел спринта демонстрирует также (в форме задач) план выпуска элементов задела продукта в соответствии с намеченной целью спринта. Является ли такое обязательство команды обоснованным?

Если группа спрогнозировала работать со скоростью 25 очков, а обязалась работать со скоростью 21 очко, то такое обязательство может считаться вполне обоснованным. Но попробуем уточнить это обязательство, перейдя на уровень задач. Сумма оценок всех задач, на которые разбиты четыре элемента задела продукта, составляет 150 человеко-часов. Допустим, что для выполнения данного спринта назначена команда, состав которой приведен в табл. 19.2. Возможности этой команды оцениваются в 140–197 человеко-часов. Обязательства выполнить работы объемом 150 человеко-часов кажутся на первый взгляд вполне обоснованными, поскольку у команды еще остается резерв на непредвиденные обстоятельства в течение планируемого спринта.

Но даже если 150 человеко-часов вполне вписываются в пределы от 140 до 197 человеко-часов, то это совсем не гарантирует обоснованность данного обязательства. Как следует из табл. 19.2, Саймон занят в данном проекте лишь два-три часа в течение девяти из десяти рабочих дней, отведенных на планируемый спринт. Это означает, что Саймон будет работать в данном спринте только

14–21 час. А что, если Саймон — единственный специалист, способный решать задачи проектирования ГПИ? В таком случае команда вряд ли сможет выполнить свои обязательства завершить четыре элемента из задела продукта, как показано на рис. 19.6, поскольку у нее не хватит возможностей справиться с проектированием ГПИ, если она исчерпает возможности Саймона.

Рис. 19.6. Задел спринта, демонстрирующий элементы из задела продукта и план выполнения задач

Несмотря на то что в Scrum задачи обычно не поручаются членам команды разработчиков при планировании спринта (обратите внимание на отсутствие имен членов команды разработчиков рядом с задачами на рис. 19.6), тем не менее, нужно хотя бы бегло оценить профессиональные возможности членов команды разработчиков, чтобы правильно взять на себя обязательства. Следовательно, даже если взятые обязательства удачно вписываются в оцененные пределы совокупных возможностей команды, это совсем не означает, что в отдельной области профессиональных навыков эти возможности не будут исчерпаны в течение спринта.

Именно поэтому некоторые команды предпочитают указывать рядом с наименованием каждой задачи имя работника, которому поручается работать над ней. Хотя это, как правило, излишняя и потенциально расточительная мера, поскольку в течение спринта могут произойти непредвиденные события, вынуждающие перепоручить задачи другим работникам. Такое перепоручение задач может оказаться даже вредным, если оно приводит к тому, что команда меньше владеет задачами. Поэтому лучше позволить членам команды разработчиков выбирать работу рационально и оперативно в течение спринта, как обсуждается в главе 20.

Уточнение цели спринта

Цель спринта подытоживает коммерческое назначение и ценность спринта. Владелец продукта должен подойти к планированию спринта с первоначальной его целью. Но первоначальная цель может быть уточнена в процессе планирования спринта, когда его участники совместно определяют то, что можно действительно выпустить.

Окончательное взятие обязательств

В завершение планирования спринта команда разработчиков окончательно берет на себя обязательства доставить коммерческую ценность в конце спринта. Эти обязательства воплощаются в цели спринта и тех элементах, которые были отобраны для него в заделе продукта.

Как упоминалось в главе 2, некоторые предпочитают пользоваться термином *прогноз* для обозначения коммерческой ценности, которую команда разработчиков считает возможным произвести к концу спринта. Я же предпочитаю пользоваться термином *обязательство*. Но независимо от употребляемого термина описанные выше методы планирования спринта остаются прежними.

Некоторые отличия в этих терминах могут оказать влияние только на объем работ, определяемый командой разработчиков, исходя из ее возможностей, а также на то, как Scrum-команда обращается с новой информацией, поступающей в течение спринта (подробнее об изменениях в ходе выполнения спринта см. в главе 4).

Заключение

В этой главе был подробно описан процесс планирования спринта. В частности, в ней обсуждалось, когда происходит планирование спринта и кто в нем должен принимать участие, а также рассматривались два разных метода такого планирования. В соответствии с первым методом команда сначала отбирает ряд элементов из задела продукта, а затем приобретает уверенность в том, что она действительно способна выпустить весь этот набор. А второй метод сочетает в себе отбор элементов из задела продукта с приобретением уверенности в том, что элемент может быть введен в постепенно нарастающие обязательства. А в следующей главе подробно обсуждается, каким образом спринты выполняются, как только они будут спланированы.

Глава 20
Выполнение спринта

Выполнение спринта — эта вид деятельности, который Scrum-команда осуществляет, чтобы достичь цели спринта. В этой главе основное внимание уделяется принципам и методам, которыми Scrum-команда руководствуется в своих планах, действиях и общении в ходе выполнения спринта.

Краткий обзор

Спринт выполняется подобно отдельному мини-проекту. В течение спринта выполняется вся работа, требующаяся для получения прироста потенциально готового к поставке продукта.

Временны*е* рамки

Выполнение спринта отнимает бо*л*ьшую часть времени, отводимого на спринт. Оно начинается после планирования спринта и завершается, когда начинается подведение итогов спринта (рис. 20.1). В частности, выполнение двухнедельного спринта может отнять от восьми до десяти рабочих дней.

Рис. 20.1. Когда происходит выполнение спринта

Участники

В ходе выполнения спринта члены команды разработчиков самоорганизуются и определяют лучший способ достижения цели, намеченной при планировании спринта. Scrum-мастер принимает участие в этом процессе как наставник, координатор и устранитель препятствий, делая все возможное, чтобы помочь команде успешно завершить спринт. Но Scrum-мастер не поручает работу команде и не указывает, как ей выполнять свою работу. Самоорганизующаяся команда решает все эти вопросы самостоятельно. А владелец продукта готов отвечать на вопросы, возникающие в ходе выполнения спринта, чтобы проанализировать промежуточные результаты работы, предоставить команде ответную реакцию, обсудить коррективы, которые придется внести в цель спринта, если того потребуют обстоятельства, а также проверить, насколько реализованные элементы задела отвечают критериям соответствия.

Процесс

Процесс выполнения спринта наглядно показан на рис. 20.2.

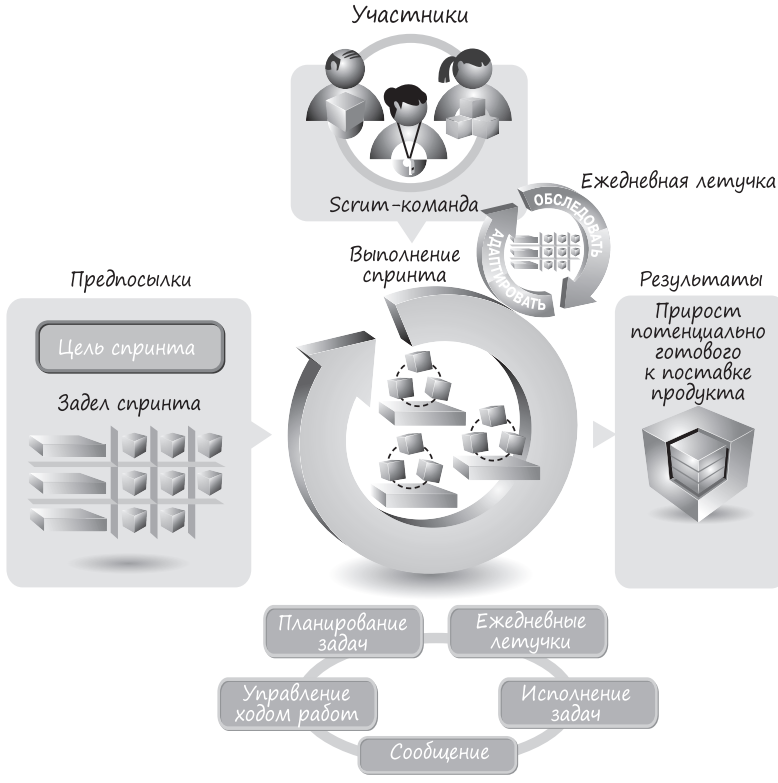


Рис. 20.2. Процесс выполнения спринта

Предпосылками для выполнения спринта являются цель и задел спринта, установленные в процессе его планирования. А конечным результатом выполнения спринта является прирост потенциально готового к поставке продукта, который представляет собой ряд элементов задела продукта, завершенных с высокой степенью уверенности, причем каждый элемент должен отвечать критерию готовности, согласованному Scrum-командой (см. главу 4). Выполнение спринта включает в себя планирование, управление, исполнение и сообщение о работе, необходимой для создания рабочих и проверенных функциональных средств.

Планирование выполнения спринта

При планировании спринта команда составляет *план* достижения цели спринта. Большинство команд создают задел спринта, в котором, как правило, перечисляются элементы из задела продукта, а также связанные с ними задачи и оцененные трудозатраты в человеко-часах (см. рис. 19.6 в главе 19). Несмотря на то что команда, вероятно, могла бы создать на уровне отдельных задач полноценный план выполнения спринта, равнозначный проектному плану для спринта (возможно, в форме диаграммы Гантта), это вряд ли было бы экономически оправданно.

Во-первых, не совсем ясно, действительно ли нужна команде из пяти–девяти человек диаграмма Гантта, предписывающая, кто и какую работу должен делать в следующем краткосрочном спринте. И во-вторых, даже если команде и потребуется составить диаграмму Гантта, она все равно окажется неточной вскоре после того, как команда приступит к работе. Выполнение спринта — это стадия, на которой, собственно, и начинается реальная разработка. Массовый эффект обучения возникает в результате построения и тестирования чего-то конкретного. Такое обучение способно нарушить даже самый идеально составленный план. В итоге команда, потратив драгоценное время на составление такого плана, потеряет еще больше времени на его изменение, чтобы отразить в нем реальное положение дел в ходе выполнения спринта.

Разумеется, некоторое предварительное планирование может быть полезным для выявления важных зависимостей на уровне отдельных задач. Так, если заранее известно, что функциональное средство, разрабатываемое в течение спринта, должно быть подвергнуто специальному двухдневному нагрузочному испытанию, команде было бы благоразумнее всего распределить работу таким образом, чтобы подобное испытание началось хотя бы за два дня до того, как завершится выполнение спринта.

Неплохим принципом выполнения спринта может стать рациональный подход к планированию на уровне отдельных задач вместо того, чтобы пытаться заранее составить полный план выполнения работ [Goldberg, Adele, and Kenneth S. Rubin. 1995]. Непрерывающееся планирование задач по ходу выполнения

спринта позволяет команде лучше приспосабливаться к обстоятельствам, возникающим в течение спринта.

Управление ходом работ

В процессе выполнения спринта управление ходом работ для достижения цели спринта входит в обязанности команды. Она должна принимать решения относительно объема параллельно выполняемых работ, начала работы над конкретным элементом, организации работы на уровне отдельных задач, характера выполняемых работ, а также распределения работы среди членов команды разработчиков.

Решая все эти вопросы управления ходом работ, команды должны отказаться от старых привычек. В частности, не следует пытаться загрузить работой полностью каждого члена команды (отрицательные последствия такой организации труда описаны в главе 2), выполнять работу последовательно или сосредоточивать внимание каждого работника только на его части проектного решения.

Параллельная работа и роение

Важной частью управления ходом работ является определение в заделе продукта количества элементов, над которыми команда должна работать параллельно, чтобы доставить как можно бо́льшую ценность в конце спринта. Но если работать сразу над слишком большим количеством элементов, то членам команды разработчиков придется перейти в многозадачный режим работы, а это, в свою очередь, увеличит время, требующееся для завершения отдельных элементов, и, скорее всего, понизит их качество. На рис. 20.3 приведен простой пример, который я обычно привожу на своих учебных курсах, чтобы продемонстрировать, во что обходится работа в многозадачном режиме.

В данном примере преследуется цель заполнить две одинаковые таблицы латинскими буквами от **`a`** до **`j`**, арабскими цифрами от **`1`** до **`10`** и римскими цифрами от **`i`** до **`x`**. Одна таблица заполняется по одной строке за раз, а другая — по одному столбцу за раз. Первая таблица представляет многозадачный режим работы, когда сначала выполняется задача заполнения ячеек таблицы латинскими буквами, затем задача их заполнения арабскими цифрами и далее задача заполнения римскими цифрами, после чего вся последовательность действий повторяется с новой буквы, арабской и римской цифры. А вторая таблица представляет однозадачный режим работы, когда выполняется одна задача заполнения ячеек таблицы только латинскими буквами, арабскими и римскими цифрами.

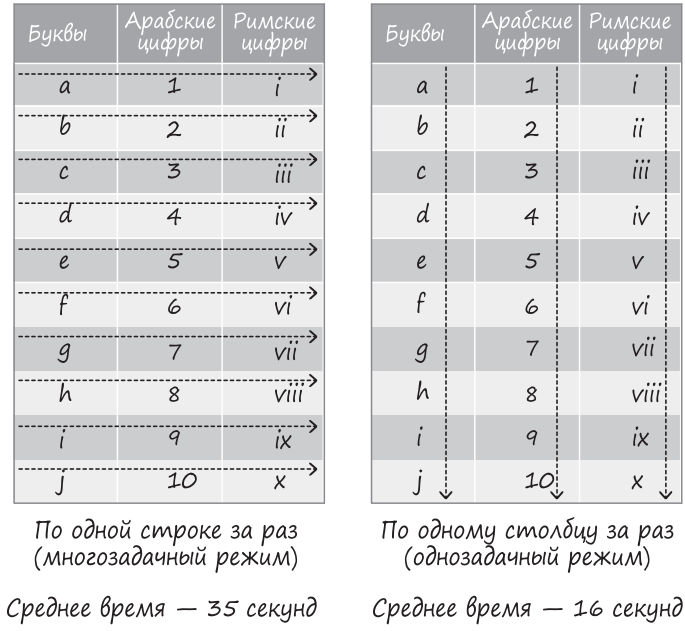


Рис. 20.3. Затраты на работу в многозадачном режиме

Типичные результаты, приведенные на рис. 20.3, означают, что большинство людей заполняют таблицу по столбцам почти в два раза быстрее, чем по строкам. Можете убедиться в этом сами! А если люди совершают при этом какие-нибудь ошибки, то они чаще всего совершают их при заполнении таблицы по строкам. И даже на таком простом примере явно видны издержки, связанные с работой в многозадачном режиме. Нетрудно представить убытки, которые может нанести организация работы над сложным проектом в многозадачном режиме.

Одновременная работа над слишком малым количеством элементов из задела продукта является столь же расточительной, как и одновременная работа над слишком большим количеством элементов. Это приводит к недоиспользованию профессиональных навыков и возможностей членов команды разработчиков, а следовательно, к выполнению меньшего объема работы и доставке меньшей ценности.

Для поиска золотой середины командам рекомендуется работать над таким количеством элементов из задела продукта, которое позволяет эффективно использовать Т-образные навыки (см. главу 11) и имеющиеся возможности членов команды разработчиков (см. главу 19), но не перегружая их работой. Самое главное — сократить, с одной стороны, время, требующееся для завершения отдельных элементов, а с другой стороны, доставить как можно бо́льшую общую ценность в течение спринта (как правило, количество элементов, завершенных в течение спринта).

Для описания такого метода организации труда нередко употребляется термин *роение*, когда члены команды разработчиков собираются вместе со своими возможностями для работы над элементом, чтобы завершить то, что уже было начато, прежде чем перейти к работе над новыми элементами. Команды с мушкетерскими отношениями и в какой-то степени Т-образными навыками собираются в рой и вместе разрешают затруднения, возникающие в работе. А в тех командах, где по-прежнему мыслят категориями отдельных ролей, одни члены вырываются в работе вперед, тогда как другие погрязают в незавершенной работе. В таких командах с отчетливым распределением ролей принято думать, что тестировщики должны сами завершить свою работу, а программисты — свою работу, и поэтому, завершив работу над одним функциональным средством, они сразу же переходят к другому средству. А члены команд, придерживающихся метода роения, ясно понимают, что лучше сосредоточить усилия на завершении тестирования одного функционального средства, чем спешить начать работу над следующим по очереди функциональным средством.

Некоторые ошибочно считают, что роение — это стратегия, обеспечивающая полную занятость членов команды разработчиков. Но ведь это не является целью роения. Если бы потребовалось обеспечить полную занятость членов команды разработчиков, достаточно было бы начать работу сразу над всеми элементами из задела продукта! Но почему мы этого не делаем? А потому, что для этого придется интенсивно работать в многозадачном режиме, что в конечном итоге замедлит поток завершенных элементов. С другой стороны, роение помогает команде сохранять свое внимание на главной цели спринта, а не на отдельных задачах. Это означает, что команда может делать больше и быстрее.

Несмотря на то что роение предполагает одновременную работу над меньшим количеством элементов, это совсем не означает, что работать над элементами задела продукта нужно по очереди. В конкретном контексте работа над элементами задела продукта по очереди может оказаться правильным решением, но было бы весьма неосторожно сказать, что все члены команды разработчиков должны совместно работать по очереди над отдельными элементами. Разное количество элементов может оказаться подходящим при рассмотрении объема работ, которые требуется выполнить, профессиональных навыков членов команды разработчиков и прочих условий, существующих на тот момент, когда принимается решение приступить или отложить работу над очередным требующим реализации элементом.

Еще один потенциально опасный подход состоит в том, чтобы применить водопадную методику разработки на уровне спринта и трактовать его выполнение как водопадный мини-проект. Применяя такой подход, мы начинаем работать сразу над всеми элементами из задела продукта. Для этого нам нужно сначала проанализировать все элементы, реализуемые в данном спринте, затем спроектировать, далее запрограммировать и, наконец, протестировать все эти элементы (рис. 20.4).

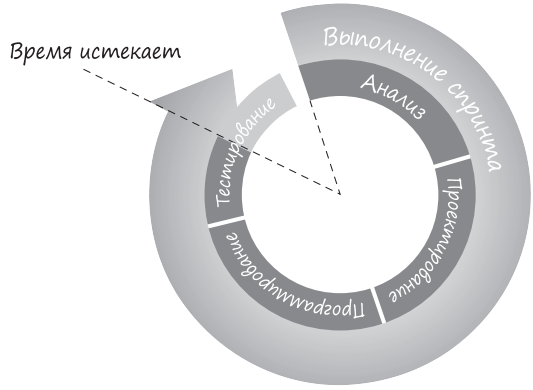


Рис. 20.4. Неудачная идея выполнять спринт как мини-проект по водопадной методике

Несмотря на то что такой подход может показаться вполне логичным, он весьма рискован. Что, если команда исчерпает время, отведенное на спринт, не проведя тестирование полностью? Получится ли в итоге прирост потенциально готового к поставке продукта? Нет. Ведь вполне обоснованный критерий готовности никогда не позволит признать готовыми непротестированные функциональные средства. Применяя водопадную методику выполнения мини-проекта, мы могли бы в итоге завершить каждое функциональное средство лишь на 90%, а не на 100%. Но ведь для владельца продукта не полностью готовая работа не имеет никакой экономической ценности.

С какой работы следует начинать выполнение спринта

Если предположить, что работа начинается не над всеми элементами из задела продукта сразу, то в какой-то момент команде придется решать, над каким из элементов следует работать дальше. Для этого проще всего выбрать следующий по приоритетности элемент по его расположению в заделе, которое обозначено владельцем продукта. Такой подход имеет следующее очевидное преимущество: любые элементы, не завершенные в течение спринта, должны иметь более низкий приоритет, чем завершенные элементы.

К сожалению, простейший подход не всегда оказывается самым подходящим, потому что ограничения на технические зависимости или профессиональные навыки и возможности могут заставить выбрать элементы в другом порядке. Поэтому команда разработчиков должна быть способна делать подобный выбор как можно более рационально.

Как организовать работу над отдельными задачами

Как только команда разработчиков решит приступить к работе над элементом из задела продукта, ей придется определить порядок работы над этим элементом на уровне отдельных задач. Если следовать водопадной методике разработки на уровне одного элемента задела продукта, то придется поочередно анализировать, проектировать, программировать и тестировать этот элемент.

Уверенность в том, что имеется единственный, преопределенный, логический порядок работы (например, построение перед тестированием), скрывает от команды возможность работать иначе и, возможно, даже эффективнее. Например, мне часто приходится слышать от новых команд вопрос вроде следующего: "Что делать нашим тестировщикам на ранней стадии спринта, когда они ожидают готовности функциональных средств к тестированию?" И я, как правило, отвечаю на этот вопрос таким образом: "В командах, практикующих проектирование, ориентированное на тестирование, тесты пишутся *до* разработки, а так называемые "тестировщики" *сначала* работают над функциональными средствами" [Crispin, Lisa, and Janet Gregory. 2009]!

Традиционным строгим распределением ролей страдает организация работы многих команд. Но вместо этого работу требуется организовать с акцентом на доставку ценности, чтобы рационально распределить выполнение задач среди членов команды разработчиков. Благодаря этому сводится к минимуму время простаивания работы и сокращается объем и частота, с которой члены команды разработчиков должны передавать одну работу другому. Это может, например, означать, что в первый день выполнения спринта организуются две пары членов команды, которые работают с интенсивным чередованием, выполняя быстрые, часто повторяющиеся циклы создания тестов и кода, проведения тестов и уточнения кода. При таком подходе работа идет плавно и не стопорится, а члены команды разработчиков с Т-образными навыками собираются в рой, чтобы завершить конкретный элемент.

Какая работа должна быть сделана

Какая работа на уровне отдельных задач должна быть сделана, чтобы завершить элемент из задела продукта? Окончательный ответ на этот вопрос дает сама команда. Владельцы продуктов и руководители должны доверить решение этого вопроса членам команды, которые являются ответственными специалистами и заинтересованы сделать работу качественно. Следовательно, члены команды разработчиков должны быть наделены полномочиями выполнять нужную работу по разработке новаторских решений экономически обоснованным способом.

Разумеется, владельцы продуктов и руководители должны также принять меры к тому, чтобы работа на уровне отдельных задач была сделана. Прежде

всего владелец продукта должен определить объем работ над каждым функциональным средством и критерии его приемки, являющиеся частью критерия готовности, описанного в главе 6. Тем самым устанавливаются границы для работы на уровне отдельных задач.

Кроме того, владельцы продуктов и руководители предъявляют коммерчески обоснованные требования к критерию готовности. Так, если коммерческие интересы требуют, чтобы функциональные средства, разрабатываемые в каждом спринте, выпускались для конечного потребителя по завершении спринта, то такое решение оказывает влияние на порядок выполнения командой работы на уровне отдельных задач. В этом случае больше работы потребуется для развертывания функциональных средств на рабочих серверах, чем для их построения и тестирования.

В целом владелец продукта должен сотрудничать с командой, чтобы экономически обоснованно принять технические решения, имеющие важные коммерческие последствия. Одни из этих решений включаются в более технические особенности критерия готовности. Например, Scrum-команда может совместно решить, что автоматизированные регрессионные тесты, имеющие экономические издержки и выгоды, важны и тем самым оказывают влияние на работу, выполняемую на уровне отдельных задач для создания и проведения автоматизированных тестов.

А другие решения зависят от конкретных функциональных средств. Нередко объем работы, которую команда должна выполнить над функциональным средством, определяется с известной степенью гибкости. Например, несмотря на то, что улучшение или доводка функционального средства могут быть привлекательными с технической точки зрения, для владельца продукта они просто не стоят дополнительных затрат как в данный момент, так и вообще. С другой стороны, упрощение проектного решения или недобросовестное отношение к месту, времени и порядку проведения тестов также влечет за собой экономические последствия, которые следует принимать во внимание (подробнее о накапливании технического долга см. в главе 8). Поэтому команда должна тесно сотрудничать с владельцем продукта, обсуждая все эти компромиссы и делая экономически обоснованный выбор.

Кто должен выполнять работу

Кто же должен работать над каждой задачей? Очевидно, тот, кто лучше, быстрее и качественнее всего выполнит такую работу. Но что, если такой специалист недоступен в настоящий момент? Возможно, он уже работает над другой, не менее важной задачей или отсутствует на работе по болезни, а задачу нужно выполнить безотлагательно.

На выбор подходящего человека для работы над задачей оказывают влияние многие факторы. Члены команды разработчиков несут коллективную ответственность за правильный выбор с учетом этих факторов.

Если члены команды разработчиков обладают Т-образными навыками, то несколько ее членов могут работать над каждой задачей. А если одни и те же навыки членов команды перекрываются, то она может собрать их в рой для совместной работы над теми задачами, которые нарушают нормальный ход реализации элементов из задела продукта при выполнении спринта, и благодаря этому команда работает более эффективно.

Ежедневная летучка

Проведение ежедневных летучек имеет решающее значение для постоянного обследования и адаптации команды с целью направить ход работ по более быстрому и гибкому пути к искомому решению. Как обсуждалось в главе 2, ежедневная летучка проводится один раз в сутки и длится около 15 минут. Она служит для обследования, синхронизации и адаптации планов на предстоящий рабочий день, помогая самоорганизующейся команде лучше выполнять свою работу.

Назначение ежедневных летучек — сосредоточить усилия членов команды разработчиков на достижении цели спринта и ознакомить их с общим положением дел, чтобы они могли совместно выяснить предстоящий объем работ, над какими элементам начинается работа и как лучше всего организовать совместную работу членов команды. Ежедневные летучки помогают также избежать откладывания неразрешенных вопросов. Если возникает вопрос, нарушающий нормальный ход работ, команда не должна откладывать его решение на следующий день. Если бы члены команды разработчиков собирались вместе лишь один раз в неделю, тем самым они лишили бы себя преимуществ, которые дает быстрая ответная реакция (см. главу 3). В целом ежедневные летучки играют важную роль в управлении ходом работ.

Исполнение задач — нормы инженерной практики

Члены команды разработчиков должны быть хорошо подготовлены технически, чтобы качественно выполнять свою работу. Это совсем не означает, что для гибкой разработки по методике Scrum требуется команда выдающихся мастеров своего дела. Тем не менее сам характер работы краткими по времени итерациями, где ожидается выпуск прироста потенциально готового к поставке продукта, вынуждает команды выполнять работу вовремя, умело контролируя технический долг. Если же членам команды разработчиков не хватает подходящих технических навыков, они, скорее всего, не смогут достичь того уровня

гибкости, который требуется для постоянной доставки коммерческой ценности в долгосрочной перспективе.

Если Scrum-команда формируется для разработки программного обеспечения, ее члены должны иметь опыт применения норм надлежащей инженерной практики в данной области. Речь идет не о каких-то особых навыках, а о тех навыках, которые применяются многие годы и имеют большое значение для успешной разработки по методике Scrum в частности и разработки программного обеспечения вообще. Это, например, навыки непрерывной интеграции, автоматизированного тестирования, реорганизации кода, разработки посредством тестирования и т.д. В настоящее время приверженцы гибкой разработки называют многие из этих норм инженерной практики *экстремальным программированием* [Beck, Kent, and Cynthia Andres. 2004], но большинство из этих норм появились задолго до понятия экстремального программирования (подмножество норм инженерной практики экстремального программирования приведено на рис. 20.5).

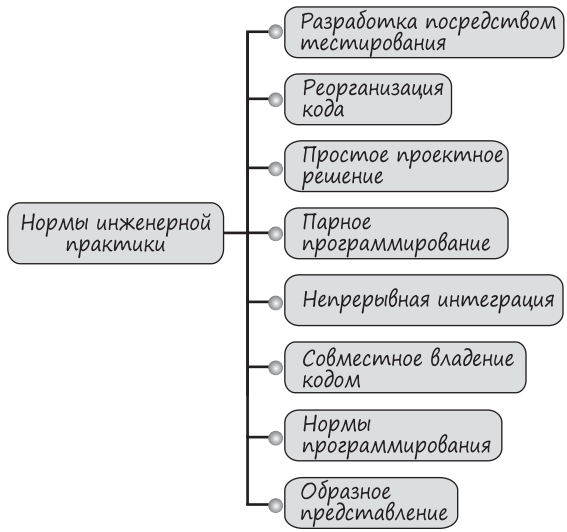


Рис. 20.5. Подмножество норм инженерной практики экстремального программирования

В качестве примера рассмотрим автоматизированное тестирование, которое требуется для поддержки ряда других норм практики, приведенных на рис. 20.5. Работа команд разработчиков, не уделяющих должного внимания автоматизации своих тестов, быстро замедляется, а количество тестов постоянно растет. В какой-то момент все время, отводимое на выполнение спринта, может быть потрачено на повторное проведение вручную регрессионных тестов разработанных ранее функциональных средств. В подобных случаях команда может вообще

отказаться от повторного проведения вручную тестов в каждом спринте, что может способствовать дальнейшему распространению дефектов, увеличивающих технический долг, а следовательно, и риск сорвать своевременный выпуск системы. Это также означает, что разработка перестанет быть гибкой до тех пор, пока тесты будут автоматизированы.

Аналогичные аргументы можно привести и в пользу остальных базовых норм практики. Большинство команд добиваются выгод от применения методики Scrum в долгосрочной перспективе только в том случае, если они придерживаются строгих норм инженерной практики, выполняя работу на уровне отдельных задач.

Сообщение

Преимущества работы в кратких временны*х* рамках небольшими группами заключается в том, что для сообщения о достигнутом прогрессе не требуется составлять сложные диаграммы и отчеты! И хотя для сообщения о достигнутом прогрессе не повредят весьма наглядные средства, большинство команд предпочитают пользоваться в определенном сочетании доской задач и диаграммой сгорания или выгорания в качестве основного *источника распространения информации*.

Доска задач

Доска задач является простым, но эффективным средством наглядного сообщения о прогрессе, достигнутом в спринте. Формально на доске задач отражаются изменения в состоянии задела спринта во времени (рис. 20.6).

На этой доске задач каждый элемент из задела продукта, работа над которым запланирована в течение спринта, отображается в виде ряда задач, необходимых для завершения этого элемента. Работа над всеми задачами начинается со столбца “Задачи для выполнения”. Как только команда определит, что уместно приступить к работе над элементом, члены команды выбирают задачи для данного элемента из упомянутого столбца и перемещают их в столбец “Выполняемые задачи”, чтобы показать, что работа над этими задачами началась. И как только задача будет завершена, она перемещается в столбец “Завершенные задачи”.

Безусловно, на рис. 20.6 приведен лишь один из примеров организации доски задач. Команда может выбрать для своей доски задач и другие столбцы, если она считает, что они будут лучше отражать изменения в ходе работ. В действительности в альтернативной методике гибкой разработки, называемой *Kanban* [Anderson, David J. 2010], применяется подобная детализированная доска задач, чтобы наглядно показать продвижение работ по различным стадиям.

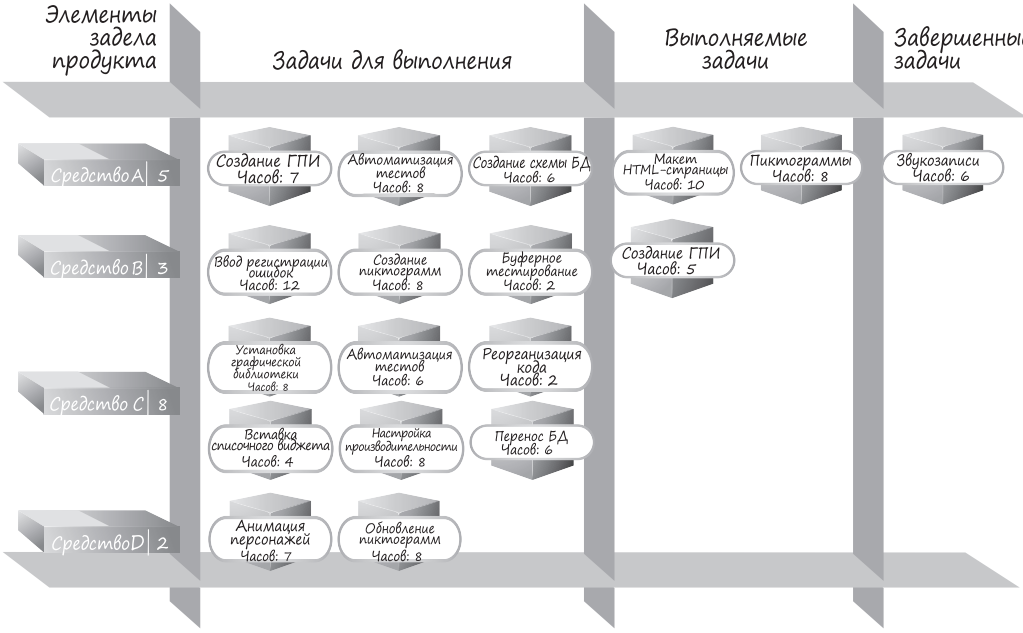


Рис. 20.6. Пример доски задач

Диаграмма сгорания спринта

Каждый день в течение спринта члены команды разработчиков обновляют оценки того объема работ, который остается выполнить для завершения задачи. Для наглядного представления этих данных можно было бы составить таблицу. Например, в табл. 20.1 показан пример 15-дневного спринта, в котором первоначально выполняются 30 задач, хотя в ней показаны не все дни задачи.

Таблица 20.1. Задел спринта с оценками объема работ, остающегося каждый рабочий день

Задачи	Д1	Д2	Д3	Д4	Д5	Д6	Д7	Д8	Д9	...	Д15
Задача 1	8	4	4	2							
Задача 2	12	8	16	14	9	6	2				
Задача 3	5	5	3	3	1						
Задача 4	7	7	7	5	10	6	3	1			
Задача 5	3	3	3	3	3	3	3				
Задача 6	14	14	14	14	14	14	14	8	4		
Задача 7						8	6	4	2		
Задачи 8–30	151	139	143	134	118	99	89	101	84		0
ИТОГО	200	180	190	175	155	130	115	113	90		0

В табл. 20.1 количество часов, оставшихся для завершения каждой задачи, следует общей тенденции уменьшения каждый рабочий день в течение спринта, поскольку работы над задачами постепенно выполняются и завершаются. Если же работа над задачей еще не начата, т.е. она все еще находится в столбце "Задачи для выполнения" на доске задач, размер этой задачи может не меняться изо дня в день до тех пор, пока не начнется работа над ней. Разумеется, задача может оказаться крупнее, чем предполагалось, и тогда ее размер может фактически увеличиваться с каждым рабочим днем (см., например, дни 4 и 5 выполнения задачи 4 в табл. 20.1) или оставаться прежним даже после того, как команда начнет работу над задачей, потому что работа вообще не велась в предыдущий рабочий день или же велась, но оценка оставшегося объема работ все равно не изменилась (см., например, дни 2 и 3 выполнения задачи 1 в табл. 20.1).

Новые задачи, связанные с элементами задела продукта, реализовать которые обязалась команда разработчиков, могут быть также введены в задел спринта в любой момент. Например, на шестой рабочий день команда обнаружила, что задача 7 отсутствует, и поэтому ввела ее в задел спринта (см. табл. 20.1). Нет никаких веских оснований, чтобы не вводить задачу в задел спринта. Ведь она представляет реальную работу, которую команда должна выполнить для завершения элемента из задела продукта, чтобы посчитать его готовым по согласованному командой критерию готовности. Разрешение вводить непредвиденные задачи в задел спринта нельзя считать лазейкой для внедрения новых видов работ в спринт. Оно просто подтверждает тот факт, что при планировании спринта невозможно было полностью определить весь ряд задач, необходимых для проектирования, построения, интеграции и тестирования элементов из задела продукта, реализовать которые обязалась команда разработчиков. По ходу выполнения работа становится понятнее, и поэтому можно и должно корректировать задел спринта.

Если составить график по данным из строки "ИТОГО" в табл. 20.1, где по дням указаны суммарные объемы работ в человеко-часах, оставшихся для завершения всех задач, то получится диаграмма сгорания спринта — еще один артефакт Scrum, сообщающий о достигнутом прогрессе (рис. 20.7).

В главе 18 рассматривались диаграммы сгорания выпусков, где по вертикальной оси откладываются числовые оценки объема оставшихся работ в очках за историю или идеальных днях, а по горизонтальной оси — количество спринтов (см. рис. 18.11). А на диаграммах сгорания спринтов по вертикальной оси откладываются числовые оценки объема оставшихся работ в человеко-часах, тогда как по горизонтальной оси — количество рабочих дней в спринте. Как показано на рис. 20.7, объем работ, остававшихся на первый рабочий день спринта, был оценен в 200 человеко-часов, а на пятнадцатый рабочий день — в нуль человеко-часов, поскольку это последний рабочий день трехнедельного спринта.

Каждый день эта диаграмма обновлялась, отражая суммарный объем работ, оставшихся для завершения всех выполняемых задач.

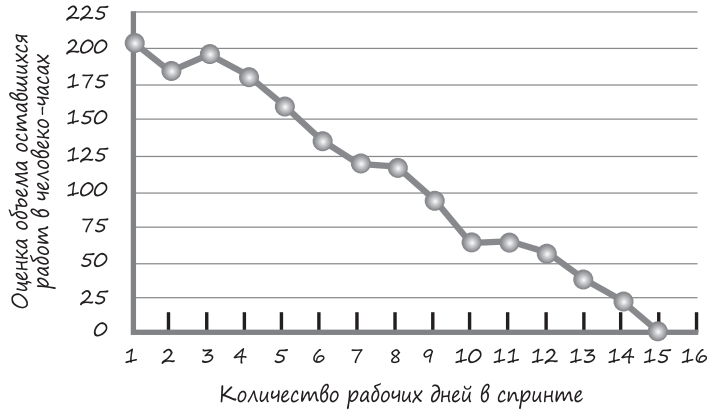


Рис. 20.7. Диаграмма сгорания спринта

Аналогично диаграммам сгорания выпусков, диаграммы сгорания спринтов полезны для отслеживания прогресса, достигнутого в работе. Они могут быть также использованы в качестве ведущего показателя для прогнозирования сроков завершения работ. В любой момент времени можно провести линию тенденции на основании хронологических данных, чтобы выяснить, когда, вероятнее всего, завершатся работы, если сохранится текущий темп их выполнения и объем (рис. 20.8).

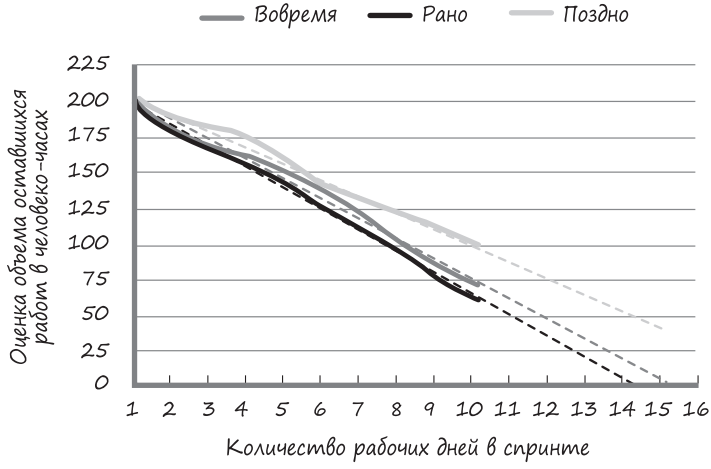


Рис. 20.8. Диаграмма сгорания спринта с линиями тенденций

На рис. 20.8 показаны три линии тенденции сгорания, иллюстрирующие разные ситуации. Если линия тенденции пересекает горизонтальную ось ближе к концу спринта, то можно сделать вывод, что работы завершатся вовремя. Если же эта линия пересекает горизонтальную ось значительно левее, то работы завершатся рано, а следовательно, можно взять дополнительную работу. Но если линия тенденции пересекает горизонтальную ось значительно правее, то работы завершатся поздно, а это означает следующее: работы не ведутся в предполагаемом темпе, был взят слишком большой объем работ или же и то и другое вместе. Проводя линии тенденций, мы получаем еще одну важную информацию, расширяющую наши знания о том, как управлять ходом работ в течение спринта.

При составлении задела спринта и диаграмм его сгорания всегда употребляется оцененный объем *оставшихся* работ. Они не фиксируют объем фактически выполненных работ. И хотя в Scrum нет особой необходимости фиксировать то, что было фактически сделано, в некоторых организациях такая потребность может возникнуть по таким не связанным непосредственно со Scrum причинам, как, например, учет затрат или уплата налогов.

Диаграмма выгорания спринта

Аналогично диаграмме выгорания выпуска, предназначенной для наглядного представления прогресса, достигнутого в выпуске, диаграмма выгорания спринта позволяет наглядно представить прогресс, достигнутый в спринте. Обе диаграммы отражают объем работ, завершенных для достижения намеченной цели: в одном случае — выпуска, а в другом — спринта. Пример диаграммы выгорания спринта приведен на рис. 20.9.

В диаграммах выгорания спринтов объем завершенных работ может быть представлен в человеко-часах (как и на диаграммах сгорания спринтов) или в очках за историю (как на рис. 20.9). Многие предпочитают употреблять в своих диаграммах выгорания спринтов очки за историю, поскольку в конце спринта для Scrum-команды на самом деле имеет значение только коммерчески ценная работа, сделанная в течение спринта, а измеряется она в очках за историю или идеальных днях, но не в часах работы над задачами.

Если же измерять завершенные элементы задела спринта в очках за историю, то можно сразу же получить ясное представление о том, как продвигается работа и насколько команда приблизилась к завершению элементов из задела продукта в течение спринта. Это положение наглядно иллюстрирует линия, обозначенная как “Неудачный ход работ” на рис. 20.9 (как правило, такая линия не включается в состав диаграммы выгорания спринта, но в данном примере это сделано для целей сравнения). Линия “Неудачный ход работ” показывает, каким

образом могла бы выглядеть диаграмма выгорания спринта, если бы команда начала спринт с одновременной работы над слишком большим количеством элементов из задела продукта, задержала их завершение до поздней стадии спринта и не сумела достичь его цели из-за снижения скорости параллельного выполнения слишком большого объема работ. Ведь работа над крупными элементами задела продукта отнимает немало времени для их завершения, а если команда предпринимает еще и другие действия в течение спринта, то в конечном итоге ход работ складывается неудачно.

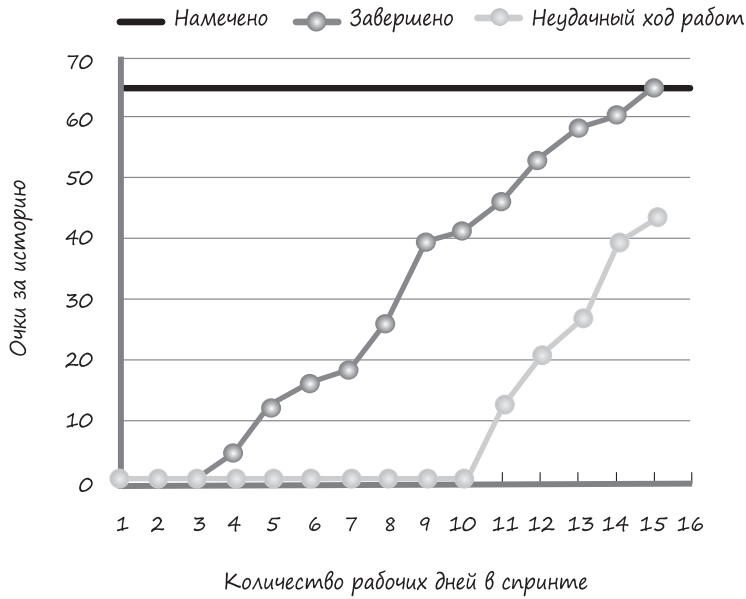


Рис. 20.9. Диаграмма выгорания спринта

Заключение

В этой главе обсуждались особенности выполнения спринта, на которое уходит большая часть времени, отводимого на спринт. В ней было подчеркнуто, что выполнение спринта происходит не по заранее составленному полному плану, в котором указаны виды работ, когда и кем они должны быть сделаны, а путем рационального и эффективного использования профессиональных навыков команд разработчиков, ответной реакции на уже завершенную работу и непредвиденные обстоятельства, возникающие в течение спринта. Но это совсем не означает, что выполнение спринта происходит хаотично. Напротив, выполняя спринт, команда руководствуется подходящими принципами управления ходом работ, чтобы определить, какой объем работ следует выполнять параллельно, с какой работы нужно начинать спринт, как организовать эту

работу, кто должен ее выполнять и какие трудозатраты для этого потребуются. В данном контексте обсуждалась ценность ежедневных совещаний в виде летучек как важного мероприятия для управления ходом работ. В этой главе было также упомянуто о значении норм надлежащей инженерной практики для достижения высокой степени гибкости процесса разработки. И в заключение этой главы были рассмотрены различные наглядные пособия, которыми Scrum-команда может сообщить о достигнутом прогрессе в работе, включая доску задач, диаграммы сгорания и выгорания спринтов. А в следующей главе будет рассмотрено мероприятие, называемое подведением итогов спринта и естественно следующее после выполнения спринта.

Глава 21
Подведение итогов спринта

Ближе к концу спринта команда проводит следующие мероприятия, имеющие большое значение для обследования и адаптации процесса гибкой разработки: подведение итогов и ретроспективу спринта. При подведении итогов спринта основное внимание уделяется самому продукту. А при ретроспективе спринта рассматривается процесс, которым команда пользуется для построения продукта.

В этой главе описывается подведение итогов спринта, его назначение, участники и работа по его подготовке. А в завершение этой главы рассматриваются некоторые вопросы, характерные для подведения итогов спринта.

Краткий обзор

При планировании спринта составляется план дальнейшей работы. При выполнении спринта эта работа делается, а при подведении итогов спринта осуществляется обследование (и адаптация) результатов проделанной работы (в виде прироста потенциально готового к поставке продукта). Подведение итогов спринта происходит ближе к концу каждого цикла спринта, сразу же после его выполнения и непосредственно перед (а иногда после) ретроспективой спринта (рис. 21.1).

Подведение итогов спринта дает всем, кто участвует в проектных работах, возможность обследовать и адаптировать то, что было сделано до сих пор. Кроме того, подведение итогов спринта дает ясное представление о текущем состоянии продукта, включая даже неудобную правду. Это самое подходящее время для того, чтобы задать вопросы, сделать наблюдения или предположения и обсудить наилучшие пути продвижения вперед в сложившихся обстоятельствах.

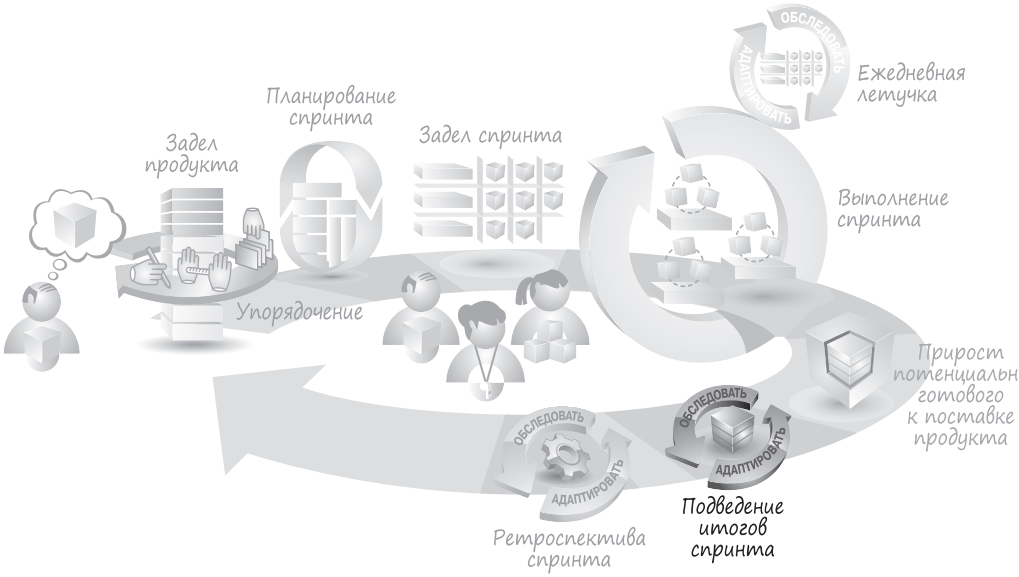


Рис. 21.1. Когда происходит подведение итогов спринта

Благодаря тому что подведение итогов спринта способствует успешной разработке продукции в организации, оно относится к одной из самых важных петель обучения в инфраструктуре Scrum. А благодаря краткосрочности спринтов такая петля совершается быстро, что дает возможность часто вносить коррективы в процесс разработки продукции, чтобы направить его в нужное русло. Если же откладывать эту ответную реакцию на более долгий срок, считая, что все идет по намеченному плану, то результат, вероятнее всего, будет уже привычным для многих: удивление, разочарование и неудовлетворенность.

Участники

Подведение итогов спринта дает Scrum-команде прекрасную возможность получить ответную реакцию от тех людей, которые, как правило, недоступны каждый день в течение спринта. А для этих людей подведение итогов спринта является первой возможностью просмотреть и обсудить работу, сделанную в течение спринта. Таким образом, в подведении итогов спринта должны участвовать все заинтересованные лица, которые могут представлять самые разные стороны, перечисляемые в табл. 21.1.

Таблица 21.1. Стороны, представляющие участников подведения итогов спринта

Сторона	Описание
Scrum-команда	Владелец продукта, Scrum-мастер и команда разработчиков должны непременно присутствовать на данном мероприятии, чтобы услышать ответную реакцию и ответить на вопросы, касающиеся полученного прироста продукта
Внутренние участники проекта	Владельцы коммерческого предприятия, высшее руководство и другие руководители, которые обязаны прежде всего следить за ходом работ, а следовательно, они могут предлагать коррективы для внесения в ход работ. Руководители внутренней разработки продукции, внутренние пользователи, эксперты в предметной области и руководители производства, функции которых связаны с разрабатываемым продуктом
Другие внутренние команды	Представители отделов сбыта, маркетинга, нормативно-правового соответствия, юристы и другие Scrum- и не Scrum-команды разработчиков, которые могут посещать подведение итогов спринта, чтобы дать свои отзывы, связанные с их производственными участками, или скоординировать работу своих команд с работой данной Scrum-команды
Внешние участники проекта	Внешние потребители, заказчики, пользователи и партнеры, которые могут дать свои отзывы, ценные как для данной Scrum-команды, так и для остальных участников мероприятия

Все члены Scrum-команды (владелец продукта, Scrum-мастер и команда разработчиков) должны присутствовать на каждом поведении итогов спринта, чтобы пояснить, что было сделано, ответить на возникшие вопросы и воспользоваться преимуществами ответной реакции из первых рук.

Данное мероприятие должны посещать и внутренние участники проекта, в том числе владельцы коммерческих предприятий, которые могут оплачивать разработку системы, высшее руководство, администраторы ресурсов и прочие руководители. Их отзывы очень важны для того, чтобы команда продвигалась к достижению экономически обоснованного результата. Кроме того, подведение итогов спринта дает удобную возможность узнать подлинное состояние проектных работ. А для внутренних проектных работ пользователями являются заинтересованные сотрудники организации. Их представитель должен присутствовать при подведении итогов спринта наряду с экспертами в предметной области, которые являются отличным источником ответной реакции на то, что было сделано в течение спринта.

Данное мероприятие могут посещать и другие сотрудники организации. На нем нередко присутствуют представители отделов сбыта и маркетинга. Они могут дать ценные отзывы о том, насколько успешным может оказаться появление разрабатываемого продукта на рынке. Другие участники, в том числе представители отделов технической поддержки, нормативно-правового соответствия и юристы, могут присутствовать при подведении итогов спринта, чтобы быть в курсе дел в команде, оказать ей своевременную помощь и точнее определить момент, когда им нужно начать свою работу, связанную с данной. А другие внутренние команды разработчиков, связанные с данными проектными работами, могут посылать на подведение итогов спринта своих представителей, чтобы они уяснили направление, в котором продвигается разработка продукта, сообщили о своих достижениях и высказали мнение о том, как это может повлиять на текущее состояние проектных работ.

На подведение итогов спринта целесообразно периодически приглашать внешних участников проекта, в том числе конкретных заказчиков или пользователей продукта, разрабатываемого командой. Когда они присутствуют на данном мероприятии, команда может получить от них непосредственные, а не косвенные отзывы, опосредованные через внутренних участников проекта. Приглашать внешних участников проекта на каждое подведение итогов спринта, возможно, и не стоит, особенно если заранее известно, что на предстоящем подведении итогов спринта намечается интенсивное внутреннее обсуждение, которое лучше провести только среди внутренних участников проекта. Если все же решено пригласить и внешних участников проекта, то они должны представлять конкретный круг потенциальных потребителей или пользователей. В целом, приглашая на подведение итогов спринта конкретных внешних участников проекта, следует руководствоваться здравым смыслом и желанием достичь намеченных целей данного мероприятия.

Подготовительная работа

Несмотря на то что подведение итогов спринта считается неформальным мероприятием, Scrum-команда должна провести хотя бы минимальную подготовительную работу к нему (рис. 21.2). Такая подготовительная работа включает в себя определение состава приглашенных, календарное планирование подведения итогов спринта, подтверждение готовности работы, выполняемой в течение спринта, подготовку к демонстрации достигнутых результатов при подведении итогов спринта, а также выбор лиц, которые будут вести это мероприятие и демонстрировать результаты спринта.

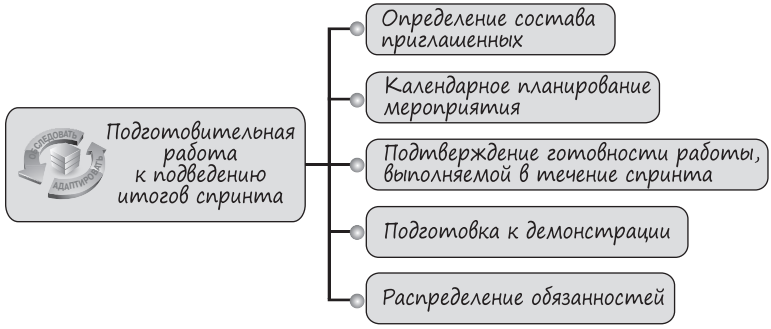


Рис. 21.2. Подготовительная работа к подведению итогов спринта

Определение состава приглашенных

Прежде всего Scrum-команда должна решить, кого следует регулярно приглашать на подведение итогов спринта. Самое главное — пригласить на данное мероприятие нужных людей, чтобы извлечь из него наибольшую пользу. И если нет никаких веских оснований не приглашать каких-то отдельных лиц или групп людей, то лучше охватить как можно более широкий круг лиц, предоставив им возможность голосовать ногами, — если данное мероприятие их заинтересует, они непременно посетят его.

Иногда команде приходится ограничивать круг приглашенных. Например, команда может быть заинтересована в присутствии на подведении итогов определенного лица или группы лиц, мнение которых имеет большое значение для анализа работы, проделанной в течение спринта. С другой стороны, команда, разрабатывающая функциональное средство для конкретного клиента в течение данного спринта, вряд ли может пригласить конкурентов этого клиента на подведение итогов спринта. Если имеется подозрение, что подобные ситуации могут возникнуть, следует сначала определить основной состав приглашенных, которые непременно должны присутствовать на каждом подведении итогов спринта, а затем приглашать отдельно определенные группы лиц или клиентов в зависимости от целей конкретного спринта.

Календарное планирование мероприятия

Подведение итогов спринта нужно планировать заранее, определяя место, время и продолжительность этого мероприятия. Из всех четырех регулярно повторяющихся в Scrum мероприятий (планирования спринта, ежедневной летучки, подведения итогов и ретроспективы спринта) заранее спланировать подведение итогов спринта труднее всего, поскольку к нему привлекается много людей, не входящих в состав Scrum-команды. А к остальным трем мероприятиям

привлекаются только члены Scrum-команды, и поэтому их удобно спланировать по отдельности.

Чтобы упростить календарное планирование подведения итогов спринта, его следует начать с определения времени, когда главным участникам проекта (основному кругу приглашенных) удобнее всего присутствовать на данном мероприятии (например, в 2 часа дня в пятницу), а затем спланировать все остальные мероприятия в спринте, исходя из этого жестко заданного времени. Как пояснялось в главе 4, если спринты имеют постоянную продолжительность (например, две недели), то все или хотя бы большинство совещаний по подведению итогов спринта можно спланировать с регулярной последовательностью (каждую вторую пятницу в 2 часа дня). Это дает следующее двойное преимущество: сокращение административного бремени и затрат и увеличение посещаемости данного мероприятия.

Длительность подведения итогов спринта зависит от нескольких факторов, включая продолжительность спринта, состав команды, а также участие в подведении итогов спринта других команд. Но, как правило, подведение итогов спринта длится не более четырех часов. Для многих команд оптимальным считается выделение одного часа на каждую неделю спринта. Иными словами, подведение итогов двухнедельного спринта должно длиться не больше двух часов, а четырехнедельного спринта — не больше четырех часов.

Подтверждение готовности работы, выполняемой в течение спринта

При подведении итогов спринта команде разрешается представлять на суд присутствующих *только* завершенную работу, которая удовлетворяет согласованному критерию готовности (подробнее об этом см. в главе 4). Это означает, что перед подведением итогов спринта кто-то должен определить, готов ли элемент задела продукта, иначе как Scrum-команда узнает, какие элементы ей представлять?

В конечном итоге ответственность за определение степени готовности работы возлагается на владельца продукта. Как упоминалось в главе 9, владелец продукта должен выполнять оперативный анализ готовности элементов из задела продукта по мере их выпуска в течение спринта. Таким образом, к моменту подведения итогов спринта команда будет точно знать, какие именно элементы завершены и готовы к представлению.

Не все согласны с тем, что владелец продукта должен просматривать выполненную работу перед подведением итогов спринта. Некоторые из практикующих методику Scrum настаивают на том, что владелец продукта должен просматривать выполненную работу и формально принимать ее только при подведении

итогов спринта. Они считают, что если разрешить владельцу продукта просматривать выполненную работу в течение спринта, он может потребовать внести в нее изменения, которые вместо уточнения изменят назначение работы, нарушая тем самым порядок выполнения спринта (см. главу 4).

И хотя в этом скрывается потенциальный риск, тем не менее преимущества заблаговременного просмотра выполненных работ владельцем продукта (и скорой ответной реакции) намного перевешивают любые недостатки. Более того, если владелец продукта впервые увидит работу, выполненную командой только при подведении итогов спринта, тогда может оказаться слишком поздно. Дело в том, что владелец продукта должен быть доступен в течение всего спринта для ответа на возникающие вопросы и уточнения степени готовности элементов из задела продукта. Выполняя эти обязанности, владелец продукта должен также анализировать постоянный прогресс в работе команды, реагируя своевременно и эффективно по затратам. Если же отложить эту ответную реакцию до подведения итогов спринта, то команда может выполнить ненужную работу, которая способна принести ей сплошное разочарование и упреки владельцу продукта в том, что он вовремя не указал команде на подобные недостатки в ее работе при выполнении спринта, не говоря уже о раздражении остальных участников проекта при подведении итогов спринта.

Но помимо этого, владелец продукта, отвергающий или подвергающий сомнению работу, представляемую при подведении итогов спринта, может оказаться по другую сторону баррикад от остальных членов Scrum-команды. В итоге все мероприятие может свестись к выявлению виновных среди участников проекта. Но ведь владелец продукта и команда разработчиков должны быть единой Scrum-командой при подведении итогов спринта.

Подготовка к демонстрации

Вся работа, выполненная командой в течение спринта (т.е. прирост потенциально готового к поставке продукта), должна быть готова к представлению при подведении итогов спринта, и поэтому подготовка к демонстрации не должна отнять много времени. Ее цель — обеспечить ясность и эффективность обследования и адаптации продукта, а не организовать показуху ради создания яркого эффекта.

Подведение итогов спринта предполагается как неформальное совещание с минимальными церемониями, но высокой ценностью. Вряд ли стоит тратить уйму времени на создание презентации в PowerPoint, поскольку такая презентация вместо демонстрации работающего программного обеспечения вызвала бы, по меньшей мере, недоумение у приглашенных на подведение итогов спринта. Ведь они могли бы подумать, что вместо демонстрации реальной работы им показывают лишь то, что должно быть сделано.

Большинство команд тратит, как правило, от 30 минут до 1 часа каждую неделю в течение спринта на подготовку к подведению итогов спринта. Кроме того, многие согласны с тем, что показывать следует лишь те артефакты, которые произведены как следствие достижения цели спринта.

Безусловно, из этого правила могут быть исключения. Мне приходилось сотрудничать с организациями, разрабатывавшими системы по контракту с армией США. И чаще всего подведение итогов спринтов посещали государственные служащие (чиновничьего ранга), тогда как высокие армейские начальники делали это лишь изредка. Ясно, что в подобных случаях команда тратила больше времени на более тщательную подготовку к подведению итогов спринта!

Распределение обязанностей

Перед подведением итогов спринта команде нужно решить, кто из членов Scrum-команды должен проводить это мероприятие и демонстрировать готовую работу. Как правило, эта обязанность поручается Scrum-мастеру, но владелец продукта может открыть совещание, вкратце сообщив собравшимся итоги спринта. Что же касается демонстрации завершенной работы, то я лично предпочитаю предоставлять каждому члену команды разработчиков возможность демонстрировать достигнутые результаты на очередном подведении итогов спринта, а не поручать это каждый раз одному и тому же лицу. Но я стараюсь не особенно вникать в то, кто именно должен это делать. Scrum-команда должна сама распределить обязанности, чтобы извлечь наибольшую пользу из данного мероприятия.

Метод

Процесс подведения итогов спринта наглядно показан на рис. 21.3. Предпосылками к подведению итогов спринта служат задел и/или цель спринта, а также прирост потенциально готового к поставке продукта, который команда фактически получила в течение спринта. А результатами подведения итогов спринта являются упорядоченный задел продукта и обновленный план выпуска.

Типичный метод подведения итогов спринта предполагает краткий обзор результатов, достигнутых в обсуждаемом спринте в сравнении с целью спринта, демонстрацию прироста потенциально готового к поставке продукта, обсуждение текущего состояния продукта и адаптацию дальнейшего направления работы над продуктом.

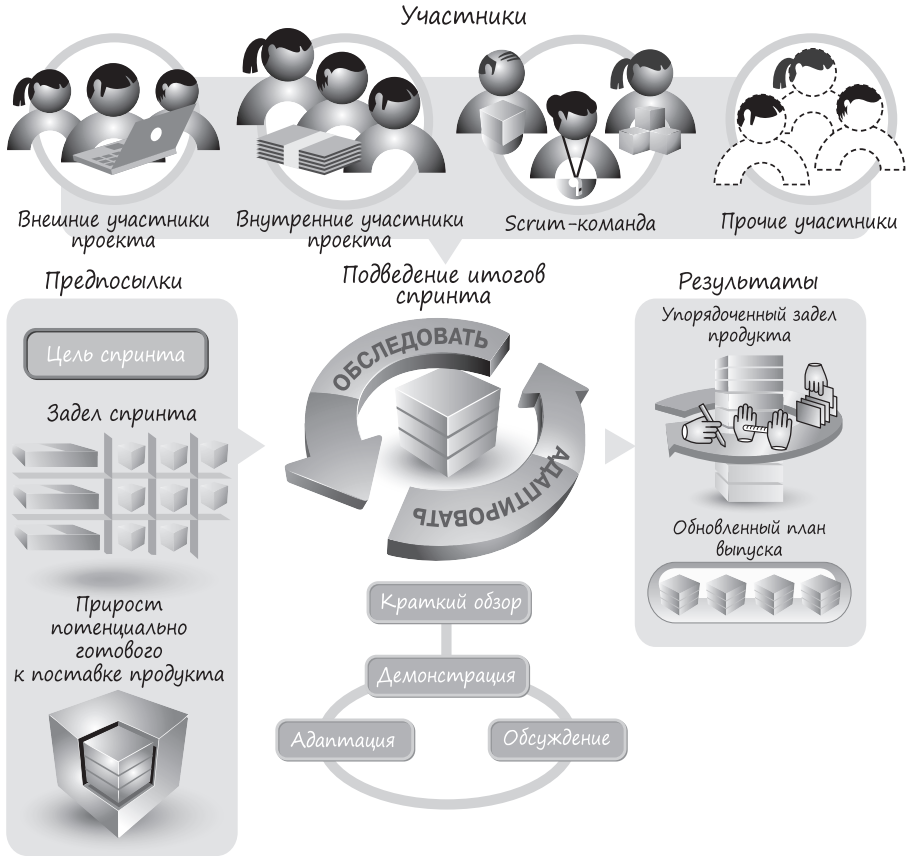


Рис. 21.3. Процесс подведения итогов спринта

Краткий обзор

Подведение итогов спринта начинается с того, что один из членов Scrum-команды (зачастую владелец продукта) представляет цель обсуждаемого спринта, связанные с ней элементы из задела продукта, а также делает краткий обзор прироста продукта, фактически достигнутого в течение спринта. Эта информация служит сводкой или кратким изложением результатов спринта в сравнении с его целью.

Если достигнутые результаты не соответствуют цели спринта, Scrum-команда дает пояснение причин. Очень важно, чтобы подведение итогов спринта проходило в атмосфере, исключающей выявление виновных. Если цель спринта не была достигнута, то все участники должны воздерживаться от обвинений. Ведь назначение подведения итогов спринта — показать сначала, что было действительно достигнуто, а затем, используя полученную информацию, наметить наилучший план дальнейших действий.

Демонстрация

Подведение итогов спринта часто называют не совсем верно “демонстрацией спринта” или просто “демонстрацией”. И хотя демонстрация весьма полезна при подведении итогов спринта, она не является его целью.

Наиболее важной особенностью подведения итогов спринта является серьезное обсуждение и сотрудничество участников этого мероприятия с целью выявить продуктивные способы адаптации процесса разработки и воспользоваться ими. А демонстрация того, что было фактически сделано, просто служит эффективным способом побудить к обсуждению чего-то конкретного. Ничто так не способствует обсуждению, как возможность увидеть в действии то, что является предметом обсуждения.

Как было определено при подготовке к подведению итогов спринта, один или несколько членов Scrum-команды должны продемонстрировать все соответствующие особенности прироста продукта, полученного в течение спринта. В некоторых организациях вроде игровых студий было бы еще эффективнее провести демонстрацию с привлечением самих участников проекта, дав им, например, возможность поиграть в игру на стадии того прироста данного продукта, который был получен в течение обсуждаемого спринта.

Но что, если демонстрировать особенно нечего? Если команда не сделала ничего, достойного показа, то подведение итогов спринта лучше сосредоточить на причинах, по которым ничего не было сделано и том влиянии, которое отсутствие прогресса в обсуждаемом спринте может оказать на дальнейшую работу. Но совсем другое дело, если то, что было сделано, нелегко продемонстрировать. Допустим, что команда выполняла только архитектурную работу в обсуждаемом спринте, т.е. создала так называемый “связующий код”. В таком случае команда разработчиков может выдвинуть аргумент, что демонстрировать связующий код не имеет смысла или непрактично. Но такой аргумент редко оказывается весомым. И вот почему.

Для того чтобы команда работала только над связующим кодом, она должна убедить владельца продукта, чтобы он разрешил работать в спринте только над техническими элементами задела продукта. Как пояснялось в главе 5, чтобы дать на это разрешение, владелец продукта должен ясно понимать ценность такой работы и знать, как определить ее готовность. Кроме того, большинство команд включают в свой критерий готовности положение о том, что Scrum-команда хорошо понимает, как продемонстрировать реализованный элемент при подведении итогов спринта.

У команды должен быть как минимум некоторый набор тестов, чтобы продемонстрировать готовую работу, удовлетворив владельца продукта. Такие тесты должны проходить, поскольку команда может демонстрировать только

завершенную работу при подведении итогов спринта. Таким образом, команда может хотя бы воспользоваться этими тестами для демонстрации достигнутых результатов при подведении итогов спринта. Как правило, члены команды разработчиков могут сделать даже больше, если хорошенько подумают. Если достигнутый результат не поддается демонстрации, это не является оправданием для его исключения из демонстрации.

Обсуждение

Демонстрация прироста продукта сосредоточивает внимание присутствующих на серьезном обсуждении увиденного. Участникам подведения итогов спринта настоятельно рекомендуется высказать свои наблюдения, дать комментарии и провести конструктивное обсуждение продукта и направления его разработки. Но подведение итогов спринта — это не место для разрешения серьезных затруднений. Этого рода деятельности лучше посвятить другое совещание.

Живое обсуждение позволяет участникам, не входящим в состав Scrum-команды, задавать вопросы, уяснить текущее состояние продукта и помочь дальнейшему направлению его разработки. В то же время члены Scrum-команды приобретают более глубокое понимание коммерческой и маркетинговой стороны разрабатываемого ими продукта, получая ответную реакцию на постепенное направление продукта на удовлетворение потребностей заказчиков или потребителей.

Адаптация

В ходе демонстрации и обсуждения команда может задавать вопросы и отвечать на них, в том числе и на перечисленные ниже. Постановка вопросов и ответы на них способствуют адаптации задела продукта и плана выпуска.

- Понравилось ли участникам проекта то, что они увидели?
- Желают ли они увидеть изменения?
- Насколько разрабатываемый продукт подходит для рынка или внутренних потребителей?
- Недостает ли разрабатываемому продукту какого-нибудь важного функционального средства?
- Не слишком ли много труда и средств вкладывается в разработку функциональных средств там, где этого не следовало бы делать?

В главе 6 пояснялось, как большинство команд естественным образом упорядочивают задел продукта при подведении итогов спринта. Как только все заинтересованные лица начинают лучше понимать текущее состояние проектных

работ и дальнейшее их направление, в заделе продукта нередко создаются новые элементы, переставляются по приоритетам существующие элементы или удаляются те из них, которые больше не нужны. Такое упорядочение может определять то, над чем команде придется работать в предстоящем спринте.

А как пояснялось в главе 18, упорядочение задела продукта, которое происходит при подведении итогов спринта, может оказать влияние и на более масштабные планы выпуска. Например, на основании обсуждения и выводов из подведения итогов спринта можно решить изменить один из главных показателей планирования выпуска: объем работ, сроки или бюджет. Например, рассматривая текущий прирост продукта, можно прийти к решению о необходимости прекратить работу над главным функциональным средством продукта, т.е. изменить объем работ. Такое решение по необходимости окажет влияние на текущий план выпуска. Подведение итогов спринта дает возможность определить в конце каждого отдельного спринта способы адаптации и реагирования на перемены, когда это уместно сделать.

Вопросы подведения итогов спринта

Тем не менее подведение итогов спринта не лишено трудных вопросов. Сотрудничая со многими организациями, применяющими методику Scrum, я выявил ряд общих вопросов, характерных для подведения итогов спринта, включая и те, что связаны с одобрениями, недостаточной посещаемостью и крупными проектами.

Одобрения

Одобрения могут быть затруднены при подведении итогов спринта. Прежде всего, возникает вопрос: насколько уместно одобрять (или утверждать) элементы задела продукта при подведении итогов спринта? Как упоминалось ранее, перед подведением итогов спринта владелец продукта проверяет готовность работы, выполненной в спринте, по критерию готовности. Следовательно, подведение итогов спринта не подходит для формального одобрения или утверждения элементов задела продукта. Напротив, они должны быть одобрены владельцем продукта еще до начала подведения итогов спринта.

Но допустим, что при подведении итогов спринта какой-нибудь участник проекта из руководства организации выразит свое несогласие, считая, что представленный элемент задела продукта еще не готов. Несмотря на всю ценность такой ответной реакции, следует сказать, что если уж владелец продукта объявил работу готовой, значит, она готова. Как обсуждалось в главе 9, владелец продукта должен быть наделен полномочиями главного руководителя разработкой

продукта. А для этого он должен иметь полное право одобрять или отклонять работу, и это право у него не может отнять ни один из участников подведения итогов спринта независимо от его ранга.

Это, конечно, не означает, что владелец продукта не должен прислушиваться к комментариям по поводу несоответствия обсуждаемого функционального средства ожиданиям участников проекта. Если произойдет нечто подобное, следует запланировать изменения в обсуждаемом функциональном средстве, создав новый элемент, отражающий желательное поведение, которое он должен проявлять по требованию участника проекта из руководства организации, а затем введя этот элемент в задел продукта для работы над ним в последующем спринте. Кроме того, владелец продукта должен выяснить причины его разногласий с остальными участниками проекта по поводу готовности обсуждаемого функционального средства и внести соответствующие коррективы, чтобы избежать в дальнейшем подобных недоразумений.

Нерегулярная посещаемость

Одна из наиболее характерных причин нерегулярной посещаемости подведения итогов спринта состоит в том, что участники проекта настолько заняты, что другие более важные обязанности не дают им возможности регулярно посещать данное мероприятие. Это явно указывает на организационные недостатки, когда участникам проекта приходится параллельно выполнять столько работы, что они не в состоянии справляться со всеми своими обязанностями. В подобных случаях организациям рекомендуется прекратить работу над низкоприоритетными продуктами до тех пор, пока они станут важными для участников проекта настолько, что те будут регулярно посещать подведение итогов спринта. Если же этот момент не настанет, то продукты, имеющие более низкий приоритет относительно других продуктов в портфеле заказов, так и останутся недостаточно ценными для разработки.

Иногда нерегулярная посещаемость происходит от того, что участники проекта просто не верят, что Scrum-команда способна произвести за несколько недель что-нибудь, заслуживающее внимания. Это особенно характерно для тех организаций, которые только начинают работать по методике Scrum. Участники проекта в таких организациях привыкли к намного более длительным периодам между просмотрами достигнутых результатов, а те совещания по подведению итогов, которые они посещали раньше, могли вызвать у них сплошное разочарование.

Самый лучший способ разрешить данный вопрос — получать в каждом спринте коммерчески ценный прирост потенциально готового к поставке продукта. И если команда это делает, то большинство участников проекта быстро осознают, насколько важно регулярно посещать подведение итогов спринта, чтобы дать свои отзывы, которыми Scrum-команда может оперативно воспользоваться.

Крупные проектные работы

Если крупные проектные работы выполняются многими Scrum-командами, то имеет смысл проводить совместное подведение итогов спринта. Такое мероприятие просто означает, что на нем обсуждается работа, завершенная несколькими тесно связанными вместе командами.

У такого подхода имеется ряд преимуществ. Во-первых, участники проекта должны посещать только одно подведение итогов спринта, а не несколько подобных мероприятий. Во-вторых, если работу предполагается выполнять совместными усилиями нескольких команд, то все внимание при подведении итогов спринта лучше сосредоточить на их совместной работе, а не на отдельных приростах продукта. С этой целью все команды должны непременно включить в свои критерии готовности комплексное тестирование, которое все равно придется проводить. А недостаток такого подхода заключается в том, что для совместного подведения итогов спринта, вероятнее всего, потребуется больше времени и места, чтобы на нем могли присутствовать все заинтересованные лица.

Заключение

В этой главе обсуждалось назначение подведения итогов спринта как важного мероприятия, замыкающего петлю обратной связи в процессе гибкой разработки по методике Scrum. В подведении итогов спринта принимают участие разные заинтересованные лица, цель которых — обследовать и адаптировать текущее состояние разработки продукции. Несмотря на то что подведение итогов спринта считается неформальным мероприятием, Scrum-команда должна хотя бы в минимальной степени подготовиться к нему, чтобы оно прошло в здоровой и продуктивной атмосфере и принесло известную пользу. В начале подведения итогов спринта Scrum-команда делает краткий обзор того, что было сделано в течение спринта. Затем она демонстрирует прирост продукта, полученный в течение спринта. Далее следует оживленное обсуждение среди присутствующих, которым предлагается задавать вопросы, высказывать свои наблюдения и предположения по поводу увиденного. На основании этого обсуждения упорядочивается задел продукта и обновляется план выпуска. А следующая глава посвящена ретроспективе спринта — мероприятию по обследованию и адаптации процесса разработки.

Глава 22
Ретроспектива спринта

В конце каждого спринта, выполняемого по методике Scrum, появляются две возможности: провести подведение итогов и ретроспективу спринта. В предыдущей главе обсуждались особенности подведения итогов спринта, где команда и остальные участники проекта рассматривают сам продукт. А в этой главе речь пойдет о ретроспективе спринта, где Scrum-команда обследует процесс разработки данного продукта.

Сначала в главе дается краткий обзор назначения ретроспективы спринта и состава ее участников. Затем описываются подготовительная работа и основные виды деятельности, связанные с ретроспективой спринта, самая важная из которых происходит после ретроспективы спринта, когда ее участники фактически внедряют в процесс разработки те усовершенствования, которые они наметили в ходе данного мероприятия.

Краткий обзор

В предисловии к своей книге *Project Retrospectives* (в русском переводе книга вышла под названием *Ретроспектива проекта* в Издательстве Дмитрия Лазарева, 2015 г.) Норм Керт [Kerth, Norm. 2001] — основатель современного движения ретроспектив, подытоживает их назначение следующей цитатой из детской книжки *Винни-Пух*:

> *А вот и медвежонок Эдвард, с шумом (бум-бум-бум) спускающийся по ступенькам лестницы на своем затылке вслед за Кристофером Робином. Насколько ему известно, это единственный способ спускаться по лестнице, но иногда ему кажется, что есть и другой способ, если бы он мог перестать на минутку ударяться головой о ступеньки лестницы и немного поразмыслить над этим.*

Ретроспективы спринтов дают всей Scrum-команде возможность перестать ударяться головой о ступеньки производственной лестницы и немного поразмыслить на тем, что она делает (рис. 22.1). Во временны*х* рамках ретроспективы спринта команды могут свободно изучать сложившуюся ситуацию, анализировать свою работу, выявлять способы ее улучшения и составлять планы реализации

этих улучшений. Все, что оказывает влияние на порядок создания продукта командой, в том числе процессы, нормы практики, общение, среда, артефакты, инструментальные средства и прочее, подлежит свободному обсуждению.

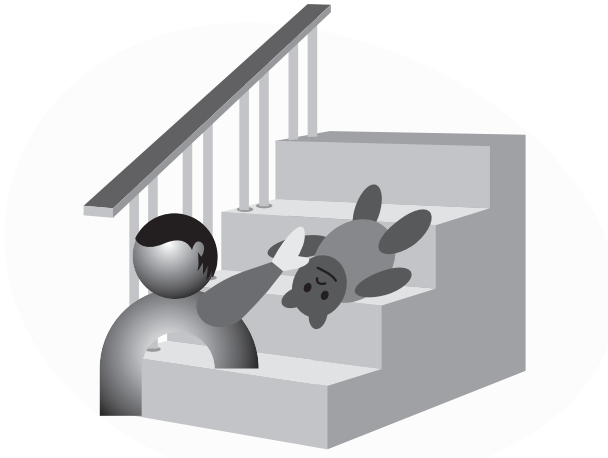


Рис. 22.1. Медвежонок Эдвард, наглядно показывающий потребность в ретроспективе

Ретроспектива спринта является одной из самых важных и недооцененных норм практики в инфраструктуре Scrum. Особое значение ретроспективы спринта состоит в том, что она дает командам возможность приспосабливать методику Scrum к конкретным обстоятельствам. А недооцененной ретроспектива спринта остается потому, что многие неверно считают, что она только отнимает время, отвлекая от выполнения "настоящего" проектирования, построения и тестирования.

Ретроспектива спринта вносит решающий вклад в непрерывное усовершенствование процесса разработки, которое предлагает методика Scrum. И хотя некоторые организации могут откладывать проведение ретроспективы до завершения крупных проектных работ, Scrum-команды должны проводить это мероприятие в конце каждого спринта (рис. 22.2). Это дает им возможность критически взглянуть на свои достижения и воспользоваться текущими сведениями о процессе разработки, прежде чем такая возможность будет утрачена. А поскольку Scrum-команда все равно собирается вместе в конце каждого спринта для обследования и адаптации Scrum-процесса, то она может применить прежние результаты постепенного обучения в ходе данного процесса и тем самым оказать существенное влияние на исход проекта.

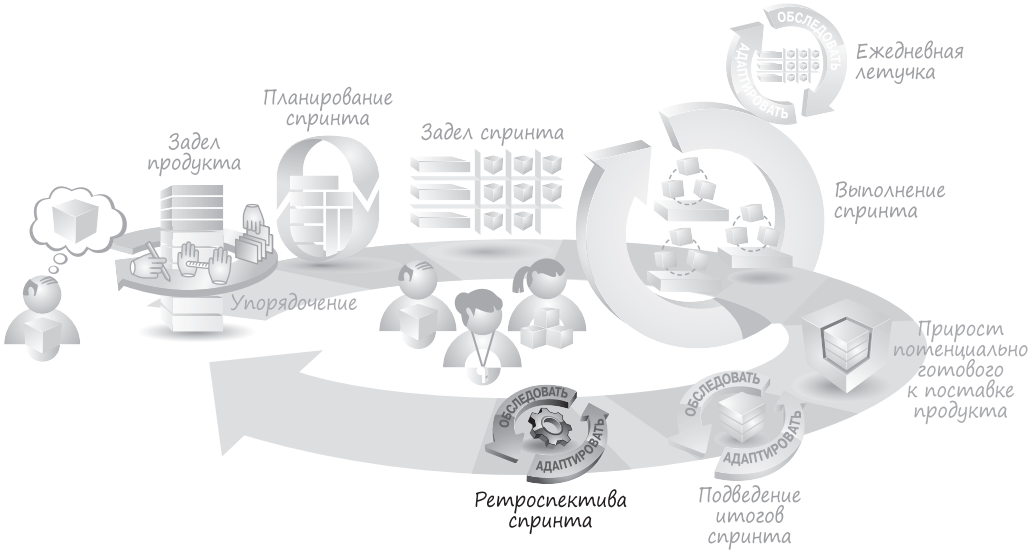


Рис. 22.2. Когда проводится ретроспектива спринта

В остальной части этой главы подробно описывается метод проведения ретроспективы спринта. Но эти подробности не должны наводить на мысль, что ретроспектива спринта — сложный и церемониальный процесс. На самом деле для проведения ретроспективы спринта члены Scrum-команды должны просто собраться вместе и ответить на приведенные ниже вопросы. Отвечая на эти вопросы в ходе обсуждения, члены Scrum-команды сначала намечают ряд изменений, которые следует внести в процесс разработки, а затем переходят к следующему спринту, постепенно совершенствуя данный процесс.

- Что удалось в данном спринте и желательно делать дальше?
- Что не удалось в данном спринте и не следует делать дальше?
- Что следует начать делать или совершенствовать?

Участники

Ретроспектива спринта — это время для размышления над процессом разработки, и поэтому в данном мероприятии должна принимать участие вся Scrum-команда, т.е. все члены команды разработчиков, Scrum-мастер и владелец продукта. От команды разработчиков в ретроспективе спринта принимают участие все, кто занимается проектированием, построением и тестированием разрабатываемого продукта. Совместно эти члены команды составляют

полное и разностороннее представление о процессе разработки, что очень важно для выявления способов его совершенствования с разных точек зрения.

Scrum-мастер принимает участие в ретроспективе спринта, потому что, во-первых, он является непосредственным участником процесса разработки, а во-вторых, надзирает за этим процессом от имени всей Scrum-команды (см. главу 10). Но это совсем не означает, что Scrum-мастер должен указывать команде, как изменить процесс разработки. Напротив, это означает, что он может подсказать команде, где она не придерживается согласованного процесса, а также поделиться с ней своими знаниями, опытом и идеями.

Некоторые считают, что присутствие владельца продукта на ретроспективе спринта может помешать команде с полной откровенностью выявить трудности в ее работе. И хотя в некоторых организациях такой риск возможен, тем не менее владелец продукта является важным участником Scrum-процесса, а следовательно, он должен участвовать в обсуждении процесса разработки. Если же отсутствует взаимное доверие владельца продукта и команды разработчиков или же высказываться откровенно не совсем удобно и даже небезопасно, то владельцу продукта, вероятно, не стоит посещать ретроспективу спринта до тех пор, пока Scrum-мастер не поможет создать в коллективе атмосферу большего доверия, безопасности и откровенности.

Если же такая атмосфера доверия, безопасности и откровенности создана, то участие владельца продукта в ретроспективе спринта имеет решающее значение для достижения быстрого и гибкого потока доставки коммерческой ценности. Например, владелец продукта является своего рода проводником потока требований от заказчика к исполнителю, т.е. команде разработчиков. А что, если что-нибудь нарушится в этом потоке требований по ходу Scrum-процесса? Возможно, элементы задела продукта оказались недостаточно упорядоченными к началу планирования спринта. В подобных случаях Scrum-команде будет трудно выявить пути совершенствования процесса разработки, если владелец продукта будет отсутствовать на ретроспективе спринта.

Остальные участники проекта или руководители, не входящие в состав Scrum-команды, должны принимать участие в ретроспективе спринта только по приглашению от самой Scrum-команды. И хотя прозрачность является базовой ценностью методики Scrum, на практике во многих организациях еще не достигнут такой уровень безопасности в общении, чтобы остальные участники проекта регулярно посещали ретроспективы спринтов. Члены Scrum-команды должны чувствовать себя в безопасности, чтобы откровенно и открыто обсуждать текущее состояние проекта, не испытывая никаких препятствий от посторонних участников данного мероприятия. Если же они не чувствуют себя в достаточной безопасности в присутствии посторонних участников, то ретроспектива спринта теряет свою эффективность.

Подготовительная работа

Перед ретроспективой спринта проводится определенная подготовительная работа (рис. 22.3). Для краткосрочных спринтов или команд, пользующихся опробованной на практике простой формой ретроспективы спринта, такая подготовительная работа потребует немного времени или вообще не потребует его.

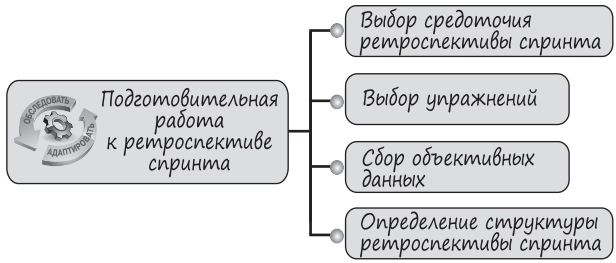


Рис. 22.3. Подготовительная работа к ретроспективе спринта

Выбор средоточия ретроспективы спринта

У каждой ретроспективы спринта должно быть вполне определенное средоточие. По умолчанию ретроспектива спринта сосредоточена на критическом анализе всех особенностей процесса разработки, которыми Scrum-команда пользовалась в течение текущего спринта. Но иногда команда может выбрать другое средоточие ретроспективы спринта, исходя из своих насущных потребностей и возможностей выявить пути усовершенствования процесса. Например, команда может сосредоточиться на следующем.

- Совершенствование навыков в разработке посредством тестирования (TDD).
- Выявление причин, по которым после демонстрации продукта, разработанного по требованиям заказчика, последний нередко заявляет, что его требования были истолкованы неверно или в продукте отсутствует важная сторона требований заказчика.

Выбор и сообщение средоточия ретроспективы спринта до ее начала дает Scrum-команде возможность решить, следует ли приглашать на данное мероприятие остальных участников проекта, не входящих в состав Scrum-команды. Кроме того, зная заранее средоточие ретроспективы спринта, команда может выбрать походящие для нее упражнения и дать своим членам время для сбора и подготовки любых данных, требующихся для обеспечения нормального проведения ретроспективы спринта.

Наличие возможности выбирать конкретное средоточие ретроспективы спринта позволяет долговременным и высокопроизводительным

Scrum-командам продолжать и дальше извлекать ощутимую ценность из ретроспективы спринта. Например, в одной организации, которую я наставлял, была зрелая Scrum-команда, члены которой дружно работали вместе почти три года. Они совместно выполнили много десятков спринтов, но начинали ощущать, что проведение ретроспективы спринта, сосредоточенной на только что завершенном спринте, нередко приносит мало пользы. Один из членов этой команды как-то заметил: "Долгое время ретроспективы спринта считались незаменимыми, а теперь они нередко похожи на процесс ради процесса". В конечном счете команда стала проводить ретроспективы спринта более короткими и сосредоточенными, что позволило ей и приглашаемым посторонним, но заинтересованным лицам вникать в весьма специфические вопросы, докапываясь до первопричин затруднений в процессе разработки. Таким образом, команда продолжала учиться и совершенствоваться, несмотря на свой значительный опыт применения методики Scrum. Этот пример показывает, что для развития всегда есть место. И чтобы выявить возможности для развития, достаточно проводить более сосредоточенные ретроспективы.

Выбор упражнений

Как только будет выбрано сосредоточие и окончательный состав участников предстоящей ретроспективы спринта, можно перейти к выбору тех упражнений, которые могут помочь участникам коллективно анализировать, исследовать и принимать решения. Ниже перечислены упражнения, типичные для ретроспективы спринта.

- Создание и глубинный анализ временной шкалы событий в спринте.
- Мозговой штурм наблюдений.
- Группирование и голосование по наблюдениям.

Но эти упражнения можно варьировать, поддерживая средоточие ретроспективы или учитывая состав ее участников. Можно также опробовать другие упражнения, чтобы не стоять на месте. Другие разновидности упражнений приведены в [Kerth, Norm. 2001] и [Derby, Esther, and Diana Larsen. 2006].

Участникам ретроспективы спринта совсем не обязательно выбирать конкретные упражнения при выполнении подготовительной работы. На самом деле некоторые упражнения лучше выбирать оперативно при проведении ретроспективы спринта, исходя из того, что больше подходит самим ее участникам. В то же время некоторые упражнения, особенно те, что требуют исходных данных или вспомогательных материалов, лучше выбрать при выполнении подготовительной работы. Таким образом, готовиться к ретроспективе спринта нужно, но в то же время проявлять гибкость в подготовке к ней.

Сбор объективных данных

Благодаря тому что ретроспектива спринта выполняется сосредоточенно и в короткий период времени, определяемый самой командой, при подготовке к ретроспективе спринта придется пойти на любые хлопоты, связанные со сбором нужных данных. Выбрав средоточие и упражнения для предстоящей ретроспективы спринта, нужно ясно понять, какие именно объективные данные следует собирать, если это вообще потребуется. Объективные данные являются точными и достоверными, а не просто мнениями. Такие данные могут, например, обозначать, какие события и когда происходили в течение спринта; количество начатых, но не завершенных элементов из задела продукта; а также объем выполнявшихся работ для построения диаграммы сгорания спринта. На стадии подготовки к ретроспективе спринта эти данные не организуются и не анализируются, а только собираются, чтобы быть доступными при проведении ретроспективы спринта.

Структура ретроспективы спринта

Аналогично подведению итогов спринта, ретроспектива проводится в конце каждого спринта, зачастую после подведения итогов спринта и, как правило, в одном и том же месте, в один и тот же день и в то же самое время в каждом спринте. Но в отличие от подведения итогов спринта, иногда может возникнуть потребность изменить место, дату или время проведения конкретной ретроспективы спринта, чтобы лучше соблюсти ее средоточие, пригласить посторонних участников или выполнить конкретные запланированные упражнения. Именно поэтому я предпочитаю пересматривать структуру ретроспективы спринта при подготовке к ней.

На конкретную продолжительность ретроспективы спринта оказывают влияние самые разные факторы, в том числе состав команды, насколько команда нова, работают ли члены команды в удаленном режиме и т.д. Как показывает мой опыт, команды, только начинающие работать по методике Scrum, стремятся выделять слишком мало времени на проведение ретроспективы спринта. Но ведь провести полезную ретроспективу спринта меньше, чем за 60 минут, довольно трудно. Как правило, для данного мероприятия следует зарезервировать около 1,5 часа, если речь идет о двухнедельных спринтах, и пропорционально больше времени при более продолжительных спринтах.

Scrum-команда должна выбрать для ретроспективы спринта место, наиболее подходящее для успешного проведения этого мероприятия. Одни команды предпочитают проводить свои ретроспективы в привычной для них рабочей обстановке, где вывешены крупные, хорошо видные диаграммы. Это дает им возможность легко получать доступ к большому массиву важной информации. А другие команды предпочитают собираться подальше от привычной рабочей обстановки, возможно, для того, чтобы им ничто не мешало открыто высказывать свое

мнение. Но в целом место проведения ретроспективы спринта не имеет особого значения, помимо того, что оно должно обеспечивать безопасную атмосферу, где члены команды могут свободно высказывать свои суждения.

Несмотря на то что Scrum-мастер нередко действует в качестве координатора ретроспективы спринта и может делать это довольно эффективно, иногда оказывается удобнее привлечь опытного и нейтрального координатора со стороны, чтобы помогать членам команды начинать ретроспективы или проводить особенно сложные или деликатные ретроспективы, где менее уместен координатор, тесно связанный с командой. С другой стороны, в организациях, где действует несколько Scrum-команд с разными Scrum-мастерами, нередко оказывается полезно и даже поучительно назначать Scrum-мастера из одной команды в качестве координатора ретроспективы спринта другой команды. И это можно сделать при подготовке к ретроспективе спринта.

Метод

Процесс проведения ретроспективы спринта наглядно показан на рис. 22.4.

Предпосылками к проведению ретроспективы спринта являются согласованное средоточение, любые упражнения и вспомогательные материалы, которыми команда может решить воспользоваться в течение ретроспективы. Кроме того, для проведения большинства ретроспектив требуются хотя бы некоторые предварительно собранные объективные данные. Еще одной предпосылкой к проведению ретроспективы спринта является задел наблюдений, выработанных в течение предыдущих ретроспектив.

К результатам проведения ретроспективы спринта относится ряд конкретных действий для совершенствования процесса разработки, выполнить которые команда согласованно решила в следующем спринте. Кроме того, к результатам проведения ретроспективы спринта может относиться задел наблюдений, которые были накоплены в течение текущей ретроспективы спринта и к которым команда собирается обратиться не в предстоящем спринте, а впоследствии. Члены команды могут также надеяться на улучшение товарищеских отношений в качестве одного из результатов проведения ретроспективы спринта.

Несмотря на то что существуют разные методы проведения ретроспективы спринта, как правило, присутствующие на данном мероприятии отвечают на следующие вопросы:

- Что удалось в данном спринте и желательно делать дальше?
- Что не удалось в данном спринте и не следует делать дальше?
- Что следует начать делать или совершенствовать?

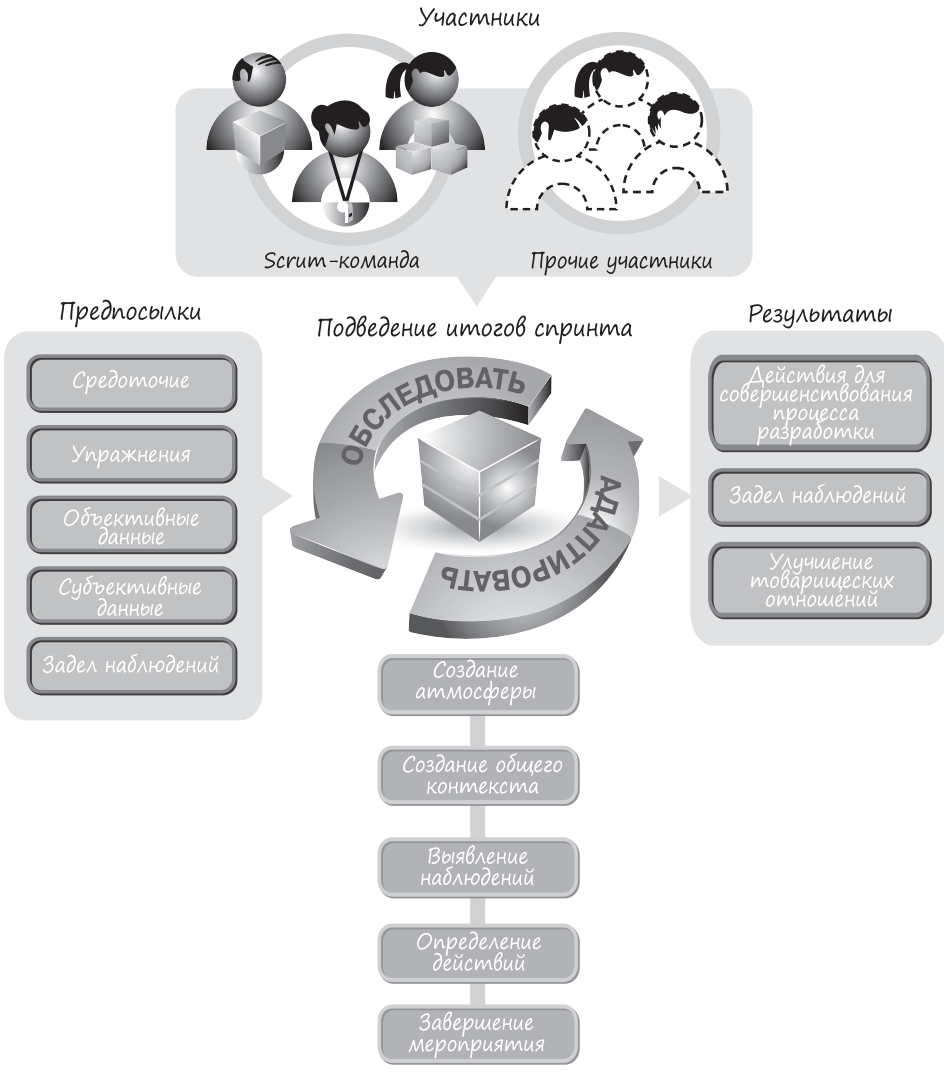


Рис. 22.4. Процесс проведения ретроспективы спринта

Удобный, на мой взгляд, метод, аналогичный описанному в [Derby, Esther, and Diana Larsen. 2006], состоит в том, чтобы создать подходящую атмосферу для ретроспективы спринта и общий контекст для всех участников данного мероприятия, выявить наблюдения, способные привести к улучшениям, определить конкретные действия на следующий спринт и на этом завершить ретроспективу спринта.

Создание атмосферы

В течение ретроспективы спринта присутствующим предлагается проанализировать поведение и производительность команды и дать конкретные рекомендации по совершенствованию ее деятельности. Конечно, рассматривать свою деятельность и команды в целом как под микроскопом не очень удобно. Поэтому начать ретроспективу спринта лучше всего с создания атмосферы, в которой присутствующие почувствовали бы себя уютно.

В такой атмосфере присутствующие должны чувствовать себя в безопасности и свободно высказывать свои мнения, не боясь понести наказание. Команды должны руководствоваться установившимися основными правилами или договором о сотрудничестве, благодаря которому можно свободно высказывать свое мнение, не боясь выносить сор из избы. В основных правилах должно быть ясно сформулировано, что главное внимание в ретроспективе спринта делается на организационных моментах производственного процесса, а не на отдельных его участниках. Благодаря этому они могут без опаски исследовать, что было сделано не так в обсуждаемом спринте.

И хотя трудности, возникающие в работе, иногда кроются в самих людях, ретроспектива спринта — не место для их разрешения. Ведь назначение ретроспективы спринта — усовершенствовать производственный процесс Scrum-команды, а не обвинять или выговаривать кому-то за его поведение. Создание подходящей атмосферы обеспечивает соблюдение основных правил, укрепляющих принцип работы в среде без взаимных обвинений.

Не менее важно создать прецедент активного участия. Проведение ретроспективы вряд ли окажется эффективным, если допустить пассивную роль его участников. Таким образом, очень важно создать подходящую атмосферу, чтобы поощрять присутствующих к активному участию в ретроспективе спринта. В некоторых командах просто предлагается каждому участнику ретроспективы спринта выразить в нескольких словах свое мнение о текущем положении дел. И не так важно, на какие именно вопросы им придется отвечать, но в целом им должно быть предложено высказаться, чтобы принять участие в обсуждении.

Создание общего контекста

Все члены группы людей одинаково переживают одно и то же событие, хотя интерпретируют его по-разному. Поэтому, чтобы успешно обследовать текущий спринт, очень важно ввести всех участников ретроспективы спринта в один общий контекст. А для того чтобы установить общий контекст, участники должны согласовать отличия в своих взглядах, как показано на рис. 22.5, *слева*, выработав общую точку зрения команды на текущее положение дел, как показано на рис. 22.5, *справа*.

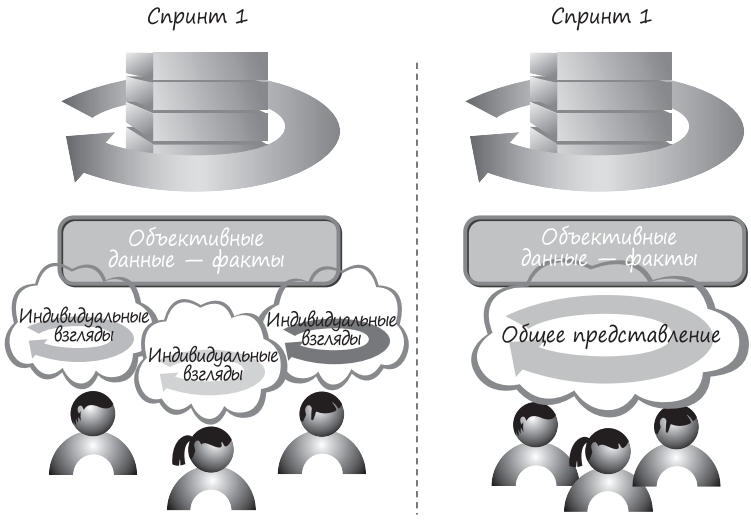


Рис. 22.5. Согласование отличий во взглядах для создания общего контекста

Как показано на рис. 22.5, *слева*, каждый участник ретроспективы спринта может иметь свое представление о текущем спринте, отличающееся, исходя из его собственного опыта, приобретенного в течение обсуждаемого спринта, вместо общего представления о событиях в спринте, достижениях и недостатках. Если допустить преобладание отдельных взглядов, то ретроспектива спринта может превратиться в полемику разных мнений вместо сосредоточения на практических результатах, опираясь на общий контекст.

Следовательно, устанавливая общий контекст, следует в первую очередь позаботиться о создании общего объективного представления о текущем спринте. После того как все присутствующие будут вовлечены в обсуждение, следует обменяться объективными данными, в том числе сведениями о количестве элементов из задела спринта, которые команда обязалась завершить и фактически завершила, количестве выявленных дефектов и т.д. (Конкретный характер подходящих объективных данных зависит от средоточия ретроспективы спринта.) Несмотря на то что бо́льшая часть объективных данных, как правило, собирается при подготовке к ретроспективе спринта, некоторые из этих данных могут быть совместно собраны участниками в течение ретроспективы спринта. Это помогает нацелить команду на анализ объективных данных. Каким бы образом объективные данные ни были собраны (при подготовке или проведении ретроспективы спринта), этот процесс имеет решающее значение для установления общего контекста на основании объективных фактов, а не субъективных мнений.

Но опора на объективные данные совсем не означает, что субъективные мнения вообще не имеют никакого значения. Каждый участник привносит

в ретроспективу спринта субъективные данные, отражающие его личное представление о спринте. Если такие данные не выявляются и не обсуждаются, то участники ретроспективы могут просто допустить, что все остальные пережили обсуждаемый спринт таким же точно образом. Подобное несоответствие может затруднить понимание одними участниками комментариев и предложений других участников ретроспективы спринта.

Участники ретроспективы спринта могут выполнить целый ряд упражнений, чтобы выработать общий контекст на основании объективных и субъективных данных. К числу наиболее распространенных упражнений относятся построение и анализ временной шкалы событий в спринте, а также сейсмограммы эмоций.

Временная шкала событий

Построение и анализ *временной шкалы* — простой, но эффективный способ формирования общего артефакта, наглядно показывающего ход событий в течение спринта. К таким событиям можно отнести срыв процесса построения, перерыв на устранение сбоя в производстве или приход работника из отпуска.

Как правило, временная шкала событий рисуется на стене или белой доске, где участники размещают карточки (или наклейки с комментариями), обозначающими важные события, произошедшие в течение спринта (рис. 22.6). Распределенные команды могут делать то же самое, совместно используя оперативно доступный вариант белой доски.

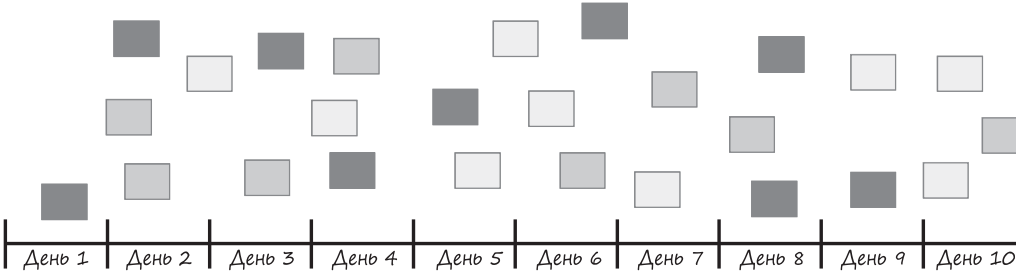


Рис. 22.6. Временная шкала событий

Карточки отдельных событий размещаются на временной шкале в хронологическом порядке. Такое представление событий в хронологическом порядке наглядно показывает ход работ в течение спринта и обеспечивает удобный контекст для быстрого выявления недостающих или забытых событий.

Чтобы нагляднее представить события, разделив их по отдельным категориям, многие команды пользуются разноцветными карточками. Так, одни команды обозначают цветом разные типы событий (например, зеленым — техническое событие, желтым — организационное, красным — личное). А другие пользуются разными цветами для обозначения ощущений или энергетических уровней

(например, зеленым — положительное событие, желтым — нейтральное, розовым или красным — отрицательное).

Сейсмограмма эмоций

Многие команды строят *сейсмограмму эмоций* в дополнение к временной шкале событий. Это графическое представление эмоциональных подъемов и падений, которые участники проявляли в течение спринта (рис. 22.7). Построение сейсмограммы эмоций помогает расширить общий контекст за пределы объективных данных о событиях, дополняя их субъективными данными о том, как восприняла их команда.

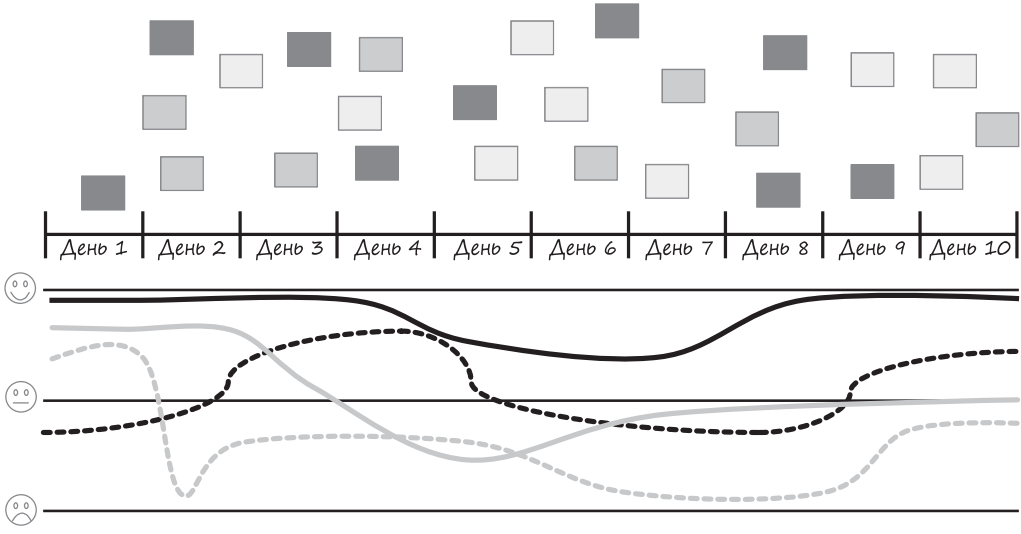


Рис. 22.7. Сейсмограмма эмоций

Для построения сейсмограммы эмоций участникам предлагают нарисовать кривые, отображающие их ощущения или энергетические уровни по ходу спринта. Нередко сейсмограмму эмоций удобно нарисовать непосредственно под временной шкалой событий, чтобы эти наглядные представления данных о спринте можно было сравнивать визуально. В дальнейшем участники ретроспективы спринта могут провести глубинный анализ этих данных, чтобы выявить интересные наблюдения по поводу усовершенствования процесса разработки.

Выявление наблюдений

Как только контекст будет установлен, участники ретроспективы спринта могут внимательно изучить, уяснить и интерпретировать данные, чтобы выявить наблюдения по поводу усовершенствования процесса разработки. Для того

чтобы сделать это эффективно, требуется сосредоточить основное внимание на более общей картине системного уровня. Если же команда сосредоточит внимание лишь на одном аспекте, имея более локализованное представление, она может упустить из виду более общую картину. А сосредоточение основного внимания на системном уровне помогает также команде докопаться до первопричин, не обращая внимания на все несущественное.

Участники ретроспективы спринта должны приступить к глубокому анализу данных, полученных из общего контекста. Например, они могли бы проанализировать временную шкалу и сейсмограмму эмоций и найти ответы на следующие вопросы, чтобы выявить интересные наблюдения:

- Что удалось вполне?
- Что не совсем удалось?
- Какие имеются возможности, чтобы поступать иначе?

Зачастую участникам ретроспективы спринта предлагается сначала провести мозговой штурм наблюдений, а затем зафиксировать их на карточках, которые размещаются на общей стене или другой поверхности, чтобы все заинтересованные лица могли их видеть (рис. 22.8).

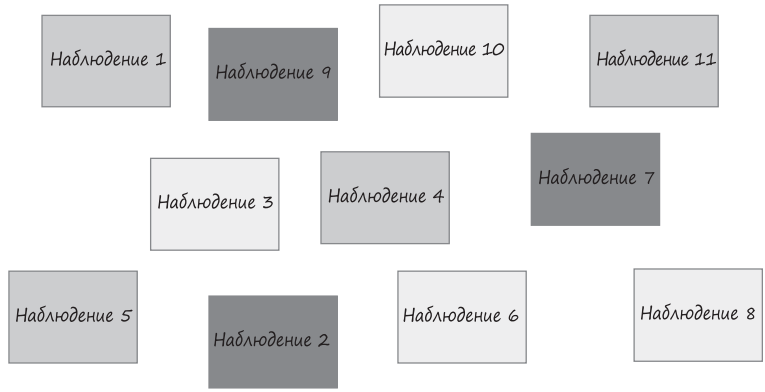


Рис. 22.8. Стена с карточками наблюдений, вывешиваемыми на стене

Еще одним источником наблюдений может стать *задел наблюдений* — приоритизированный список сформированных ранее наблюдений, которые еще не были приняты к действию. Любые существующие наблюдения должны быть представлены в виде карточек и размещены на стене наряду с новыми наблюдениями.

Как только карточки будут размещены на стене, участникам ретроспективы спринта придется их организовать. С этой целью многие команды решают провести упражнение в виде *негласного группирования*, чтобы разделить наблюдения по целенаправленным группам, обозначающим сходные или дублирующиеся карточки (рис. 22.9).

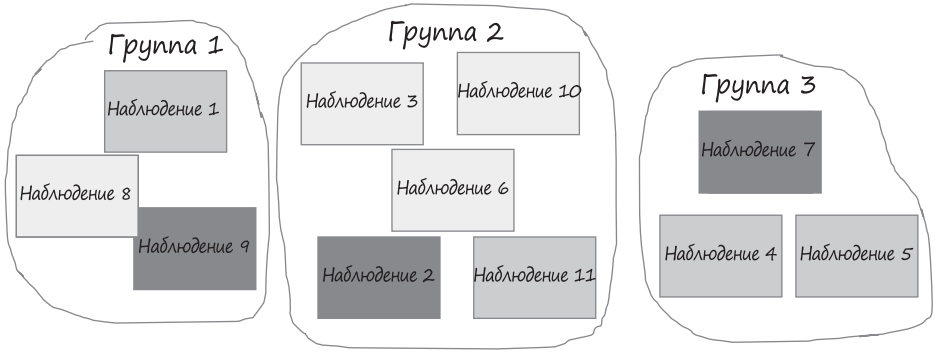


Рис. 22.9. Карточки наблюдений, разделенные на сходные группы

В течение негласного группирования участники ретроспективы спринта совместно формируют группы без устного обсуждения, опираясь только на расположение и перемещение отдельных карточек как средств сообщения и координирования действий среди участников. Таким образом, негласное группирование — весьма эффективное по времени упражнение.

Некоторые команды предпочитают разделять стену на участки по отдельным категориям (например, по сохраняемым, прекращаемым и опробуемым нормам практики), прежде чем начинать ретроспективу спринта. Затем по мере создания карточек наблюдений участники размещают каждую карточку на участке стены соответствующей категории (рис. 22.10). Но даже если категории назначены заранее, участники могут эффективно выполнить негласное группирование сходных карточек в пределах одной категории.

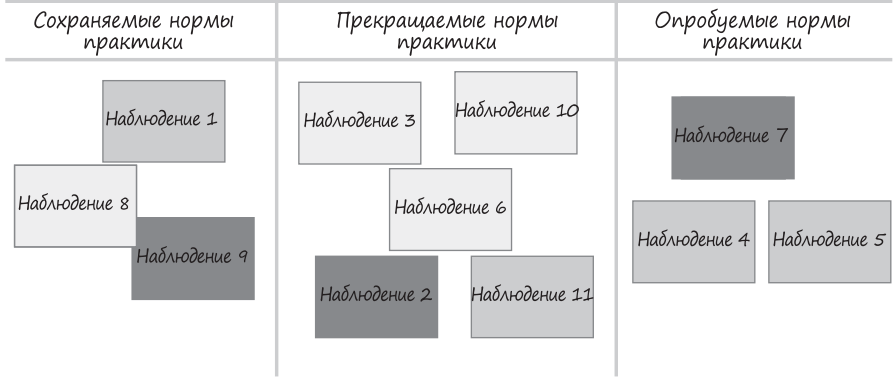


Рис. 22.10. Карточки наблюдений, распределенные по отдельным категориям

После того, как будет создан общий контекст и проведен глубокий анализ данных для выявления наблюдений, участники ретроспективы спринта должны получить в свое распоряжение немало участков для совершенствования их

практики применения методики Scrum, а также совместной работы для доставки ценности. Некоторые из выявленных наблюдений могут способствовать более основательным обсуждениям среди участников, чтобы лучше понять первопричины, важные образцы или взаимоотношения. Как только все наблюдения будут обсуждены, а их карточки размещены на стене, можно решить, что делать дальше со всей этой информацией.

Определение действий

Наблюдения — это идеи или представления о нормах практики, которые требуется усовершенствовать. Чтобы извлечь пользу из этих наблюдений в долгосрочной перспективе, следует перейти от их обсуждения к принятию демонстрируемых действий с целью их эффективного применения. Так, если сложилось наблюдение, что команда тратит слишком много времени из-за постоянных сбоев в системе контроля кода, то в качестве улучшающего действия можно вставить в систему контроля кода заплаты, предоставляемые поставщиком, чтобы сделать ее работу более устойчивой. Выполнение этого действия может быть поручено отдельному члену команды в следующем спринте.

Участники должны также рассмотреть результаты улучшающих действий, предпринятых после предыдущей ретроспективы спринта. Если эти действия не были завершены или даже начаты, участники должны выяснить соответствующие причины, прежде чем обращаться к новым наблюдениям. Они могут решить перенести дальше предыдущие действия или сделать их более приоритетными по сравнению с только что определенными наблюдениями.

Выбор наблюдений

Очень важно понять, что в течение ретроспективы спринта нередко удается выявить намного больше улучшающих наблюдений, чем Scrum-команда или организация способна реализовать или воплотить в конкретные действия за короткий период времени. Поэтому участники ретроспективы спринта должны выбрать те улучшающие наблюдения, которые они готовы воплотить в конкретные действия немедленно, отложив до времени остальные наблюдения. С этой целью участники должны расставить наблюдения по приоритетам, исходя из своих представлений о том, что, на их взгляд, важнее и лучше всего усовершенствовать. Вполне возможно, что конкретное улучшающее наблюдение может оказаться очень важным, но если нет особого желания выполнять работу для его реализации, то, возможно, и не стоит делать это сразу. Если участники готовы реализовать наблюдение, они, скорее всего, предпримут для этого конкретные действия. Расставить наблюдения по приоритетам можно, в частности, методом *точечного голосования*, как показано на рис. 22.11.

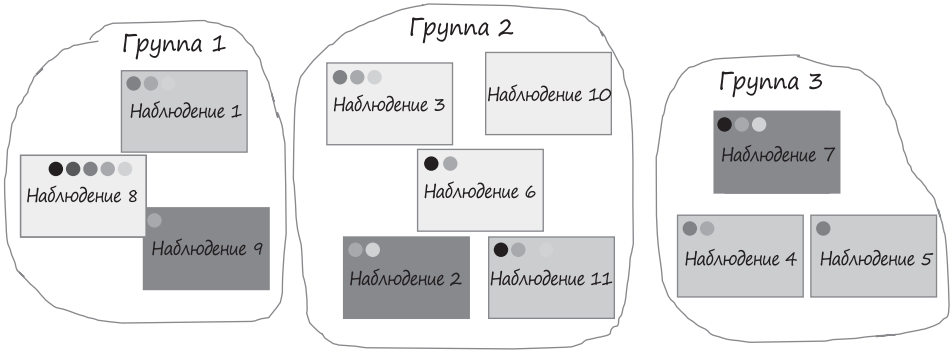


Рис. 22.11. Пример точечного голосования

При точечном голосовании каждому участнику ретроспективы спринта сначала дается небольшое количество (от трех до пяти) цветных точечных меток. Затем все участники одновременно накладывают свои точечные метки на карточки улучающих наблюдений, чтобы расставить их по приоритетам реализации. Отдельный участник может наложить все точечные метки на одну карточку или распределить их по нескольким карточкам. Как только все участники проголосуют, карточки, набравшие наибольшее количество голосов, должны быть рассмотрены в первую очередь. Количество наблюдений, отобранных участниками для реализации, зависит от их возможностей и времени, которое они готовы уделить наблюдениям.

Как правило, время для реализации наблюдений выделяется в следующем спринте. Так, если Scrum-команда выполняет двухнедельные спринты, она, скорее всего, рассмотрит наблюдения, которые она готова реализовать в следующий двухнедельный период времени. И даже если наблюдение окажется слишком крупным, чтобы полностью реализовать его в следующем спринте, участники ретроспективы спринта могут взяться за работу над ним, чтобы продемонстрировать определенный прогресс.

Кроме того, участники должны определить свои возможности, чтобы посвятить себя реализации наблюдений в течение следующего спринта (или другого периода времени, который они посчитают удобным). Если же команда собирается уделить время в следующем спринте действиям, оставшимся из предыдущей ретроспективы спринта, такое решение, очевидно, повлияет на возможности команды выполнить новые действия, намеченные в текущей ретроспективе.

Необходимость уделять время прежним и новым наблюдениям оставляет меньше времени для работы непосредственно над функциональными средствами. Так сколько же времени должна выделить команда для тех наблюдений, реализация которых окупится в отдаленной перспективе? Для ответа на этот вопрос потребуется участие владельца продукта. Именно поэтому так важно его

присутствие на ретроспективе спринта. Ведь если Scrum-команда не выделит достаточно времени для работы над наблюдениями, то такая работа, скорее всего, не начнется вообще.

Зная возможности, имеющиеся для работы над наблюдениями, участники ретроспективы спринта могут составить хотя бы приблизительное представление о тех высокоприоритетных наблюдениях, которые не терпят отлагательства. Но окончательное решение принимается лишь после того, как будут определены конкретные действия.

Выбор действий

Итак, выявленные наблюдения расставлены по приоритетом и определены возможности для работы над ними. Но никакой ощутимой ценности не будет извлечено из ретроспективы спринта до тех пор, пока не будут определены конкретные оперативные шаги, чтобы эффективно воспользоваться наблюдениями и усовершенствовать Scrum-процесс.

Большинство действий примет форму конкретных задач, которые члены одной или нескольких Scrum-команд будут выполнять в течение предстоящего спринта. Так, если наблюдение заключается в том, что на выявление нарушений в коде уходит слишком много времени, соответствующее действие может состоят в следующем: сервер сборки приложений должен посылать (в том числе и по электронной почте) сообщение о нарушении сборки. Такое действие требует привлечения к работе на уровне отдельных задач одного или нескольких членов команды разработчиков. С этой целью команда должна определить, кто способен выполнить такую работу и сколько времени для этого потребуется. И лишь тогда команда может быть уверена, что работа над конкретным наблюдением вполне ей по силам.

Не все наблюдения требуют работы на уровне отдельных задач. Например, наблюдение вроде взаимного уважения, которое члены команды разработчиков должны проявлять, вовремя приходя на ежедневные летучки, должно потребовать от команды очень мало (если вообще потребовать) какой-то работы на уровне отдельных задач. И хотя намерение вовремя приходить на ежедневные летучи можно считать настоящим действием, оно никак не сокращает возможности команды.

Иногда действия представляют определенные препятствия, которые способен устранить не Scrum-мастер, а кто-нибудь другой в организации. Например, наблюдение может состоять в том, что завершить элементы задела спринта нельзя, потому что для их тестирования требуется последняя версия стороннего программного обеспечения. В таком случае действие может состоять в том, чтобы поручить Scrum-мастеру обратиться в отдел снабжения с заявкой

на приобретение стороннего программного обеспечения, отсутствие которого мешает команде завершить работу над элементами задела продукта. Это действие начнется со следующего спринта и потребует от Scrum-мастера некоторых возможностей, а на его выполнение может уйти несколько спринтов.

Определяя подходящие действия, не следует забывать, что реализовать наблюдение сразу, возможно, и не удастся. Вместо этого может потребоваться исследование наблюдения, прежде чем сделать усовершенствование. В подобных случаях правильное действие может состоять в том, чтобы провести обследование и собрать данные в течение следующего спринта, чтобы лучше понять возникшее затруднение.

Например, наблюдение может заключаться в том, что неясны причины, по которым собственные тесты двух компонентов, которые полностью проверены, не проходят, когда они объединяются в межкомпонентный автоматизированный тестовый набор, где каждый компонент по-прежнему выполняется отдельно. В таком случае отсутствует конкретное действие, которое члены Scrum-команды могут предпринять, чтобы реализовать данное наблюдение, поскольку они просто не понимают, что на самом деле происходит. Тем не менее команда может сформировать действие для исследования этого вопроса отдельными членами команды в следующем спринте. Кроме того, команда может определить те возможности, которые потребуется выделить для подобного исследования.

Задел наблюдений

Как упоминалось ранее, многие команды создают *задел наблюдений*, иногда еще называемый *заделом усовершенствований*, чтобы хранить любые недостатки, обнаруженные в течение ретроспективы спринта, но не требующие немедленного устранения. Назначение такого задела состоит в том, чтобы на следующей ретроспективе спринта участники смогли выбрать из задела те наблюдения, которые требуется реализовать прежде новых наблюдений, когда определяется, чему именно в первую очередь следует уделить время в следующем спринте. Разумеется, задел наблюдений должен периодически упорядочиваться, чтобы в нем оставались лишь ценные наблюдения.

В некоторых командах просто отвергают любые наблюдения, над которыми решено не работать в следующем спринте. Они считают, что если наблюдение действительно заслуживает внимания, оно все равно будет выявлено снова на следующей ретроспективе спринта.

Завершение мероприятия

Как только окончательные действия будут определены, участники завершают ретроспективу спринта. Многие команды завершают это мероприятие, подводя

итог тем действиям, которые участники решили предпринять, исходя из приобретенных ими знаний. Для этого достаточно упомянуть каждое действие, которое команда обязуется выполнить, а также работника, которому поручено его выполнение.

Завершение данного мероприятия служит также удобным моментом, чтобы поблагодарить всех присутствующих за участие в нем. Каждый участник должен сказать несколько слов благодарности в адрес тех, кто внес свой посильный вклад в результативное проведение данного мероприятия. Следует не забыть также выразить признательность и любым другим участникам проекта, кроме членов Scrum-команды, которые нашли время в своем плотном рабочем графике, чтобы принять активное участие в ретроспективе спринта.

И наконец, стоит уделить еще несколько минут, попросив членов команды высказать свои предложения относительно того, как улучшить порядок проведения ретроспективы спринта. Ведь ретроспектива спринта является неотъемлемой частью инфраструктуры Scrum, а следовательно, она должна подвергаться обследованию и адаптации.

Доведение дела до логического конца

Чтобы итоги ретроспективы спринта не остались только на бумаге, участники данного мероприятия должны довести выбранные действия до логического конца. Одни действия, в том числе и те, что обсуждаются на ежедневных летучках, должны быть лишь повторены или укреплены членами команды и Scrum-мастером. А другие действия должны быть выполнены в течение предстоящего мероприятия по планированию спринта.

Нередко самый простой способ выполнения улучшающих действий состоит в том, чтобы наполнить задел спринта задачами, соответствующими каждому действию перед вводом в него новых функциональных средств. Затем возможности, имеющиеся у команды для работы над новыми функциональными средствами, корректируются в сторону сокращения по оцененному времени, которое потребуется для выполнения задач улучшения. Откровенно говоря, в данном случае подойдет любой метод, позволяющий команде взять на себя реалистичные обязательства при планировании спринта и в то же время дающий возможность работать над улучшающими действиями.

Один из *непригодных* методов состоит в том, чтобы составить план "усовершенствования" отдельно от работы, которую команда должна выполнять в каждом спринте. Такой двухсторонний подход почти всегда приводит к тому, что план усовершенствования подчиняется типичному плану проведения спринта с ориентацией на функциональные средства. Чтобы улучшающие действия действительно произошли, их следует не отделять, а объединять!

Действия, не требующие затрат времени от членов команды, скорее всего, найдут свое место в перечне препятствий, которые должен устранить Scrum-мастер. А действия, которые предназначены для других команд или организаций, могут быть целиком размещены в соответствующем заделе для тех людей, которые должны работать над ними. Как правило, Scrum-мастер сотрудничает с внешними участниками проекта, чтобы обеспечить фактическое выполнение этих действий.

Вопросы проведения ретроспективы спринта

Тем не менее проведение ретроспективы спринта не лишено трудных вопросов. Сотрудничая со многими организациями, применяющими методику Scrum, я выявил целый ряд общих вопросов, характерных для проведения ретроспективы спринта (рис. 22.12).

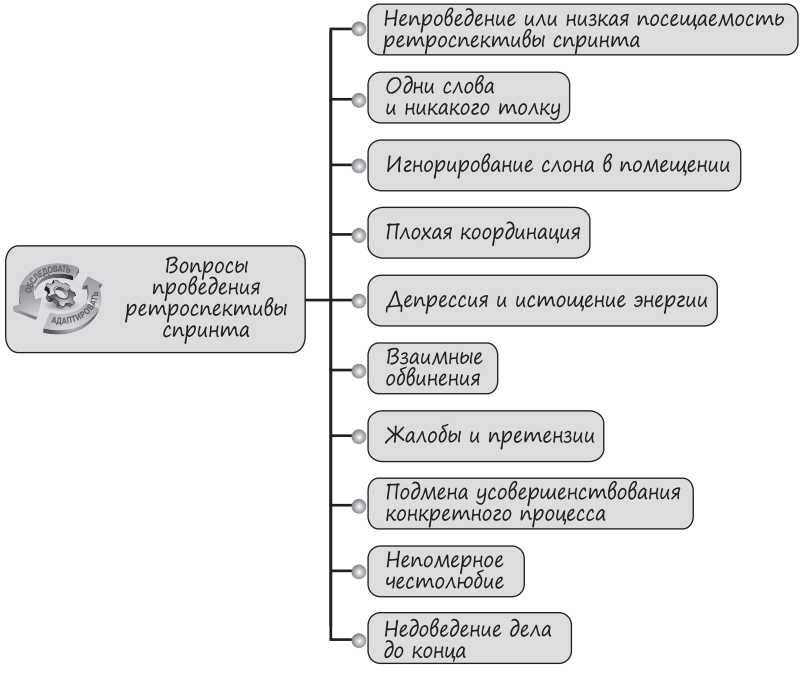


Рис. 22.12. Вопросы проведения ретроспективы спринта

Прежде всего, команду могут преследовать неудачи, когда она просто не проводит ретроспективу спринта, а если и проводит, то с низкой посещаемостью. Причины неудач в обоих случаях одинаковы. Если людям поручено работать с несколькими командами, то им трудно спланировать свой рабочий график, чтобы посещать все подобные мероприятия. Этот организационный недостаток

должны исправить руководители. А может быть, членам команды просто надоели подобные мероприятия, они разобщены или не особенно заинтересованы применять методику Scrum. Некоторые из них могут думать, что не стоит зря тратить время на работу, которая не входит в их основные обязанности (они, например, считают просто напрасным все, что не касается программирования и тестирования). Нередко подобное отношение возникает из-за недостаточно глубокого понимания той особенности методики Scrum, что она сосредоточена на непрерывном совершенствовании.

А порой возникает совершенно противоположное отношение, когда члены команды считают, что они уже достигли вершин в применении методики Scrum, и поэтому им нечему больше научиться из только что проведенного спринта у своих коллег по работе или извлечь полезный урок из своих удач и неудач. Если люди не видят никакой пользы в проведении ретроспективы спринта или в ее посещении, то следует рассмотреть возможность посвятить часть или всю следующую ретроспективу спринта исследованию этого вопроса.

Иногда низкая посещаемость объясняется тем, что удаленным участникам неудобно присоединиться к обсуждению по телефону или в режиме видеоконференции. Если удаленным участникам неудобно посещать ретроспективу спринта из-за намеченных сроков ее проведения, следует менять или циклически сдвигать время ее проведения, чтобы оно было удобным всегда и для всех. А если проведение ретроспективы спринта неудобно потому, что в ней просто трудно участвовать в удаленном режиме, следует пересмотреть текущую инфраструктуру дальней связи и порядок проведения упражнений, чтобы эффективнее задействовать удаленных участников.

Несмотря на то что некоторые ретроспективы проводятся весьма активно, они не достигают ничего практически полезного. Такие ретроспективы можно назвать следующим образом: *одни слова и никакого толку*. Если результатом ретроспективы оказываются одни только слова, то зря теряется драгоценное время. Чтобы от ретроспективы спринта был какой-то толк, полезно пригласить со стороны опытного координатора ретроспективы.

Некоторые ретроспективы просто занимательно наблюдать. Но в ходе данного мероприятия может быть так и не рассмотрен очевидный и важный вопрос, способный оказать значительное влияние на команду. Как говорится, слона-то никто и не приметил. Возможно, этот вопрос не был поднят из соображений безопасности. В подобных случаях Scrum-мастер должен взять на себя лидирующую роль, чтобы помочь команде в частности и организации вообще устранить, прежде всего, препятствия для безопасного обсуждения насущных вопросов разработки.

Ретроспектива не приносит порой никакой пользы из-за плохой координации. Возможно, новый Scrum-мастер в роли ее координатора и старается изо

всех сил, но у него ничего не получается. В таком случае следует пригласить со стороны опытного координатора, чтобы провести с его помощью несколько ретроспектив и приобрести необходимый опыт.

Некоторые ретроспективы приводят к депрессии и истощению энергии. Возможно, спринты проводятся неудовлетворительно, и люди рассматривают ретроспективу как мероприятие, на котором просто подытоживаются постигшие их неудачи, которые им приходится переживать снова. В таком случае следует уделить больше времени созданию подходящей атмосферы в самом начале ретроспективы спринта. Кроме того, привлечение со стороны координатора может дать больший эффект и помочь участникам сосредоточиться на положительных усовершенствованиях.

Нередко ретроспективы вызывают депрессию из-за того, что люди начинают искать виновных, опускаясь до взаимных обвинений. Координатор должен немедленно пресечь подобное поведение, как только оно проявится, чтобы не возникла цепная реакция взаимных обвинений.

А бывает и так, что ретроспектива спринта превращается в сплошные жалобы и претензии. Возможно, некоторые и считают полезным, чтобы участники просто излили душу или хотя бы высказались о том, как они воспринимают сложившееся положение дел. Но ведь у них нет ни малейшего желания поправить дело, они только жалуются. В таком случае следует пригласить на ретроспективу тех людей, которые действительно способны что-то изменить, и поговорить с ними откровенно, а не жаловаться в их отсутствие.

Еще одна неприятная ситуация возникает в том случае, если участники рассматривают ретроспективу спринта как удобное время для усовершенствования процесса, умаляя тем самым совершенствование конкретного процесса в течение спринта. Ретроспектива спринта — удобное время для размышления над проделанной работой и обсуждения путей улучшения сложившейся ситуации, но она никогда не подменяла усовершенствование конкретного процесса. Поэтому Scrum-мастер должен предусмотрительно способствовать внедрению здравых усовершенствований в конкретный процесс на протяжении спринта.

Иногда желания превышают наши возможности. Новые команды, заряженные и нацеленные на совершенствование, нередко проявляют непомерное честолюбие, ставя перед собой совершенно нереальные цели. Это может привести к большому разочарованию, если команде не удастся достичь честолюбивой цели. Поэтому Scrum-мастер должен быть начеку, напоминая участникам об их реальных возможностях внедрять усовершенствования и помогая им умерить свой пыл.

Вероятно, самый трудный вопрос возникает, когда работа над улучшающими действиями, выбранными в течение ретроспективы спринта, не доводится до логического завершения. Если эта работа не доводится до конца, то

время на ретроспективы оказывается потраченным впустую. В этой связи Scrum-мастер должен взять на себя лидирующую роль, помогая команде постоянно совершенствовать процесс разработки. Если же совершенствования не доводятся до логического конца, то Scrum-мастер должен проявить настойчивость, чтобы выявить вместе с командой первопричины и помочь членам команды преодолеть препятствия на пути усовершенствования процесса разработки.

Заключение

Ретроспектива спринта — это время, когда команда анализирует, насколько удачно она применяет методику Scrum, и намечает меры для усовершенствования процесса разработки. Ретроспектива спринта — это совместное мероприятие, в котором принимают участие члены Scrum-команды и любые заинтересованные лица по мере необходимости.

Как только подготовительная работа к ретроспективе спринта будет завершена, сам ход ретроспективы спринта состоит в том, чтобы создать сначала подходящую атмосферу для успешного проведения данного мероприятия; затем ввести всех присутствующих в общий контекст, предоставив исходные данные; далее выявить улучшающие наблюдения; выбрать улучшающие действия; а после этого завершить данное мероприятие. По завершении ретроспективы спринта очень важно, чтобы ее участники довели улучшающие действия до логического конца, что позволило бы команде работать эффективнее в следующем спринте. Не менее важно своевременно выявлять трудные вопросы, которые могут помешать успешному проведению ретроспективы спринта, быстро реагируя на них.

Глава 23
Дальнейшие действия

В предыдущих главах были рассмотрены составляющие инфраструктуры и пояснены основы методики Scrum. Теперь вы должны понимать механизм применения методики Scrum для гибкой разработки новаторских решений. Вы должны также ясно представлять причины, по которым методика Scrum предписывает конкретные роли, нормы практики, артефакты и правила. И теперь вы готовы к дальнейшим действиям. В этой главе обсуждается представление о том, что для реализации методики Scrum не существует единого, универсального пути к конечной цели. Вместо этого вам придется выбрать свой особый путь, чтобы достичь гибкости в разработке. В завершение этой главы описывается особая роль норм передовой практики, а также повторяющийся и постепенный подход к применению методики Scrum в качестве основания для обнаружения собственного пути дальнейшего продвижения вперед.

Конечного состояния в Scrum не существует

У каждой организации имеется свое представление о том, чего она стремится достичь. Применение методики Scrum способно помочь организациям так организовать свою работу, чтобы реализовать их представления. Но конечная цель состоит не столько в грамотном применении методики Scrum, а следовательно, в умении действовать гибко, сколько в более эффективном и экономически целесообразном достижении коммерческих целей. Так как же узнать, когда цель внедрения методики Scrum достигнута?

Дело в том, что не существует критерия готовности для принятия методики Scrum или перехода к ней. Не существует также модели зрелости гибкой разработки вроде модели CMMI [SEI. 2010], где цель состоит в попытке достичь уровня 5. Любая попытка определить критерий готовности для реализации методики Scrum предполагает, что когда достигается такое состояние, то дальнейшее совершенствование невозможно. Это явно означает, что методика Scrum является формой непрерывного совершенствования, чтобы всегда работать с целью лучше приспособить методику Scrum к сложной обстановке, в которой приходится разрабатывать продукцию. Хуже того, если попытаться определить такое конечное состояние для конкретной отрасли, то далее пришлось бы допустить,

что конечное состояние может быть достигнуто в каждой организации — даже в тех организациях, где разрабатываются принципиально разные виды продукции в совершенно разных условиях.

Если кто-нибудь заявит, что он наконец-то достиг искомой гибкости в своей работе, то такое заявление не имеет никакого смысла. Это положение Майк Кон точно подытожил следующим образом: “Гибкость означает добиться не чего-то конкретного, а чего-то большего” [Cohn, Mike. 2010]. Таким образом, в Scrum не существует какого-то конечного состояния, которое можно было бы назвать гибким. Напротив, совершенствование навыков применения методики Scrum и повышения гибкости процесса разработки происходит непрерывно, никогда не кончается и не достигает итоговой черты.

Обнаружение собственного пути

Как никто не может сказать, где должна оканчиваться реализация методики Scrum, так никто не в состоянии сказать, по какому именно пути следует идти, чтобы гарантировать успешное внедрение этой методики. Вместо этого вам придется постоянно изучать, обследовать и адаптировать методику Scrum, чтобы найти свой путь продвижения вперед, исходя из особых целей и культурных традиций в вашей организации и постоянно развивающейся сложной среде, где вам приходится действовать. Попытка следовать по чужому пути, используя чужие достижения, может показаться на первый взгляд быстрым способом добиться гибкости в разработке. Но ведь не существует двух одинаковых организаций или даже двух одинаковых команд в одной и той же организации. Поэтому, следуя по чужому пути, можно прийти совсем не туда, куда нужно.

Обойтись без процесса обучения никак нельзя. Напротив, нужно уметь быстро замыкать свои петли обучения, проводя обследование и адаптацию на основании того, что было изучено. Это совсем не означает, что нужно пренебрегать опытом тех, кто раньше стал на путь гибкой разработки. Изучайте их опыт и достижения в гибкой разработке, но старайтесь обнаружить свой путь, чтобы действовать еще более гибко.

Соблюдение общих норм передовой практики

Если не следовать по чужому пути, то какие нормы передовой практики лучше всего подойдут для гибкой разработки? Как не существует единого пути для следования, так и не существует единых норм передовой практики для всех организаций.

Когда меня просят описать нормы передовой практики, применяемые в других организациях с целью работать гибко, я обращаюсь к конкретным примерам. Но я всегда привожу их в соответствующем контексте других организаций, чтобы те, кто спрашивает, могли оценить, стоит ли им применять аналогичный подход в своей организации. Даже в масштабах одной организации нужно очень осторожно подходить к универсальному применению норм передовой практики. Во многих организациях пытаются описать достижения Scrum-команд, зафиксировать, а затем узаконить их как нормы передовой практики. Поступать так вредно потому, что подходы, применяемые одними командами, могут не подойти для других команд.

Обратите внимание на то, что в предыдущем абзаце слово *подход* было употреблено неоднократно. Рассмотрим вкратце его отличие от термина *норма практики*, употреблявшего в предыдущих главах данной книги. Норма практики обозначает основную или существенную особенность методики Scrum, тогда как подход — конкретную реализацию нормы практики в Scrum. Поэтому, когда меня спрашиваются о *нормах передовой практики*, я считаю, что имелись в виду *передовые подходы*.

Несмотря на то что две разные команды или организации внедряют методику Scrum по-своему, каждая из них должна придерживаться одних и тех же норм практики в Scrum. Например, в обеих организациях должны быть сформированы Scrum-команды, состоящие из владельца продукта, Scrum-мастера и команды разработчиков. Обе команды должны выполнять планирование спринта, проводить ежедневные летучки, подведение итогов и ретроспективу спринта.

Ежедневная летучка является основной нормой практики в Scrum. Если она не проводится, то и методика Scrum не соблюдается. В ходе ежедневной летучки каждый член команды разработчиков сообщает о своих последних достижениях и согласует свои действия со всеми остальными членами команды, чтобы получить общее представление о всем объеме работ. Но когда ежедневная летучка начинается, то кто должен первым сообщить о своих последних достижениях? Это в Scrum не определяется. У каждой команды свой подход.

Например, работая с командой в канадском г. Ванкувер, я узнал об интересном подходе к тому, кто первым должен выступить с сообщением на ежедневной летучке. В начале каждой ежедневной летучки Scrum-мастер этой команды подбрасывал вверх игрушечного плюшевого лосенка. И кто ловил его, тот и говорил первым, а затем справа налево по очереди высказывались остальные члены команды разработчиков. И такой простой, но забавный подход оказался вполне пригодным для данной команды из Ванкувера.

Как оказалось, у команды из Ванкувера была родственная команда в Китае, которая сформировалась через несколько месяцев после нее. Однажды член китайской команды обратился к члену канадской команды за советом по поводу “правила

или нормы передовой практики" для предоставления первого слова на ежедневной летучке. Член канадской команды сказал ему, что с этой целью они "подбрасывают вверх игрушечного лосенка" на каждой ежедневной летучке. Очевидно, что для китайца такой прием означал совсем другое: то, что годилось для канадской команды, было совершенно неприемлемо для китайской! И тогда в китайской команде был принят свой подход, как, впрочем, и следовало поступить.

В методике Scrum определяются основные нормы практики, которым нужно следовать. Но каждая команда вправе выбирать те подходы (или нормы передовой практики), которые наиболее для нее пригодны. Таким образом, подходы являются особыми для каждой команды, и поэтому их можно и должно применять в других командах лишь в том случае, если это целесообразно для них.

Обнаружение пути дальнейшего продвижения по методике Scrum

Те, кто лишь начинает осваивать или уже имеет опыт применения методики Scrum для разработки продукции, могут руководствоваться принципами Scrum, чтобы найти свой путь для дальнейшего продвижения. В своей книге *Succeeding with Agile* [Cohn, Mike. 2009] Майк Кон подробно описывает данный подход, и поэтому я отсылаю читателя к этой отличной книге за подробным изучением данного вопроса.

А здесь я постараюсь разъяснить суть данного подхода на конкретном примере. В 2007 году мне пришлось длительное время наставлять и обучать внедрению и применению методики Scrum работников крупной международной организации. Отделение информационных технологий в этой организации насчитывало 100 сотрудников в Нью-Йорке и 400 сотрудников в индийском городе Мумбай. Одновременно в этой организации велось около 45 проектных работ.

В этой организации решили, что в помощь каждой новой команде, собиравшейся применять методику Scrum, должен быть назначен свой наставник. Ведь при ограниченном числе наставников было бы нелегко перевести сразу всех сотрудников отделения информационных технологий на методику Scrum. Поэтому данная организация сначала выбрала для этой цели небольшое количество пилотных проектных работ, как обычно и поступают в подобных случаях. При этом преследовалась цель — постепенно перевести на методику Scrum остальные команды по мере расширения возможностей обучения благодаря увеличению числа внутренних наставников.

Пилотные проектные работы охватывали широкий спектр: от сопровождения простых систем до разработки крупных новых продуктов. Исходя из этого разнообразия, каждая Scrum-команда реализовала свой вариант инфраструктуры

Scrum, применяя подходы, пригодные для членов команды и той работы, которую они выполняли. А для изучения в масштабах всей организации разных подходов, применявшихся каждой командой, и обмена опытом между командами применялась Википедия.

Через несколько месяцев после принятия методики Scrum настало время для ее переноса с уровня отдельных команд на уровень целой организации. И тогда было создано то, что Майк Кон называет ETC (Enterprise Transition Community — Сообщество перехода на уровень предприятия) [Cohn, Mike. 2009]. В данном случае это сообщество получило название Working Software Group (Рабочая группа для разработки программного обеспечения). Группа руководителей и администраторов вела задел элементов, связанных с улучшениями, и выполняла трехнедельные спринты для реализации элементов из этого задела. Эти элементы представляли собой организационные инициативы для внесения изменений (например, обновить компенсационную модель, чтобы в большей степени сосредоточить ее на командах) или значительные препятствия, мешавшие Scrum-командам нормально работать (например, повысить устойчивость работы серверов, чтобы команды могли вовремя завершить тестирование).

Реализуя элементы из задела усовершенствований в последовательности спринтов, организация сумела неизменно и постепенно добиваться прогресса на своем пути к успешному внедрению методики Scrum. У этой организации не было предопределенного конечного состояния в применении методики Scrum. Ведь пытаться определить такое состояние заранее было бы напрасной тратой времени, как, впрочем, и составлять заранее полный перечень требований к совершенно новому продукту, в котором никто толком не разбирался.

Вместо этого группа Working Software Group брала исходные данные у Scrum-команд и прочих участников проекта и постепенно вносила усовершенствования в организационную структуру, чтобы точнее соответствовать ценностям гибкой разработки. Благодаря постоянному обучению, обследованию и адаптации данной организации удалось найти подходящий путь продвижения, согласованный с ее общими коммерческими целями.

Такой образец ETC ныне весьма распространен. Во многих организациях уже поняли, что применение самой методики Scrum для ее же внедрения служит вполне благоразумным путем к неуклонному и постепенному переходу на гибкие принципы разработки.

Итак, за дело!

Меня часто удивляет, когда одни и те же люди, которые считают, что невозможно составить заранее и правильно требования к разрабатываемому продукту, а следовательно, разработку следует вести по методике Scrum, пытаются,

тем не менее, пояснить, что они не готовы приступить к применению методики Scrum, поскольку еще не выработали во всех подробностях свой подход к Scrum! Такого рода рассуждения плохо согласуются с основополагающими принципами Scrum.

Применяя методику Scrum, можно не беспокоиться преждевременно о достижении совершенства. А любые попытки достичь совершенства заранее вынудят теряться в догадках за счет важных результатов обучения, которые могут быть получены только в том случае, если применять методику Scrum и смотреть, что из этого получится. Как показывает мой опыт, большинство команд выполняют первые два спринта далеко не идеально, и это нормально. Но мне приходилось наблюдать лишь одно: с каждым последующим спринтом любая Scrum-команда только улучшала свои показатели по сравнению с предыдущим. Таким образом, откладывать начало применения методики Scrum не стоит. Какими бы знаниями вы ни обладали в настоящий момент о применении методики Scrum, только представьте, сколько полезного вы узнаете после того, как проведете свой первый спринт!

Не следует, однако, ожидать, что принятие методики Scrum может пройти без сучка и задоринки. Могу смело гарантировать, что в какой-то момент ваша организация столкнется с препятствиями, которые затруднят разработку по методике Scrum. Ведь эта методика позволяет выявить нарушения, пренебрежение которыми не дает организации раскрыть свой истинный потенциал. Но в то же время она не дает организации ответа на вопрос, как устранить подобные нарушения. И этот нелегкий труд ложится на плечи самих сотрудников организации.

Существующий порядок — крепкий орешек, который не так-то просто расколоть. Зачастую людям проще пренебречь методикой Scrum или изменить ее, чем вносить коррективы в устоявшиеся организационные процессы, правила или режимы. А культура, враждебная выявлению нарушений в нормальной работе, быстро скрывает в тени свои изъяны от яркого света, который безжалостно их разоблачает. Чтобы противодействовать этой тенденции, следует настойчиво и терпеливо внедрять перемены в своей организации. При этом важно понять, что сопротивление переменам вполне естественно. Поэтому нужно помочь преодолеть худшие его последствия, обучая других принципам, положенным в основу методики Scrum, а также разъяснять цели, которых она позволяет достичь. С теми, кто сопротивляется переменам, нужно работать вместе, а не против них, чтобы преодолеть препятствия, мешающие нормальной работе команды, проведению проектных работ, а организации — полностью реализовать преимущества от внедрения методики Scrum.

А мне остается только надеяться, что эта книга позволила вам получить прочные знания основ Scrum, чтобы пролить яркий свет на свой дальнейший путь. Желаю вам добиться немалых успехов на этом пути с помощью методики Scrum.

Приложение А
Словарь терминов

Краткий обзор

Статьи в данном словаре организованы в алфавитном порядке. Отдельная статья может состоять из одного слова, например, *Scrum*, фразы вроде *критерии приемки* или сокращения типа *TDD*. Если же термин имеет не одно определение, то он нумеруется.

Для обозначения взаимосвязи между терминами в словаре употребляются следующие ссылки:

- *См.* — делается на предпочтительный термин или же такой термин, определение которого служит в качестве дополнения рассматриваемого термина.
- *См. также* — делается на термин, связанный с данным.
- *Синоним* — делается на синоним или термин с близким значением.
- *Ср.* — делается на термин с противоположным значением.

Определения

ATTD (Acceptance-test-driven development). *См. разработка посредством приемочного тестирования.*

Cynefin. Смыслообразующая инфраструктура, помогающая лучше понять ситуацию, в которой приходится действовать, и выбрать подход, наиболее пригодный для данной ситуации [Snowden, David J., and Mary E. Boone. 2007].

DEEP. Сокращение, придуманное Романом Пихлером и Майком Коном для удобства запоминания ряда критериев, применяемых для оценивания качества задела продукта. Эти критерии следующие: *должным образом детализируемый*, *развивающийся*, *оцениваемый*, *приоритизированный задел продукта* (*Detailed appropriately*, *Emergent*, *Estimated*, *Prioritized*).

INVEST. Сокращение, придуманное Биллом Уэйком для удобства запоминания ряда критериев, применяемых для оценивания качества пользовательских историй. Эти критерии следующие: *независимая*, *обсуждаемая*, *ценная*, *оцениваемая*, *мелкая*, *тестируемая история* (*Independent*, *Negotiable*, *Valuable*, *Estimatable*, *Small*, *Testable*). *См. также пользовательская история.*

JIT (Just in time). *См. своевременный или оперативный.*

Kanban. Гибкая методика, внедряемая в существующий процесс с целью наглядно показать порядок выполнения работ в системе, ограничивая незавершенные работы, измеряя и оптимизируя ход работ. *См. также гибкий, незавершенные работы.*

LRM (Last responsible moment). *См. последний ответственный момент.*

MMF (Minimum marketable feature). *См. минимально пригодные для продажи функциональные средства.*

MRF (Minimum releasable feature). *См. минимально выпускаемые функциональные средства.*

MVP (Minimum viable product). *См. минимально жизнеспособный продукт.*

PBI (Product backlog item). *См. элемент задела продукта.*

Scrum. Термин, заимствованный из регби, где он обозначает *схватку* за мяч. **1.** Простая инфраструктура для ведения гибкой разработки продукции или услуг. **2.** Методика итеративной и инкрементной разработки продукции и управления ходом проектных работ. *См. также гибкий, инфраструктура Scrum.*

Scrum-команда. Команда, состоящая из владельца продукта, Scrum-мастера и команды разработчиков, выполняющих проектные работы по методике Scrum. *См. также команда разработчиков, владелец продукта, Scrum-мастер.*

Scrum-мастер. Наставник, координатор, устранитель препятствий и лидер-служитель в Scrum-команде. Scrum-мастер выполняет одну из трех ролей в Scrum-команде. Он обеспечивает руководство процессом разработки и помогает Scrum-команде и остальным сотрудникам организации выработать высокоэффективный подход к разработке по методике Scrum с учетом особенностей данной организации. *См. также Scrum-команда, лидер-служитель.*

SoS (Scrum of Scrums). *См. схватка над схватками.*

TDD (Test-driven development). *См. разработка посредством тестирования.*

Т-образные навыки. Метафора, употребляемая для описания лица с глубокими по вертикали навыками в специальной области (например, в проектировании взаимодействия с пользователем), а также широкими, но не обязательно глубокими навыками в других подходящих областях (например, в тестировании или документировании). Члены команды разработчиков с Т-образными навыками больше приспособлены к такому поведению, как роение. *См. также роение.*

WIP (Work in process). *См. незавершенные работы.*

WSJF (Weighted shortest job first). *См. самая коротковзвешенная работа.*

XP (Extreme Programming). *См. экстремальное программирование.*

Адаптация. Один из трех столпов управления эмпирическим процессом; ответная реакция, используемая для внесения корректив в разрабатываемый продукт или рабочий процесс его разработки. *См. также управление эмпирическим процессом, обследование, прозрачность.*

Артефакт. Ощутимый побочный продукт, производимый в течение разработки продукции. Примерами артефактов в Scrum служат задел продукта, задел спринта и прирост потенциально готового к поставке продукта. *См. также норма практики.*

Беспорядочная область. Опасное место, поскольку непонятно, как осмыслить сложившуюся ситуацию. Самое главное — выйти из этой области. **2.** Одна из областей в инфраструктуре Cynefin. *См. также Cynefin. Ср. сложная область, комплексная область, хаотичная область, простая область.*

Быстрый провал. Стратегия опробования чего-нибудь, получения скорой ответной реакции и быстрого обследования и адаптации. При наличии большой степени неопределенности нередко оказывается дешевле начать работу над продуктом, выяснив, насколько верным было такое решение. Если же оно окажется неверным, то работа быстро прекращается, чтобы не тратить лишние средства. *См. также скорая ответная реакция, обследование и адаптация, резкая смена стратегии.*

Вид деятельности. 1. Норма практики или мероприятие в Scrum, подразумевающие возможность предпринять действие или выполнить процесс, например, планирование спринта, ежедневную летучку, тестирование, оценивание и т.д. *См. также норма практики.*

Владелец продукта. Главное лицо, наделенное полномочиями руководить разработкой продукта. Одна из трех ролей в Scrum-команде. Единственный голос сообщества участников проекта в Scrum-команде. Владелец продукта определяет, что нужно делать и в каком именно порядке. *См. также Scrum-команда.*

Внешние участники проекта. Те участники проекта, которые, как правило, не являются сотрудниками организации, разрабатывающей продукт, например, заказчики, партнеры и регуляторные органы. *См. также участники проекта. Ср. внутренние участники проекта.*

Внутренние участники проекта. Те участники проекта, которые работают в организации, разрабатывающей продукт, например, высшее руководство, руководители подразделений и внутренние пользователи. *См. также участники проекта. Ср. внешние участники проекта.*

Внутрипроцессный продукт. Продукт, который разрабатывается в настоящее время, уже находится в эксплуатации или продается. *См. также планирование портфеля заказов.*

Водопад. Термин, обозначающий графическое представление такого процесса разработки, в котором последовательные стадии работы показаны неуклонно протекающими вниз, как по каскадам водопада. *См. также плановый процесс.*

Водопадная схватка. Наложение водопадного стиля разработки на инфраструктуру Scrum. Примером тому может служить последовательное выполнение спринтов анализа, проектирования, программирования и тестирования. *Синоним схваткопада.*

Водопадный процесс. *См. плановый процесс.*

Возможности. 1. Количество ресурсов, доступных для выполнения полезной работы. **2.** Понятие, употребляемое для определения предела незавершенных работ, чтобы начать работу только тогда, когда имеются возможности завершить ее. *См. также незавершенные работы.*

Возникающая возможность. Возможность, которая раньше не была известна или считалась маловероятной, а следовательно, на нее не имело смысла тратить средства.

Возникновение. 1. Отдельное локализованное поведение, собирающееся в глобальное поведение, отделяемое от его истоков. **2.** Свойство сложных адаптивных систем. **3.** Применительно к разработке программного обеспечения означает признание того, что изначально нельзя составить подходящий набор функциональных средств, проектных решений и планов. Вместо этого в процессе обучения возникает важная информация из опыта предыдущей работы. *См. также сложная адаптивная система.*

Временная шкала событий. Наглядное представление в хронологическом порядке важных событий, происходивших в течение некоторого периода времени. Типичная методика, применяемая при проведении ретроспективы спринта. *См. также ретроспектива спринта.*

Временные рамки. Фиксированный период времени, в течение которого осуществляется определенный вид деятельности. Спринты в Scrum являются ограниченными временными рамками итерациями, в которых команда работает в постоянном темпе над завершением выбранного ряда работ с установленным пределом для незавершенных работ. *См. также спринт, ограничение по времени.*

Выпуск. 1. Сочетание функциональных средств, совместно образующих продукт, поставляемый заказчикам или пользователям. **2.** Версия продукта, распространяемая для практического применения или развертывания. Выпуски обозначают ритм доставки коммерческой ценности и поэтому должны быть согласованы с определенными деловыми циклами.

Выпуск с жестко заданным объемом работ. Выпуск, который должен иметь конкретный ряд функциональных средств. Сроки выпуска этих средств и/или затраты на их выпуск должны быть гибкими. *Ср. выпуск с жестко заданными сроками.*

Выпуск с жестко заданными сроками. Выпуск, который должен быть завершен к известной будущей дате. Объем работ, а возможно, и затраты на выпуск должны быть гибкими. *Ср. выпуск с жестко заданным объемом работ.*

Выпускной поезд. Метод согласования концепции, планирования и взаимозависимостей между многими командами с целью синхронизировать их работу, опираясь на размеренный темп. Выпускной поезд сосредоточен на быстром и гибком потоке доставки ценности на уровне крупного продукта. *См. также схватка над схватками.*

Выработка общего замысла. Вид деятельности с целью схватить суть потенциального продукта и составить приблизительный план его разработки. Выработка общего замысла начинается с создания концепции продукта, затем формируется задел продукта и зачастую составляется график его выпуска. *Синоним планирования продукта. См. также график выпуска продукта.*

Гибкий. 1. Конкретный ряд ценностей и принципов, выраженный в документе "Манифест гибкой разработки" [Agile Manifesto, Beck et al. 2001]. **2.** Обобщающий термин, употребляемый группой взаимосвязанных методик и подходов к разработке программного обеспечения, опирающихся на итеративную и инкрементную разработку. В частности, Scrum является методикой гибкой разработки. *См. также экстремальное программирование*, *Kanban*, *Scrum.*

Главный владелец продукта. Общий владелец продукта в команде владельцев продукта при выполнении крупных проектных работ. *См. также владелец продукта.*

Готовность. *См. критерий готовности.*

График выпуска продукта. Описание постепенного характера разработки продукта и его выпуска со временем, наряду с важными факторами, побуждающими к каждому отдельному выпуску. Полезен при разработке продукта, для которого планируется несколько выпусков. *См. также выработка общего замысла.*

Группа. Собрание людей под общим названием. Но это еще не команда, члены которой научились работать вместе и доверять друг другу. *Ср. команда.*

Демонстрация спринта. 1. Мероприятие, проводимое при подведении итогов спринта для показа готовых элементов задела продукта, чтобы дать больше информации для обсуждения итогов спринта Scrum-командой и прочими участниками. **2.** Термин, нередко употребляемый в качестве синонима для обозначения всего подведения итогов спринта. *См. также подведение итогов спринта.*

Диаграмма выгорания. График, показывающий продвижение работы к *целевой линии*, связанной с объемом работ, откладываемым по вертикальной оси. По мере завершения работы во *времени*, откладываемом по горизонтальной оси, линия ее продвижения смешается вверх (выгорает), приближаясь к целевой линии. На диаграмме выгорания можно отобразить намеченные результаты, рассчитав и проведя линию тенденции, чтобы показать, когда работа может быть завершена. *Ср. диаграмма сгорания.*

Диаграмма сгорания. График, на котором по вертикальной оси откладывается *объем работ* (в человеко-часах или единицах измерения элементов из задела продукта), оставшихся выполнить во *времени*, откладываемом по горизонтальной оси. Со временем работы остается все меньше и меньше, и поэтому на данной диаграмме демонстрируется тенденция к сгоранию до того момента, когда работы больше не останется. На диаграмме сгорания можно отобразить

намеченные результаты, рассчитав и проведя линию тенденции, чтобы показать, когда работа может быть завершена. *Ср. диаграмма выгорания.*

Доска задач. Источник распространения информации, применяемый при выполнении спринта для сообщения о достигнутом прогрессе и ходе работ в течение спринта. *См. также источник распространения информации, задача.*

Единовременная разработка. Метод рационального выполнения всех видов работ (например, анализа, проектирования, программирования, интеграции и тестирования) в течение одной итерации.

Ежедневная летучка. Мероприятие по синхронизации, обследованию и адаптивному планированию, которое команда разработчиков проводит каждый день. Основная норма практики в инфраструктуре Scrum, на которую отводится не больше 15 минут. *Синоним ежедневного стояния. См. также обследование и адаптация.*

Ежедневное стояние. Типичный подход к выполнению ежедневной летучки, состоящий в том, что участники стоят на протяжении всего мероприятия. Стояние способствует краткости и помогает провести данное мероприятие в установленный срок. *См. ежедневная летучка.*

Желательные функциональные средства. Функциональные средства, которые намечены в предстоящем выпуске, но могут быть исключены из него, если для завершения их разработки окажется недостаточно ресурсов. *Ср. обязательные функциональные средства, необязательные функциональные средства.*

Задача. Техническая работа, которую выполняет команда разработчиков, чтобы завершить элемент задела продукта. Большинство задач определяются как мелкие и требующие для работы над ними от нескольких часов до целого дня.

Задел наблюдений. Приоритетный список сформированных ранее наблюдений или представлений о совершенствовании процесса разработки, которые требуется реализовать. Задел наблюдений формируется и применяется при проведении ретроспективы спринта. *См. также ретроспектива спринта.*

Задел портфеля заказов. Задел, состоящий из продуктов, программ, проектов или высокоуровневых эпических историй. *См. также планирование портфеля заказов.*

Задел продукта. Приоритетный список элементов, работа над которыми еще не началась. *См. также элемент задела продукта.*

Задел спринта. 1. Артефакт, получаемый на совещании по планированию спринта, непрерывно обновляемый при выполнении спринта и помогающий самоорганизующейся команде лучше спланировать и вести работу, которую требуется выполнить, чтобы достичь намеченной цели спринта. **2.** Список элементов, извлекаемых из задела продукта в спринт и связанных с планом их реализации. Нередко выражается в виде отдельных задач, оцениваемых в идеальных часах. *См. также идеальный час, планирование спринта, задача.*

Заместитель владельца продукта. Лицо, назначаемое владельцем продукта, чтобы действовать от его имени в особых случаях. *См. также владелец продукта.*

Запасы. *См. незавершенные работы.*

Зарождение проекта. *См. составление положения о проекте.*

Идеальный день. Единица оценивания размеров элементов из задела продукта, исходя из продолжительности работы над элементом, если бы она выполнялась только над ним, не прерывалась, а для ее завершения имелись бы все необходимые ресурсы. *См. идеальный час. См. также очки за историю.*

Идеальный час. Единица оценивания объема работ по проектированию, построению, интеграции и тестированию, представленных в виде отдельных задач в заделе спринта. Нередко обозначается как человеко-час. *См. также идеальный день.*

Известный технический долг. Категория состояния технического долга, обозначающая такой технический долг, который известен команде разработчиков и выявлен для последующего рассмотрения. *Ср. случайно обнаруженный технический долг, намеченный технический долг. См. также технический долг.*

Излишняя формальность. 1. Церемония, которая имеет реальную стоимость, но доставляет малую ценность или вообще не доставляет ее (форма расточительства). **2.** Процесс ради процесса. *См. также церемония, расточительство.*

Изменчивость. Разброс или дисперсия ряда данных, представляющих неодинаковые результаты. В производстве изменчивость всегда означает убытки и расточительство. А в разработке продукции некоторая изменчивость просто необходима для создания новаторских решений. *См. также расточительство.*

Инкрементная разработка. 1. Разработка, основанная на принципе построения *части*, прежде чем построить *все*. **2.** Постановочная стратегия для разработки и доставки продукта пользователям по частям в разное время с намерением адаптироваться к внешней ответной реакции. *См. также итеративный и инкрементный процесс, итеративная разработка.*

Интеграция. Сочетание различных компонентов или ресурсов части или всего продукта для формирования связного, требующего большего объема работ продукта, который может быть проверен на правильность функционирования в целом. *См. также непрерывная интеграция.*

Инфраструктура Scrum. Совокупность ценностей, принципов, норм практики и правил, образующих основу для гибкой разработки по методике Scrum. *См. также Scrum.*

Инфраструктура. *См. инфраструктура Scrum.*

Использование. Принятие решения исходя из определенности информации, которая имеется в данный момент. *Ср. исследование.*

Исследование. Акт приобретения знаний в результате некоторого вида деятельности, в частности, построения прототипа, создания опытного образца, изучения и экспериментирования. *Ср. использование.*

История. *См. пользовательская история.*

Источник распространения информации. Наглядное пособие для представления актуальной, достаточно подробной и важной информации в простом и удобном для беглого осмотра формате.

Итеративная разработка. Стратегия планируемой переделки, когда работа выполняется в несколько проходов для достижения наилучшего решения. *См. также инкрементная разработка, итеративный и инкрементный процесс.*

Итеративный и инкрементный процесс. Методика разработки, эффективно использующая особенности как итеративной, так и инкрементной разработки. *См. также инкрементная разработка, итеративная разработка.*

Итерация. Обособленный цикл разработки, сосредоточенный на выполнении всей работы, требующейся для получения ценного результата. *См. также единовременная разработка, спринт.*

Команда для компонентов. Это команда, уделяющая основное внимание созданию одного или нескольких компонентов крупного продукта, который заказчик готов приобрести. Команды для компонентов создают ресурсы или компоненты, которые затем повторно используются другими командами для разработки решений, представляющих определенную ценность для заказчиков. *Ср. команда для функциональных средств.*

Команда для функциональных средств. Межфункциональная и межкомпонентная команда, которая может извлекать функциональные средства конечного пользователя из задела продукта и реализовывать их. *См. также межфункциональная команда. Ср. команда для компонентов.*

Команда разработчиков. Самоорганизующаяся, межфункциональная команда людей, несущих коллективную ответственность за всю работу, которую требуется выполнить для производства рабочих, проверенных ресурсов. Одна из трех основных ролей, которые составляют каждую Scrum-команду. *См. также межфункциональная команда, владелец продукта, Scrum-мастер, Scrum-команда.*

Команда. Небольшое межфункциональное собрание разных, сотрудничающих вместе людей для достижения общей цели. Члены команды доверяют друг другу и работают вместе для достижения общей цели, отвечая друг перед другом за конечный результат. *Ср. группа.*

Комплексная область. 1. Ситуация, которая может оказаться более непредсказуемой, чем предсказуемой. Если и имеется правильный ответ, то его можно узнать только задним числом. **2.** Одна из областей в инфраструктуре Cynefin. *См. также Cynefin. Ср. сложная область, хаотичная область, беспорядочная область, простая область.*

Конечная неопределенность. Неопределенность, связанная с тем, что именно будет создано (конкретный продукт). *См. также неопределенность. Ср. неопределенность средств.*

Концепция продукта. Краткое описание желательного будущего состояния, которое будет достигнуто в результате разработки и развертывания продукта. Грамотно сформулированная концепция должна быть простой, обеспечивая согласованное направление для тех людей, которым предлагается ее реализовать. *См. также выработка общего замысла.*

Критерии приемки. 1. Характеристики внешнего качества, обозначаемые владельцем продукта с точки зрения руководства организации или участников проекта. Критерии приемки определяют желательное поведение и служат для определения степени успешности разработки элемента из задела продукта. **2.** Критерии завершения, которым должны соответствовать компонент или система, чтобы быть принятыми пользователем, заказчиком или другим официально уполномоченным органом (Стандарт IEEE 610).

Критерий готовности. 1. Контрольный список видов работ, которые команда предполагает успешно завершить к концу спринта, прежде чем объявить свою работу потенциально пригодной к поставке. Самый минимальный критерий готовности должен определять готовую часть функциональных возможностей продукта, как только она будет спроектирована, построена, интегрирована, проверена, задокументирована и доставит достоверную потребительскую ценность. **2.** Иногда описывается как критерий приемки применительно ко всем элементам задела продукта в целом. *Ср. критерий подготовленности.*

Критерий подготовленности. Контрольный список условий, которые должны быть соблюдены, чтобы считать элемент задела продукта готовым к реализации в течение планируемого спринта. *Ср. критерий готовности.*

Лидер-служитель. 1. Лицо, добивающееся результатов для своей организации, уделяя основное внимание потребностям его коллег и тех, кому оно служит. **2.** Принцип и норма практики лидерства, основывающиеся на способности слушать, сопереживать, исцелять, быть на чеку, убеждать, осмысливать, предвидеть, брать на себя обязательства, развивать общественные отношения. *См. также Scrum-мастер.*

Лицо. 1. Архетип пользователя, составленный из этнографических сведений о реальных пользователях и помогающий принимать решения относительно функциональных средств продукта, требующихся для перемещения и взаимодействия с его пользовательским интерфейсом, а также для графического оформления последнего. **2.** Вымышленное лицо, являющееся прототипом конкретной роли пользователя. *См. также пользовательская история.*

Межфункциональная команда. Команда, состоящая из членов со всеми функциональными навыками вроде проектирования пользовательского интерфейса, программирования и тестирования, а также специальностями, необходимыми для завершения работы, которая требует опыта и знаний не только в одной дисциплине.

Мера относительного размера. Средство для выражения общего размера элемента безотносительно к его абсолютному размеру, но в сравнении с размерами других элементов. Например, относительный размер **2** одного элемента наполовину меньше относительного размера **4** другого элемента, хотя эти размеры не дают никакого представления об истинной величине элемента в абсолютном измерении. *См. также идеальный день*, *очки за историю*.

Методика. Определенная процедура, применяемая для выполнения некоторых или всех видов деятельности или для поддержки принятого подхода. *См. также вид деятельности*, *подход*.

Минимально выпускаемые функциональные средства. 1. Минимальный набор функциональных средств, которые должны присутствовать в выпуске, чтобы сделать его минимально жизнеспособным, т.е. полезным для конечных потребителей настолько, что они захотят его иметь и охотно заплатят за него. **2.** Функциональные средства, входящие в набор минимально пригодных для продажи средств. *Синоним обязательных функциональных средств*. *См. также минимально пригодные для продажи функциональные средства*.

Минимально жизнеспособный продукт. Продукт, обладающий лишь такими функциональными средствами, которые позволяют только развернуть его, но не более того.

Минимально пригодные для продажи функциональные средства. Наименьший или минимальный набор функциональных средств, которые должны быть доставлены потребителю, чтобы считаться ценными, т.е. пригодными для продажи. *Ср. минимально выпускаемые функциональные средства*.

Мушкетерские отношения. 1. Один за всех и все за одного. **2.** Такие отношения между членами команды разработчиков, когда все они находятся как бы в одной лодке, добиваясь успеха или терпя неудачи вместе как единая команда.

Наивный технический долг. Форма технического долга, накапливающегося вследствие безответственного поведения или применения незрелых норм практики теми, кто участвует в процессе разработки. *Ср. стратегический технический долг*, *неизбежный технический долг*. *См. также технический долг*.

Намеченный технический долг. Категория состояния технического долга, представляющая такой технический долг, который известен и намечен для обслуживания командой разработчиков. *Ср. случайно обнаруженный технический долг*, *известный технический долг*. *См. также технический долг*.

Негласное группирование. Методика координирования с целью сгруппировать людей по связанным элементам без лишних слов, но опираясь на расположение и перемещение элементов (обычно в виде карточек и наклеек для заметок) на доске в качестве средства сообщения и координирования действий участников. Такая методика нередко применяется при проведении ретроспективы спринта. *См. также ретроспектива спринта*.

Незавершенные работы. Работы, с которых был начат процесс разработки, но они еще не завершены, а их результаты недоступны для заказчика или пользователя. Обозначает все активы или рабочие варианты продукции или услуги, которые еще находятся в работе или ожидают своей очереди для завершения.

Неизбежный технический долг. Форма технического долга, который обычно непредсказуем, неотвратим и накапливается, не мешая, тем не менее, команде разрабатывать продукт. *Ср. наивный технический долг*, *стратегический технический долг*. *См. также технический долг*.

Неизвестные неизвестные. Факторы, о которых мы еще знаем, что они нам неизвестны.

Необязательные функциональные средства. Ряд средств, которые объявлены как не обязательно входящие в предстоящий выпуск. *Ср. желательные функциональные средства*, *обязательные функциональные средства*.

Неопределенность потребителя. Неопределенность в том, кто именно является потребителями продукта. *См. также неопределенность*. *Ср. конечная неопределенность*, *неопределенность средств*.

Неопределенность средств. Неопределенность, связанная с тем, как выполнять разработку. *См. также неопределенность*. *Ср. неопределенность потребителя*, *конечная неопределенность*.

Неопределенность. Нечто неизвестное или неустановившееся. Нередко считается синонимом риска, но на самом деле является более широким понятием, поскольку неопределенность включает в себя не только риски (отрицательные последствия), но и возможности (положительные последствия). *См. также риск*.

Непрерывная доставка. *См. непрерывная разработка*.

Непрерывная интеграция. Норма инженерной практики, когда члены одной команды или нескольких команд объединяют результаты своей работы настолько часто, насколько это практически целесообразно. *См. также интеграция*, *нормы инженерной практики*.

Непрерывная разработка. Развертывание каждого нового для пользователей функционального средства сразу же после его построения, интеграции и тестирования. *Синоним непрерывной доставки*, *интеграции*.

Нефункциональное требование. 1. Требование, связанное не с функциональными возможностями, а с такими свойствами, как надежность, эффективность, применимость, сопровождаемость и переносимость. Такими свойствами должны обладать элементы задела продукта, чтобы быть принятыми участниками проекта. **2.** Каждое нефункциональное требование претендует на включение в критерий готовности. *См. также критерий готовности*.

Неявное знание. Неписаное и невысказанное знание, включая наблюдения, интуицию и чутье, которые трудно, но вполне возможно выразить на формальном языке. Противоположно явному или формальному знанию. Иногда обозначается как “ноу-хау”.

Норма практики. Способ поддержки или реализации определенного принципа на практике. Например, принцип демонстрирования достигнутого прогресса поддерживается нормой практики, принятой в Scrum для подведения итого спринта. *См. вид деятельности*, *артефакт*, *роль*, *правило*. *См. также принцип*, *ценности*.

Нормы инженерной практики. Конкретные нормы или приемы практики, применяемые в течение спринта, чтобы правильно выполнять работу для выпуска функциональных средств, которые имеют управляемые уровни технического долга и соответствуют критерию готовности, согласованному Scrum-командой.

Обследование и адаптация. 1. Типичная стадия Scrum-процесса, выполняемая по принципам обследования и адаптации управления эмпирическим процессом. **2.** Принцип обследования продукта или процесса и его адаптации на основании полученных знаний о нем. **3.** Главная часть петли обучения. *См. также адаптация*, *управление эмпирическим процессом*, *обследование*, *петля обучения*.

Обследование. Один из трех столпов управления эмпирическим процессом; подразумевает тщательное изучение и обработку ответной реакции с целью принять решение об адаптации продукта или процесса его разработки. *См. также управление эмпирическим процессом*, *адаптация*, *прозрачность*.

Объем партии. Количество элементов, которые должны быть обработаны на какой-то последующей стадии процесса. *См. также незавершенные работы*.

Обязательные функциональные средства. Ряд функциональных средств, которые должны непременно присутствовать в предстоящем выпуске, чтобы считать его жизнеспособным. *Синоним минимально выпускаемых функциональных средств*. *Ср. желательные функциональные средства*, *необязательные функциональные средства*.

Обязательство. Акт связывания себя с порядком действий. Принятие обязательства поощряется в Scrum. Обязательство означает, что каждый член команды посвящает себя достижению конечной цели, намеченной всей командой, как бы ни складывались обстоятельства. *Ср. прогноз*.

Ограничение по времени. Методика планирования времени, помогающая эффективно организовать работу и управлять объемом работ. *См. также временные рамки*.

Определенный процесс. Процесс с вполне определенными стадиями. При одних и тех же предпосылках в результате определенного процесса всякий раз должен быть получен один и тот же конечный результат во вполне определенных пределах отклонений. *Ср.* **управление эмпирическим процессом**.

Основание проекта. *См. составление положения о проекте*.

Оценивание. Приблизительное вычисление значения, числа, количества или меры чего-нибудь. Обычно в Scrum оценивается размер элементов из портфеля заказов или задела продукта, а также задач из задела спринта. *См. также прогноз*.

Очередь. Место для размещения элементов (запасов) в ожидании следующего действия по ходу выполнения работ. *См. также запасы, незавершенные работы.*

Очки за историю. Мера относительного размера элементов задела продукта, принимающая во внимание такие факторы, как сложность и физический размер. Как правило, оценки в очках за историю выставляются при проведении покера планирования. *См. также идеальный день, покер планирования, мера относительного размера.*

Очконадувательство. Предосудительное поведение, связанное с искусственным раздуванием величины оценок размеров отдельных элементов в заделе продукта при попытке согласовать или оптимизировать неразумно выраженную меру (например, для достижения намеренной скорости работы).

Петля обучения. Петля обратной связи, сосредоточенная на расширении процесса обучения. Этот процесс обычно проходит следующие стадии: выработка предположения (или постановка цели), построение чего-нибудь (выполнение некоторых видов работ), получение ответной реакции на то, что было построено, а затем использование этой ответной реакции для обследования того, что было сделано в сравнении с тем, что предполагалось сделать.

План выпуска. 1. Результат планирования выпуска. В плане выпуска с жестко заданными сроками указывается ряд функциональных средств, которые должны быть готовы к ясно намеченной будущей дате, а в плане выпуска с жестко заданным объемом работ — количество спринтов и затраты на выполнение четко определенного объема работ. **2.** План, сообщающий настолько точно, насколько это возможно, когда именно будет готов выпуск, какие функциональные средства войдут в его состав и во что это обойдется. *См. также выпуск с жестко заданными сроками, выпуск с жестко заданным объемом работ.*

Планирование выпуска. Долгосрочное планирование, дающее ответы на вопросы о сроках готовности выпуска, составе его функциональных средств или затратах на выпуск. Назначение планирования выпуска — найти золотую середину между коммерческой ценностью и общим качеством с учетом ограничений, накладываемых на объем работ, сроки их выполнения и бюджет. *См. также план выпуска.*

Планирование портфеля заказов. Мероприятие по определению тех продуктов (или проектов), над которыми предстоит работать, а также продолжительности проектных работ. Иногда еще *называется управлением портфелем заказов.*

Планирование продукта. *См. выработка общего замысла.*

Планирование спринта. Мероприятие, на котором собравшиеся члены Scrum-команды согласуют цель спринта и определяют ту часть задела продукта, которую требуется выпустить в течение планируемого спринта. При планировании спринта его задел формируется для того, чтобы помочь команде приобрести уверенность в том, что ей по силам выпустить элементы задела продукта,

которые она обязуется завершить в предстоящем спринте. *См. также задел спринта*, *цель спринта*.

Плановый процесс. Методика разработки, состоящая в том, чтобы попытаться запланировать и предусмотреть заранее все функциональные средства, которые могут потребоваться пользователю в конечном продукте, а также определить наилучший способ построения этих функциональных средств. Рабочий план составляется в результате выполнения последовательного ряда стадий с конкретными видами работ. *Синоним упреждающего процесса*, *прогноризующего процесса*, *предписывающего процесса*, *последовательного процесса*, *традиционного процесса разработки*, *водопадного процесса*.

Подведение итогов спринта. Мероприятие по обследованию и адаптации, проводимое по завершении выполнения каждого спринта, когда Scrum-команда демонстрирует всем заинтересованным сторонам свои достижения в течение обсуждаемого спринта. Подведение итогов спринта дает каждому его участнику возможность составить представление о состоянии проектных работ, обследовать то, что было сделано до сих пор, и адаптировать то, что предстоит сделать дальше. *См. также обследование и адаптация*, *демонстрация спринта*.

Поддающаяся спринту история. *См. реализуемая история*.

Подход. Конкретный метод или способ реализации нормы практики или вида деятельности, Например, в Scrum определяется ретроспектива спринта. Порядок ее проведения, выбираемый командой, представляет собой подход, который может отличаться от подходов других команд. *См. также вид деятельности*, *норма практики*.

Покер планирования. Методика достижения согласия при оценивании относительных размеров элементов в заделе продукта.

Пользовательская история. Удобная форма выражения желательной коммерческой ценности многих видов элементов из задела продукта. Пользовательские истории составляются таким образом, чтобы они были понятны как деловым людям, так и техническим работникам. Оно должны иметь простую структуру и обычно выражаются в следующей форме: “Мне как <роль пользователя> требуется достичь <цель>, чтобы извлечь <выгода>.” Пользовательские истории служат удобным заменителем обсуждения. Кроме того, они могут быть написаны с разной степенью детализации и должны легко поддаваться последующему уточнению. *См. также эпическая история*, *постепенное уточнение*, *тема*, *роль пользователя*.

Порог доверия. 1. Критерий готовности для выработки общего замысла (т.е. планирования продукта). **2.** Информация, которая требуется для принятия решения, чтобы иметь достаточную уверенность в принятии решения о целесообразности или нецелесообразности более детализированной разработки.

Поросята. Метафора, употребляемая в некоторых Scrum-командах, для обозначения тех людей, которые вносят свой вклад в достижение конечной цели,

намеченной Scrum-командой, на уровне обязательств, т.е. ответственности за конечный результат. Обычно к поросятам относят членов Scrum-команды. *См. также Scrum-команда. Ср. цыплята.*

Последний ответственный момент. Стратегия, состоящая в том, чтобы не принимать преждевременное решение, а отложить взятие на себя обязательств и оставить свободу выбора в принятии важных и необратимых решений до тех пор, пока не окажется, что непринятие решения обойдется дороже, чем его принятие.

Последовательный процесс. *См. плановый процесс.*

Постепенное уточнение. Своевременное разбиение крупных, слабо детализированных элементов задела продукта на ряд более мелких и детализированных элементов.

Постепенное финансирование. Финансирование некоторой части разработки продукции без обязательства финансировать все остальное. Применяя постепенное финансирование, можно профинансировать лишь первую небольшую часть проектных работ, а после важного утвержденного обучения в первой оплаченной части — пересмотреть решение о дальнейшем финансировании. *См. также порог доверия, утвержденное обучение.*

Постоянный темп. Выбранный должным образом темп работы команды для поддержания нормального потока доставки коммерческой ценности в течение продолжительного периода времени, но без истощения трудовых ресурсов.

Построение карты историй. 1. Методика составления пользовательских историй с точки зрения самих пользователей. Каждый вид пользовательской деятельности на высоком уровне разлагается на последовательность операций, которые могут быть далее разложены на ряд подробных задач. **2.** Двумерное представление традиционного одномерного списка из задела продукта. *См. также задел продукта, пользовательская история.*

Потеря новаторства. Потеря возможности разработать новаторское решение. Нередко происходит, когда в качестве элемента из задела продукта предлагается предписанное решение.

Поток единичных изделий. Состояние, в котором элементы производятся по очереди и проходят через процесс разработки отдельными единицами.

Поток. 1. Плавный, постоянный ход работ в течение всего процесса разработки, обеспечивающий доставку экономически пригодной ценности. **2.** Исключение простоев в работе экономически обоснованными способами. **3.** Противоположен крупной партии, крупному выпуску и выполнению работ “одним махом”.

Правило бойскаутов. 1. Всегда оставляй стоянку под лагерь чище, чем она была до твоего прихода. Если бойскауты обнаружат беспорядок на стоянке, они приберут ее, кто бы этот беспорядок ни оставил. **2.** Когда вы переходите к программированию, всегда оставляйте код чуть более ясным и упорядоченным, чем он был до вас. *См. также технический долг.*

Правило. Общая норма практики или надежный порядок действий в конкретной ситуации. Правило может быть нарушено, если возникает особая ситуация, требующая изменить порядок действий. Инфраструктура Scrum включает в себя определенные правила. *См. также Scrum*, *инфраструктура Scrum*.

Правильность. Обозначает, насколько оценка близка к фактическому значению — близость измеренной величины к ее истинному значению. Например, оценка вероятности поставки продукта в октябре 2015 года считается правильной, если данный продукт поставляется в любой день на протяжении октября 2015. *Ср. точность*.

Предписывающий процесс. *См. плановый процесс*.

Предположение. Догадка или мнение о том, что может быть истинно, реально или определенно, несмотря на отсутствие утвержденного обучения, позволяющего узнать, что это истинно. *Ср. утвержденное обучение*.

Препятствие. Помеха или преграда, не позволяющая сделать что-нибудь. Нередко служит для обозначения некоторого затруднения или осложнения, препятствующего команде или организации эффективно работать по методике Scrum.

Прибыль в течение срока эксплуатации. 1. Общая прибыльность продукта в течение срока его эксплуатации. **2.** На стадии планирования портфеля заказов это общая потенциальная прибыльность всего портфеля заказов, а не только одного продукта.

Приемочное тестирование. 1. Вид тестирования, выполняемого для проверки на соответствие критериям приемки. **2.** Вид тестирования, выполняемого для определения коммерческой ценности, которую должен доставлять каждый элемент задела продукта. Позволяет проверять достоверность функциональных или нефункциональных требований вроде производительности или надежность. Применяется в качестве вспомогательного средства для ведения разработки [Crispin, Lisa, and Janet Gregory. 2009]. **3.** Вид формального тестирования, выполняемого с учетом пользовательских нужд, требований и бизнес-процессов, чтобы определить, удовлетворяет ли проверяемая система критериям приемки, а также дать пользователям, заказчикам или другому официально уполномоченному органу возможность решить, следует ли принимать систему (Стандарт IEEE 610).

Принцип наименьшего удивления. Действие или разработка продукции таким образом, чтобы меньше всего поразить окружающих.

Принцип. Главная истина или убеждение, на основании которого выбирается подход к разработке продукции. Примером принципа в Scrum служит частое демонстрирование достигнутого прогресса. *См. также норма практики*, *ценности*.

Прирост потенциально готового к поставке продукта. Результаты, достигнутые с высокой степенью доверия и представляющие качественную работу, которая потенциально готова к поставке конечным потребителям по завершении спринта. Но готовность полученных результатов к поставке совсем не означает, что они

будут действительно доставлены потребителям. Поставка — это коммерческое решение, а потенциальная готовность к поставке — состояние доверия.

Прогноз. 1. Заявления, предсказания или оценки событий, конкретные последствия которых еще не наблюдались. **2.** Термин, употребляемый в документе "The Scrum Guide" (Руководство по Scrum) 2011 года для обозначения результатов, достигнутых командой разработчиков при планировании спринта. *См. также оценивание. Ср. обязательство.*

Прогноризующий процесс. *См. плановый процесс.*

Продукт. 1. Результат проектных работ. **2.** Товар или услуга, обладающие рядом осязаемых и неосязаемых свойств, удовлетворяющих потребителей и получаемых в результате обмена на деньги или другую единицу ценности. **3.** Как правило, долговечный, устойчивый артефакт, для создания которого организации могут вести один или несколько проектов. *См. также проектные работы. Ср. проект.*

Проект. 1. Временное мероприятие, проводимое для создания особой продукции, услуги или получения иного результата [PMI. 2008]. **2.** Вид деятельности, который завершается, чтобы конечные цели были достигнуты. В сравнении со сроком эксплуатации продукта проект заметно короче. Нередко несколько проектов выполняются одновременно в течение полного жизненного цикла продукта. *Ср. продукт.*

Проектирование, ориентированное на тестирование. Норма инженерной практики, предполагающая написание тестов перед разработкой. Пример разработки посредством тестирования. *См. также нормы инженерной практики, разработка посредством тестирования.*

Проектные работы. Полный объем работ, выполняемых для создания или усовершенствования продукции или услуги. *Ср. проект.*

Прозрачность. Один из трех столпов управления эмпирическим процессом; открытый доступ к объективной информации, требующейся для обследования и адаптации. *См. также адаптация, управление эмпирическим процессом, обследование.*

Простаивающая работа. Работа, которая активно не продвигается, а ждет своей очереди. *Ср. простаивающие работники.*

Простаивающие работники. Люди, имеющие возможность работать больше, поскольку они не заняты полностью. *Ср. простаивающая работа.*

Простая область. 1. Ситуация, когда всем ясно видны причины и следствия. Нередко правильный ответ на возникающий вопрос очевиден и неоспорим. **2.** Одна из областей в инфраструктуре Cynefin. *См. также Cynefin. Ср. сложная область, комплексная область, беспорядочная область, хаотичная область.*

Размеренный ритм. Постоянный, предсказуемый ритм или темп. Спринты с постоянной продолжительностью устанавливают размеренный ритм для выполнения проектных работ. *См. также синхронизация.*

Разработка посредством приемочного тестирования. Методика разработки, позволяющая участникам совместно обсудить сначала критерии приемки на конкретных примерах, а затем выделить их в ряд конкретных приемочных тестов, прежде чем приступать к разработке. *Синоним спецификации по образцу*.

Разработка посредством тестирования. 1. Эволюционный подход к разработке, основанный на написании непроходящих автоматизированных тестов до написания функционального кода, который вынудит их проходить. Как только будет написан код для прохождения теста, цикл повторяется, включая реорганизацию существующего кода с целью обеспечить согласованное межфункциональное проектирование. Цель разработки посредством тестирования — определение, а не проверка достоверности. Это означает, что сначала обдумывается проектное решение, а затем пишется чистый код, который всегда работает. **2.** Пример проектирования, ориентированного на тестирование. *См. также реорганизация кода*, *нормы инженерной практики*, *проектирование, ориентированное на тестирование*.

Расточительство. Любой вид деятельности, потребляющий ресурсы и не производящий никакой дополнительной ценности продукции или услуги, получаемой потребителем.

Реализуемая история. Пользовательская история достаточно малого размера, чтобы вписываться в спринт. *Синоним поддающаяся спринту история*.

Резкая смена стратегии. 1. Смена направления, но с учетом приобретенных знаний. **2.** Структурированная коррекция курса с целью проверить новую основополагающую гипотезу о продукте, стратегии и движущей силе роста [Ries, Eric. 2011].

Реорганизация кода. Методика перестройки существующего тела кода путем улучшения или упрощения его внутренней структуры, но без изменения его внешнего поведения. Реорганизация кода — одна из основных методик управления техническим долгом. *См. также технический долг*, *нормы инженерной практики*.

Ретроспектива спринта. Мероприятие по обследованию и адаптации, проводимое в конце каждого спринта. Ретроспектива спринта предоставляет Scrum-команде постоянную возможность анализировать процесс разработки (подходы к ее проведению по методике Scrum) и выявлять пути совершенствования данного процесса. *См. также обследование и адаптация*, *подведение итогов спринта*.

Ретроспектива. *См. ретроспектива спринта*.

Решение. Продукция или услуга, получаемая в результате проектных работ.

Риск. 1. Вероятность того, что событие будет иметь нежелательные последствия. Риск измеряется вероятностью наступления события и серьезностью его последствий. **2.** Любая неопределенность в том, что определенный вид деятельности может иметь отрицательный результат. *См. также неопределенность*.

Роение. Такое поведение, когда члены команды с имеющимися возможностями и подходящими навыками собираются вместе (т.е. роятся) для работы

над начатым раньше элементом, чтобы завершить его и перейти к работе над следующими элементами. *См. также Т-образные навыки.*

Роль пользователя. 1. Наименование категории пользователей разрабатываемой продукции. **2.** Один из главных элементов пользовательской истории, определяющий получателя ценности, доставляемой пользовательской историей. *См. также пользовательская история.*

Роль. Согласованный ряд обязанностей, которые могут выполняться одним человеком или несколькими людьми. К трем ролям в Scrum относится владелец продукта, Scrum-мастер и команда разработчиков. *См. также норма практики, принцип.*

Самая коротковзвешенная работа. Экономически оптимальный алгоритм календарного планирования в среде, где стоимость задержки и продолжительность разных видов работ отличается. *См. также стоимость задержки.*

Самоорганизация. 1. Свойство организации сложной адаптивной системы по восходящей, возникающее со временем в качестве реакции на ее окружение. **2.** Способность команды разработчиков организоваться со временем без применения по нисходящей внешней господствующей силы, характерной для традиционного командно-административного управления. **3.** Отражение принципа управления, при котором принятие большинства оперативных решений поручается тем, кто лучше всего знает их последствия и практические результаты. *См. также сложная адаптивная система, возникновение.*

Своевременный или оперативный. Характеристика процесса, которая обозначает, что в ходе работ ресурсы или мероприятия становятся доступными или проводятся, как только они требуются.

Сейсмограмма эмоций. Графическое представление эмоциональных подъемом и падений, проявляющихся у членов команды по ходу спринта. Эта методика нередко применяется при проведении ретроспективы спринта. *См. также ретроспектива спринта.*

Синхронизация. Обусловливает наступление сразу нескольких событий. Нередко применяется для координирования действий многих работающих вместе Scrum-команд таким образом, чтобы они начинали и завершали спринты в одни и те же дни. *См. также размеренный ритм.*

Скорая ответная реакция. Принцип, устанавливающий, что ответная реакция сегодня намного ценнее, чем та же самая ответная реакция завтра, поскольку с помощью нынешней ответной реакции можно устранить затруднение, прежде чем оно превратится в еще большее затруднение. Это позволяет раньше отсечь экономически нецелесообразные пути следования (т.е. произвести быстрый провал). *См. также быстрый провал.*

Скорость работы. Мера быстроты, с которой работа выполняется в единицу времени. Скорость работы в Scrum, как правило, измеряется в виде суммы

оценок размеров элементов из задела продукта, завершенных в течение спринта. Скорость работы указывается в тех же самых единица измерения, что и для элементов задела продукта, как правило, в очках за историю или идеальных днях. Кроме того, скорость работы позволяет измерить итог (т.е. объем того, что выпускается), а не конечный результат (т.е. доставляемую ценность).

Сложная адаптивная система. Система, состоящая из многих элементов, по-разному взаимодействующих друг с другом, причем эти взаимодействия осуществляются по простым, локализованным правилам, действующим в контексте постоянной ответной реакции. К примерам таких систем относятся фондовая биржа, мозг, колонии муравьев и Scrum-команды.

Сложная область. 1. Ситуация, в которой может быть несколько правильных ответов, но для их выбора требуется квалифицированная диагностика. **2.** Одна из областей в инфраструктуре Cynefin. *См. также Cynefin. Ср. хаотичная область, комплексная область, беспорядочная область, простая область.*

Случайно обнаруженный технический долг. Категория состояния технического долга, обозначающая такой технический долг, о существовании которого команда разработчиков не подозревала до тех пор, пока он не проявился при нормальном выполнении работы над продуктом. *Ср. известный технический долг, намеченный технический долг. См. также технический долг.*

Сначала одно, а затем другое. Характеристика процесса последовательной разработки, когда разрабатываемый продукт переносится из предыдущей стадии в следующую по критерию полного на 100% объема партии. *См. также объем партии.*

Совещание по написанию пользовательских историй. Мероприятие, которое длится от нескольких часов до нескольких дней, когда разные участники проекта совместно формулируют желательную коммерческую ценность и создают заменители того, что должна делать продукция или услуга, в виде пользовательских историй. *См. также пользовательская история.*

Составление положения о проекте. Ряд предварительных работ для определения проекта на таком уровне детализации, которого достаточно для принятия решения о его финансировании. *Синоним зарождения проекта, основания проекта.*

Спецификация по образцу. *См. разработка посредством приемочного тестирования.*

Спринт. Краткосрочная, ограниченная по времени итерация. Как правило, временны*е* рамки спринта находятся в пределах от одной недели до одного календарного месяца, и в течение этого периода времени Scrum-команда сосредоточена на получении прироста потенциально готового к поставке продукта, соответствующего согласованному этой Scrum-командой критерию готовности. *См. также критерий готовности, итерация, прирост потенциально готового к поставке продукта.*

Стоимость задержки. Финансовые расходы, связанные с задержкой работ или достижением какого-то промежуточного этапа. Стоимость задержки подкрепляет тот принцип, что время обусловливает реальные финансовые расходы. А ведь для того, чтобы пойти на какой-то экономически обоснованный компромисс, очень важно знать эти расходы.

Стратегический технический долг. Форма технического долга, употребляемая как средство, помогающее организациям количественно оценить и эффективно использовать экономические показатели для принятия важных и зачастую своевременных решений. Иногда технический долг берется из стратегических соображений как вполне обоснованный коммерческий выбор. *Ср. наивный технический долг*, *неизбежный технический долг*. *См. также технический долг*.

Стратегический фильтр. Критерии принятия решений, применяемые в организации для оценивания соответствия предлагаемого продукта стратегическим критериям с целью перейти к дополнительному рассмотрению этого продукта. *Ср. экономический фильтр*.

Сущность Scrum. Ценности, принципы и нормы практики из инфраструктуры Scrum в сочетании с правилами и опробованными подходами и методами применения норм практики в Scrum. *См. также подход*, *норма практики*, *правило*, *инфраструктура Scrum*.

Схватка над схватками. Метод координирования работы многих Scrum-команд, когда члены каждой из Scrum-команд собираются вместе для обсуждения и разрешения зависимостей, возникающих между командами. *См. также выпускной поезд*.

Схваткопад. *См. водопадная схватка*.

Тема. Собрание связанных вместе историй. Тема предоставляет удобный способ указать на нечто общее у ряда историй, например, их принадлежность к одной и той же функциональной области. *См. также эпическая история*, *пользовательская история*.

Технические истории. "Пользовательская" история (элемент задела продукта), не доставляющая конечному пользователю никакой ощутимой ценности, но в то же время удовлетворяющая важные архитектурные или инфраструктурные потребности для последующей доставки ценности конечному пользователю. *См. также пользовательская история*.

Технический долг. 1. Термин, употребляемый для описания обязательства, которое берет на себя организация, разрабатывающая программное обеспечение, когда она выбирает проектное или конструктивное решение, которое оказывается выгодным в краткосрочной перспективе, но увеличивает сложность и более затратное в долгосрочной перспективе. **2.** Метафора, упрощающая общение деловых людей с техническими работниками по поводу несоответствий в реализации артефактов. *См. также наивный технический долг*, *стратегический наивный технический долг*, *неизбежный технический долг*.

Точечное голосование. Методика, позволяющая участникам голосовать размещением цветных точечных меток на карточках тех элементов задела продукта, которые они считают наиболее приоритетными. В итоге элементы с наибольшим количеством точечных меток на карточках получают более высокий приоритет. Такая методика нередко применяется при проведении ретроспективы спринта. *См. также ретроспектива спринта.*

Точность. Обозначает точность оценки. Например, сказать, что продукт будет поставлен 7 октября 2015 года, будет точнее, чем сказать, что он будет поставлен в октябре 2015 года. *Ср. правильность.*

Традиционный процесс разработки. *См. плановый процесс.*

Упорядочение задела продукта. Мероприятия по составлению и уточнению, оцениванию и расстановке элементов задела продукта по приоритетам.

Упорядочение. *См. упорядочение задела продукта.*

Управление эмпирическим процессом. Стиль работы, при котором эффективно используются принципы обследования, адаптации и прозрачности. *Ср. определенный процесс.*

Упреждающий процесс. *См. плановый процесс.*

Условия удовлетворения. Условия, при которых владелец продукта будет удовлетворен завершением работы над элементом задела продукта. Условия удовлетворения являются критериями приемки, уточняющими желательное поведение. *См. также критерии приемки.*

Утвержденное обучение. Термин, предложенный Эриком Ризом [Ries, Eric. 2011], для описания прогресса, достигнутого в подтверждении или опровержении важных предположений, каждое из которых было подвергнуто одной или несколькими проверками на соответствие требованиям заказчика. *Ср. предположения.*

Участник проекта. Отдельное лицо, группа лиц или организация, оказывающая влияние или находящаяся под влиянием действий, совершаемых организацией, выполняющей проект. *См. также внешние участники проекта, внутренние участники проекта.*

Учет нововведений. Система измерения и учета, в которой действенные количественные показатели применяются для оценивания скорости обучения как важной меры продвижения к коммерчески достижимому результату [Ries, Eric. 2011].

Функциональное средство. 1. Часть функциональных возможностей коммерческого продукта, представляющих ценность для заказчика или пользователя. **2.** Служит иногда для обозначения пользовательской истории среднего размера, которую можно и должно разделить на ряд более мелких историй, совместно реализуемых для доставки ценности данного функционального средства. *См. также тема, пользовательская история.*

Хаотичная область. 1. Ситуация, требующая быстрой ответной реакции. Критическое состояние, требующее немедленных действий, чтобы предотвратить дальнейший ущерб или хотя бы восстановить порядок. **2.** Одна из областей в инфраструктуре Cynefin. *См. также Cynefin. Ср. сложная область, комплексная область, беспорядочная область, простая область.*

Цель выпуска. Ясно сформулированное назначение и желательный результат выпуска. Цель выпуска намечается с учетом самых разных факторов, включая целевых потребителей, архитектурные вопросы, решаемые на высоком уровне, а также важные события, происходящие на рынке. *См. также выпуск.*

Цель спринта. Конечная цель, которая ставится на высоком уровне владельцем продукта и которой ему хотелось быть достичь в течение спринта. Нередко ставится в виде конкретного ряда элементов из задела продукта, которые требуется реализовать в течение предстоящего спринта.

Ценности. 1. Те вещи, которые считаются дорогими или заветными. **2.** Основание для общего соглашения о принципах деятельности, к которому приходят члены команды разработчиков. К основным ценностям Scrum относятся честность, открытость, смелость, уважение, собранность, доверие, расширение прав и полномочий, сотрудничество.

Ценность для участника проекта. Ценность, которую полученное решение доставляет участникам проекта. Иногда употребляется попеременно с термином *коммерческая ценность. См. также участник проекта.*

Церемония. Ритуальное или символическое мероприятие, проводимое во вполне определенных случаях. Некоторые относят к церемониям такие основные виды деятельности или мероприятия в Scrum, как планирование спринта, ежедневная летучка, подведение итогов и ретроспектива спринта. *См. также вид деятельности, излишняя формальность.*

Цыплята. Метафора, употребляемая в некоторых Scrum-командах для обозначения тех людей, которые вносят свой вклад в достижение конечной цели, намеченной Scrum-командой, но делают это лишь на уровне формального участия, а не обязательств. Лучше всего употреблять для обозначения людей, не входящих в состав Scrum-команды. Заимствован из следующей старой шутки по поводу цыплят и поросят: “В завтраке с яичницей и ветчиной цыпленок только участвует, а поросенок втянут в него полностью”. *Ср. поросята.*

Экономический фильтр. Критерии принятия решений, применяемые в организации для оценивания экономических показателей предлагаемого продукта, чтобы решить, стоит ли финансировать его разработку. *Ср. стратегический фильтр.*

Экстремальное программирование. Методика гибкой разработки, дополняющая методику Scrum. В экстремальном программировании определяются важные нормы инженерной практики, которыми команды разработчиков могут

пользоваться для управления ходом работ на уровне отдельных задач в течение спринта. *См. также гибкий.*

Элемент задела продукта. 1. Элемент вроде функционального средства, дефекта, а иногда и технического долга, который представляет определенную ценность для владельца продукта. **2.** Элемент в заделе продукта. *См. также задел продукта.*

Эпическая история. Крупная пользовательская история, имеющая размер от нескольких до многих месяцев и способная охватить один или несколько выпусков. Эпические истории служат удобными заменителями крупных требований. Они постепенно уточняются и разбиваются в подходящий момент на ряд более мелких пользовательских историй. *См. также функциональное средство, постепенное уточнение, тема, пользовательская история.*

Приложение Б
Библиография

Adkins, Lyssa. 2010. Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition. Addison-Wesley Professional.

Anderson, David J. 2010. *Kanban*. Blue Hole Press.

Appelo, Jurgen. 2011. Management 3.0: Leading Agile Developers, Developing Agile Leaders. Addison-Wesley Professional.

Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. 2001. *Manifesto for Agile Software Development*. `www.agilemanifesto.org/`.

Beck, Kent, and Cynthia Andres. 2004. *Extreme Programming Explained*, 2nd ed. Addison-Wesley Professional.

Boehm, Barry W. 1981. *Software Engineering Economics*. Prentice Hall.

Brooks, Frederick P. 1995. *The Mythical Man-Month: Essays on Software Engineering*, 2nd ed. Addison-Wesley Professional. (Originally published in 1975.)

Cohn, Mike. 2004. *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional.

Cohn, Mike. 2006. *Agile Estimating and Planning*. Addison-Wesley Professional.

Cohn, Mike. 2009. *Succeeding with Agile. Addison-Wesley Professional*.

Cohn, Mike. 2010. Основная презентация на Agile 2010.

Cook, Daniel. 2008. "The Laws of Productivity: 8 productivity experiments you don't need to repeat." Презентация, доступная по адресу `http://www.lostgarden.com/2008/09/rules-of-productivity-presentation.html`.

Crispin, Lisa, and Janet Gregory. 2009. *Agile Testing: A Practical Guide for Testers and Agile Teams*. Addison-Wesley Professional.

Cunningham, Ward. 1992. "The WyCash Portfolio Management System," OOPSLA 1992 experience report. OOPSLA '92, Object-Oriented Programming Systems, Languages and Applications, Vancouver, BC, Canada, October 18–22.

Denne, Mark, and Jane Cleland-Huang. 2003. *Software by Numbers: Low-Risk, High-Return Development*. Prentice Hall.

Derby, Esther, and Diana Larsen. 2006. *Agile Retrospectives: Making Good Teams Great*. Pragmatic Bookshelf.

Fowler, Martin. 2009. "Technical Debt Quadrant." Статья в Блики (блоке, совмещенном с Википедией), доступная по адресу http://martinfowler.com/bliki/TechnicalDebtQuadrant.html.

Fowler, Martin, Kent Beck, John Brant, William Opdyke, and Don Roberts. 1999. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional.

Goldberg, Adele, and Kenneth S. Rubin. 1995. *Succeeding with Objects: Decision Frameworks for Project Management*. Addison-Wesley Professional.

Grenning, James. 2002. "Planning Poker." www.objectmentor.com/resources/articles/PlanningPoker.zip.

Highsmith, Jim. 2009. *Agile Project Management: Creating Innovative Products*, 2nd ed. Addison-Wesley Professional.

Hohmann, Luke. 2003. *Beyond Software Architecture*. Addison-Wesley Professional.

IEEE. 1990. IEEE Std 610.12-1990 (revision and designation of IEEE Std 792-1983). IEEE Standards Board of the Institute of Electrical and Electronics Engineers, New York, September 28, 1990.

Jeffries, Ron. 2001. "Essential XP: Card, Conversation, Confirmation." http://xprogramming.com/articles/expcardconversationconfirmation/.

Katz, Ralph. 1982. "The Effects of Group Longevity on Project Communication and Performance." *Administrative Science Quarterly 27:* 81–104.

Kennedy, John Fitzgerald. 1961. Special Message to the Congress on Urgent National Needs, May 22.

Kerth, Norm. 2001. *Project Retrospectives: A Handbook for Team Reviews*. Dorset House.

Larman, Craig, and Bas Vodde. 2009. "Lean Primer." Документ, загружаемый по адресу www.leanprimer.com/downloads/lean_primer.pdf.

Laufer, Alexander. 1996. *Simultaneous Management: Managing Projects in a Dynamic Environment*. American Management Association.

Leffingwell, Dean. 2011. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley Professional.

McConnell, Steve. 2007. "Technical Debt." Статья из блога, доступная по адресу http://blogs.construx.com/blogs/stevemcc/archive/2007/11/01/technical-debt-2.aspx.

Mar, Kane. 2006. "Technical Debt and the Death of Design: Part 1." Статья из блога, доступная по адресу http://kanemar.com/2006/07/23/technical-debt-and-the-death-of-design-part-1/.

Martin, Robert C. 2008. Clean Code: *A Handbook of Agile Software Craftsmanship*. Prentice Hall.

Page, Scott. 2007. *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies*. Princeton University Press.

Patton, Jeff. 2008. Example of incremental releasing. Personal communication.

Patton, Jeff. 2009. "Telling Better User Stories: Mapping the Path to Success." *Better Software*, November/December, 24–29.

Pelrine, Joseph. 2011. "Is Software Development Complex." Статья в гостевом блоге, доступная по адресу http://cognitive-edge.com/blog/entry/4597/is-software-development-complex.

Pichler, Roman. 2010. *Agile Product Management with Scrum: Creating Products That Customers Love*. Addison-Wesley Professional.

PMI. 2008. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 4th ed. Project Management Institute, Inc.

Poppendieck, Mary, and Tom Poppendieck. 2003. *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional.

Putnam, Doug. 1996. "Team Size Can Be the Key to a Successful Project." Статья из серии *QSM's Process Improvement Series*. www.qsm.com/process_01.html.

Putnam, Lawrence H., and Ware Myers. 1998. "Familiar Metrics Management: Small Is Beautiful—Once Again." *IT Metrics Strategies IV:8:* 12–16. Cutter Information Corp.

Reinertsen, Donald G. 2009a. "Types of Processes." Статья в гостевом блоге, доступная по адресу www.netobjectives.com/blogs/Types-of-Processes.

Reinertsen, Donald G. 2009b. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing.

Ries, Eric. 2011. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business.

Schwaber, Ken. 1995. "Scrum Development Process." *Из семинара OOPSLA Business Object Design and Implementation Workshop*, ed. J. Sutherland et al. Springer.

Schwaber, Ken. 2004. *Agile Software Development with Scrum*. Microsoft Press.

Schwaber, Ken, and Mike Beedle. 2001. *Agile Software Development with Scrum*. Prentice Hall.

Schwaber, Ken, and Jeff Sutherland. 2011. "The Scrum Guide." Документ, загружаемый по адресу www.scrum.org.

SEI. 2010. "CMMI for Development, Version 1.3." Software Engineering Institute, Carnegie Mellon University. Документ, загружаемый по адресу www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm.

SEI. 2011. Second International Workshop on Managing Technical Debt, May 23. Colocated with ICSE 2011, Waikiki, Honolulu, Hawaii. Документ, загружаемый по адресу www.sei.cmu.edu/community/td2011/.

Smith, Preston G., and Donald G. Reinertsen. 1998. *Developing Products in Half the Time: New Rules, New Tools*. Van Nostrand Reinhold.

Snowden, David J., and Mary E. Boone. 2007. "A Leader's Framework for Decision Making." *Harvard Business Review*, November.

Staats, Bradley R. 2011. *Unpacking Team Familiarity: The Effects of Geographic Location and Hierarchical Role*. University of North Carolina at Chapel Hill.

Takeuchi, Hirotaka, and Ikujiro Nonaka. 1986. "The New New Product Development Game." *Harvard Business Review*, January, 137–146.

Tuckman, Bruce W. 1965. "Developmental Sequence in Small Groups." *Psychological Bulletin* 63: 384–399. Статья, перепечатанная из *издания Group Facilitation: A Research and Applications Journal*, no. 3, Spring 2001.

VersionOne. 2011. "The State of Agile Development: Sixth Annual Survey." Статья, опубликованная в виде документа формата PDF, загружаемого из библиотеки Library of White Papers по адресу `www.versionone.com`.

Wake, William C. 2003. "INVEST in Good Stories, and SMART Tasks." `www.xp123.com`.

Wheelwright, Steven C., and Kim B. Clark. 1992. *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality.* The Free Press.

Wiseman, John "Lofty." 2010. *SAS Survival Guide: For Any Climate, in Any Situation*, rev. ed. Collins Reference.

Предметный указатель

S

Scrum-команды
  координирование работы
    выпускной поезд, метод, 298
    схватка над схватками, метод, 294
  структурирование, 287
Scrum-мастер
  выполнение роли, 263
  кандидаты на роль, 262
  обязанности, 55; 255
  ответственность, 257
  полномочия, 56; 255
  сочетание с другими ролями, 264
  типичный график работы, 261
  характеристики и навыки, 258

Б

Быстрый провал, стратегия, 150; 391

В

Владелец продукта
  главный, роль, 253
  заместитель, назначение, 252
  кандидаты на роль, 244
  обязанности, 55; 232
  полномочия, 55
  роли, 232
  типичный график работы, 241
  характеристики и навыки, 238
Внутрипроцессные продукты
  анализ, 346
  стратегии управления, 365
Водопадная схватка, 77
Возможности команды
  выяснение, 432
  оценивание
    в очках за историю, 433
    в человеко-часах, 433
  факторы влияния, 431
Возникающие возможности
  быстрый охват, 359
  экономическая ценность, 359
Выпуски
  ограничения
    коррекция, 402
    наложение, 397
  периодические, фиксированные, стратегия, 379
  размеренный темп, определение, 393
  расчет затрат, 416
  с жестко заданными сроками
    планирование, 408
    сообщение о достигнутом прогрессе, 420
  с жестко заданным объемом работ
    планирование, 414
    сообщение о достигнутом прогрессе, 418
  цель, постановка, 380
Выпускной поезд
  координирование работы команд, 298
  метод, 296
  порядок проведения, 298
  правила, 296
Выработка общего замысла
  временные рамки, 370
  концепция продукта, 337
  назначение, 337
  определение, 369
  основная цель, 369
  первоначальная, 370
  по методике Scrum, особенности процесса, 385
  предпосылки, 372
  пример процесса, 373
  прохождение первоначальной идеи через стратегический фильтр, 370
  составление концепции продукта, 374

участники, 371
экономически обоснованная, рекомендации, 385

Г

График выпуска продукта
временные рамки, 381
составление, 381

Д

Диаграммы
выгорания
выпуска
с жестко заданными сроками, 419
с переменным объемом работ, 420
специальная форма, 421
спринтов, 454
сгорания
выпуска с жестко заданным объемом работ, 418
спринтов, 452
Доска задач
назначение, 450
Достигнутый прогресс
определение, 100
основные принципы, 100
сообщение, способы, 418; 450

Е

Ежедневные летучки
назначение, 59; 66; 448
определение, 65
порядок проведения, 66; 448

З

Задачи
выполнение, 141
исполнение, нормы практики, 448
назначение, 141
оцениваемые, разбиение, 429; 436
Задел
наблюдений
назначение, 484
формирование, 489
портфеля заказов
ввод и расположение элементов по порядку, 358
извлечение продуктов, порядок, 362
порядок расположения продуктов, 346
уравновешивание притока и оттока продуктов, 357
элементы, разновидности, 345
продукта
ввод элементов, 41
определение, 40; 155
отбор элементов, 435
представление элементов, 133
размеры элементов, оценивание, 61
расстановка элементов по приоритетам, 61
содержимое, 59
создание на высоком уровне, 378
упорядочение, 60; 161; 402
характеристики по критериям DEEP, 157
элемента, определение, 155
спринта
назначение, 58
содержимое, 428
составление из задач, 63
формирование, 425; 436
Запасы, оптимальное управление, 96

И

Иерархические заделы продуктов, создание, 172
Изготовление продукции, главная цель, 73
Изменчивость и неопределенность, основные принципы, 73
Инфраструктура
Cynefin
беспорядочная область, 48
комплексная область, 47
назначение, 45
области, разновидности, 46
сложная область, 47-48
хаотическая область, 48
Scrum
артефакты, 60
виды деятельности, 57
задел продукта, 155
нормы практики, 54
определение, 29
подведение итогов спринта, 458
применение, 29
ретроспектива спринта, 472
спринты, 109

ценности, 53
от компании Pragmatic Marketing, назначение, 246
Исполнение
ожидаемые характеристики, 102
основные принципы, 102
Исследования
для приобретения знаний, 148
назначение, 83
проведение, 84
Истории
поддающиеся спринту, определение, 141
пользовательские
иерархия абстракций, 140
карточки, применение, 136
критерии оценки INVEST, 142
назначение, 135
независимость, 142
обсуждаемость, 143
обсуждение, проведение, 137
оцениваемость, 146
подтверждение, критерии, 138
подходы к написанию, 152
подходящие размеры, 146
собирание, способы, 151
совещания по написанию, назначение и проведение, 151
тестируемость, 147
условия удовлетворения, 138
ценность, 144
построение карт, особенности, 153
приобретения знаний, назначение, 149
реализуемые, определение, 141
технические
определение, 144
ценность и недостатки, 145
эпические, назначение, 141
Итерации
определение, 40
планирование, 40

К

Карта спринтов
назначение, 397
составление, 405
Команды
владельцев продукта, назначение, 251
для компонентов
назначение, 171
определение, 288
организация работы, логика, 290
роли сеятелей и жнецов, 292
для функциональных средств
назначение, 250
определение, 288
организация работы, логика, 291
разработчиков
T-образные навыки, 274
долговечность, 284
как денежная единица гибкой разработки, 285
как единицы производительности в Scrum, 364
межфункциональная неоднородность, 273
межфункциональные, назначение, 40
мушкетерские отношения, 276
обмен информацией с высокой пропускной способностью, 278
обязанности, 55; 268
определение, 267
оптимальный штат, 280
отличие от групп, 284
полномочия, 56; 267
работа в постоянном темпе, 283
самоорганизация, 273
с конкретными ролями, целесообразность сохранения, 268
состав, 56
характеристики и навыки, 270
целеустремленность и обязательность, 281
Критерии
INVEST для оценки историй, 142
готовности
включение нефункциональных требований, 148
возможное развитие, 127
готовность полная и неполная, 129
определение, 125
строгие, употребление, 214
условия удовлетворения, 128
подготовленности
назначение, 165
установка, 166
приемки
определение, 129
условия удовлетворения, 129

М

Меры относительного размера
идеальные дни, 61
назначение, 61
очки за историю, 61
Методика
Kanban
преимущества, 49
применение доски задач, 450
рекомендации, 49
Scrum
главные препятствия к внедрению, 301
истоки, 41
как форма непрерывного совершенствования, 495
обнаружение собственного пути внедрения, 496
общие нормы передовой практики, соблюдение, 497
определение, 39
основные принципы гибкой разработки, категории, 73
поросята и цыплята, обязанности, 66
преимущества, 45
пригодность, 46; 49
принципы планирования, 325
причины для внедрения, 43
происхождение термина, 41
размеренный ритм, преимущества, 116
роли, разновидности, 54
сравнение с традиционными методиками разработки, 71
водопадная
определение, 71
плановые процессы, назначение, 72
плановая
недостатки, 72
пригодность, 72
Минимально выпускаемые функциональные средства
определение, 379
первоначально планируемый набор, 397
уточнение состава, 404
Минимально жизнеспособный продукт, определение, 379
Минимально пригодные для продажи функциональные средства, определение, 379

Н

Наименьшее удивление, принцип, 279
Незавершенные работы
определение, 93
основные принципы, 93
установление предела, 363
эффективный учет, 95
Неопределенность
основные категории, 79
устранение, способы, 80
Нормы практики
общие, соблюдение, 497
определение, 497

О

Обязательства
на спринт, порядок взятия, 426
обозначение, 58
окончательное взятие, 438
реалистичные, выработка, 428
Ограничение по времени
временные рамки, 110
определение, 110
спринтов, преимущества, 110
Ответная реакция, быстрая, достижение, 91
Оценивание
в идеальных днях, 188
в очках за историю, 187
задач, порядок, 181
затрат по размерам футболок, преимущества, 354
коллективное, принцип, 182
независимость от обязательств, принцип, 183
относительных размеров, принцип, 184
покер планирования, методика, 189
правильность, а не точность, принцип, 184
элементов из задела
портфеля заказов, порядок, 180
продукта
порядок, 180
принципы, 181
Очконадувательство, поведение, 199

П

Планирование
более мелких и частых выпусков, преимущества, 332; 360

в Scrum
  иерархия уровней, 344
  многоуровневое, 335
  основные принципы, 325
выполнения спринта
  назначение, 441
  на уровне отдельных задач, 441
  составление плана достижения цели спринта, 441
выпуска
  временные рамки, 341; 394
  назначение, 340; 394
  первоначальное, проведение, 394
  план выпуска, назначение, 397
  предпосылки, 395
  процесс, 395
  расчет затрат, 417
  с жестко заданными сроками, 408
  с жестко заданным объемом работ, 415
  синонимы, 394
  сообщение о достигнутом прогрессе, 418
  составление плана выпуска, 340
  упорядочение задела продукта, 402
  уточнение состава минимально выпускаемых функциональных средств, 404
  участники, 395
ежедневное, проведение ежедневных летучек, 343
подведения итогов спринта, календарное, 461
портфеля заказов
  внутрипроцессные продукты, стратегии управления, 365
  временные рамки, 346
  календарное планирование, стратегии, 349
  назначение, 337
  определение, 345
  отток продуктов, стратегии управления, 361
  предпосылки, 340; 346
  приток продуктов, стратегии управления, 355
  процесс, 347
  стратегии, категории, 348
  участники, 346
предварительное и оперативное, золотая середина, 326
продукта
  временные рамки, 370
  график выпуска продукта, составление, 338; 379
  концепция продукта, 337
  назначение, 337
  определение, 369
  предпосылки, 372
  процесс, 371
  составление концепции продукта, 375
  участники, 371
  формирование задела продукта на верхнем уровне, 338
спринта
  временные рамки, 425
  выполнение, 425
  двухэтапное , 429
  назначение, 341
  одноэтапное, 430
  окончательное взятие обязательств, 438
  определение возможностей команды, 431
  отбор элементов из задела продукта, 435
  предпосылки, 426
  приобретение уверенности, 436
  процесс, 426
  уточнение цели спринта, 438
  участники, 426
  формирование задела спринта, 342
убытки от переделывания перспективных планов, 331
Подведение итогов спринта
  адаптация, 467
  время проведения, 457
  демонстрация результатов, 466
  метод проведения, 464
  назначение, 68; 458
  обсуждение результатов, 68; 467
  общие вопросы, разрешение, 468
  подготовительная работа, 460
  подготовка к демонстрации, 463
  предпосылки и результаты, 464
  процесс, 464
  распределение обязанностей, 464
  участники, 458
Подход, определение, 497
Покер планирования
  определение, 189
  оценочная шкала, 189

правила игры, 192
преимущества, 193
принципы, 189
участники, 190
Порог доверия
назначение, 386
реалистичный, установка, 387
установка, 372
Последний ответственный момент, принцип, 81
Постоянный темп, определение, 62
Построение карт историй, методика, 152
Потеря новаторства, последствия, 144
Поток единичных изделий, подход, 94
Предположения
определение, 89
проверка достоверности, 89
Прибыли в течение срока эксплуатации
контекст, 349
определение, 100
Прирост потенциально готового к поставке продукта, определение, 67
Прогнозирование и адаптация, основные принципы, 80
Прогноз, обозначение, 58
Продукт, определение, 171
Простаивающая работа, опеределение, 96
Простаивающие работники, о пределение, 96
Процессы
плановые и Scrum, сравнение, 78
эмпирические, управление и принципы, 78

Р

Разработка
инкрементная
назначение, 76
преимущества и недостатки, 76
итеративная
назначение, 75
преимущества и недостатки, 76
непрерывная, норма практики, 394
партиями
единичными, особенности, 94
мелкими, преимущества, 94
посредством приемочного тестирования, принцип, 139
посредством тестирования, методика, 210
продукции
главная цель, 75
единовременная, особенности, 41
по традиционной методике, 71
Резкая смена стратегии, определение, 334
Реорганизация кода
как норма практики, 213
преимущества, 214
Ретроспектива спринта
временная шкала событий, построение и анализ, 482
время проведения, 472
выбор средоточия, 475
завершение, 489
метод проведения, 478
наблюдения
выявление, 483
задел, назначение, 484
мозговой штурм и фиксация, 484
негласное группирование, 484
определение, 486
расстановка по приоритетам методом точечного голосования, 486
назначение, 69; 471
обсуждение, 69
общие вопросы, разрешение, 491
подготовительная работа, 475
предпосылки и результаты, 478
процесс, 478
сбор объективных данных, 477
сейсмограмма эмоций, построение, 483
структура, 477
типичные упражнения, 476
улучшающие действия
выбор, 488
доведение до логического конца, 490
назначение, 486
участники, 473
Роение
назначение, 444
проведение, 444
Роли
Scrum-мастер
выполнение роли, 264
защита от вмешательства, 257
инициирование перемен, 258
как лидер-служитель, 257

кандидаты на роль, 262
наставничество, 255
ответственность за процесс разработки, 257
полномочия, 255
сочетание с другими ролями, 264
типичный график работы, 261
устранение препятствий, 257
характеристики и навыки, 258
владелец продукта
ведение экономического анализа, 233
главный, обязанности, 253
заместитель, назначение, 252
иерархическое расширение, 253
кандидаты на роль, 244
определение и проверка соответствия критериев приемки, 235
полномочия, 231
при внутренней разработке, 244
при коммерческой разработке, 245
при разработке компонентов, 249
при субподрядных работах, 248
сотрудничество
с командой разработчиков, 236
с участниками проекта, 238
сочетание с другими ролями, 250
типичный график работы, 241
упорядочение задела продукта, 235
участие в планировании, 235
характеристики и навыки, 238
команда разработчиков
Т-образные навыки, 274
выполнение спринта, 268
долговечность, 284
межфункциональная неоднородность, 273
мушкетерские отношения, 277
обмен информацией с высокой пропускной способностью, 278
обследование и адаптация, 269; 270
оптимальный штат, 280
открытость в общении, 279
планирование спринта, 270
полномочия, 267
работа в постоянном темпе, 283
самоорганизация, 273
упорядочение задела спринта, 269
характеристики и навыки, 270
целеустремленность и обязательность, 281
расписывание в лицах, 152
Руководители
обязанности, 55
проектов
обязанности, 315
сохранение отдельной роли, 318
функциональных подразделений
наделение команд полномочиями, 306
обязанности, 301
опека команд, 308
согласование и адаптация производственной среды, 310
управление ходом работ по созданию ценности, 313
устранение организационных препятствий, 311
формирование команд, 303

C

Самоорганизация, определение, 270
Скорость работы
злоупотребление, 199
измерение, 178
нормализация, 117
определение, 117; 193
причины употребления, 194
прогнозирование, 196
расчет скоростного диапазона, 194
факторы влияния, 198
Словарь терминов, организация, 501
Сложные адаптивные системы
самоорганизация, 272
характеристики, 272
Составление положения о проекте
определение, 384
особенности процесса, 385
Спецификация по образцу, определение, 139
Спринты
выполнение
виды работ, 446
временные рамки, 439
исполнители, 448
на уровне отдельных задач, 446
начало работ, 445
порядок, 64
предпосылки и конечный результат, 441
процесс, 440

участники, 440
контрольные точки, частость, 114
краткосрочность, преимущества, 112
назначение, 61
ненормальное завершение, 122
ограничение по времени, 62; 110
определение, 109
основные характеристики, 110
параллельная работа над элементами, 442
планирование, порядок проведения, 62
подведение итогов, назначение, 458
постоянная продолжительность, основания, 116
размеренный ритм, преимущества, 117
результаты, 67
ретроспектива, назначение, 472
роение, метод организации труда, 444
управление ходом работ, 442
цель
запрет на изменение, 118
изменение, последствия, 120
постановка, 62; 118
уточнение, особенности, 119
циклы проведения, 59
Стоимость задержки
определение, 99
особое значение, 351
порядок расчета, 352
профили, 352
учет, 99
Схватка над схватками
координирование работы команд, 294
метод, 294
порядок проведения, 295
схваток или на уровне программы, 295
Схваткопад, 77

Т

Темы, назначение, 141
Технический долг
варианты выхода из затруднения, 212
выявление
на деловом уровне, особенности, 218
на инженерном уровне, способы, 219
доска, применение, 220
известный, определение, 221
наивный, определение, 202
намеченный, определение, 221
неизбежный, определение, 202
неявное знание, 219
обслуживание
алгоритм, 221
неполное погашение, методы, 222
погашение высокопроцентного долга, 226
постепенное погашение, 225
применение правила бойскаутов, 224
сокращение долга при выполнении ценной работы, 227
определение, 201
последствия накапливания, 203
причины появления, 201; 208
случайно обнаруженный, определение, 221
стратегический, определение, 203
управление
контроль накапливания, 212
правильное понимание экономических последствий, 214
применение норм надлежащей инженерной практики, 213
употребление строгого критерия готовности, 214
Требования
заменители элементов в заделе продукта, 132
нефункциональные
включение в критерий готовности, 148
назначение, 147
составление, 147
обсуждение, 134
постепенное уточнение, 135
рассмотрение, особенности, 131
составление
в письменной форме, недостатки, 134
на одном уровне детализации, недостатки, 135

У

Упорядочение задела продукта
коллективное, участники, 162
назначение, 161
первоначальное, проведение, 164
порядок проведения, 163
разделение на области, 168
управление потоками
выпусков, 167
спринтов, 168
Утвержденное обучение

определение, 89
основные принципы, 89
петли обучения, применение, 90
Участники проекта
внешние, состав, 238
внутренние, состав, 238
Учет нововведений, этапы проведения, 315

Ф

Функциональные средства
желательные, определение, 168
как элементы задела продукта, 155
назначение, 155
необязательные, определение, 168
обязательные, определение, 168

Ц

Церемонии
как излишние формальности, примеры, 104
шкала, 104

Э

Экономический фильтр, применение, 356
Экстремальное программирование, нормы практики, 449
Элементы задела продукта
определение, 155
оценивание, 159
размещение по размерам, 157
разновидности, 155
расстановка по приоритетам, 160

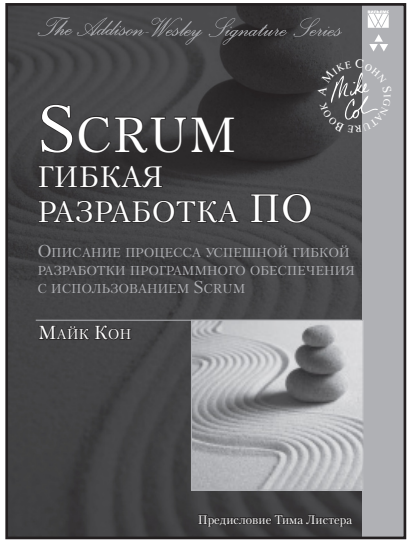
Я

Язык визуальных образов, назначение, 32

SCRUM
ГИБКАЯ РАЗРАБОТКА ПО

ОПИСАНИЕ ПРОЦЕССА УСПЕШНОЙ ГИБКОЙ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ SCRUM

Майк Кон



www.williamspublishing.com

Данная книга предназначена для прагматичных специалистов в области разработки программного обеспечения, которые хотят получить надежные, заслуживающие доверия ответы на большинство трудных вопросов, с которыми им приходится сталкиваться в процессе внедрения Scrum. В своей книге автор описывает все аспекты процесса внедрения: запуск процесса, оказание людям помощи в освоении новых ролей, структуризация коллективов, увеличение охвата, работа с рассредоточенным коллективом и, наконец, внедрение эффективных показателей и непрерывное совершенствование.
В книге встречаются врезки под заголовком “Попробуйте прямо сейчас”, включающие наиболее эффективные советы автора. Во врезках под заголовком “Возражения” автор воспроизводит типичные дискуссии с теми, кто сопротивляется переменам.

ISBN 978–5–8459–1924–3 **в продаже**

ПОЛЬЗОВАТЕЛЬСКИЕ ИСТОРИИ
ГИБКАЯ РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

М йк Кон

www.williamspublishing.com

В эт□й книге, вых□да к□т□р□й с нетерпением □жидал□ с□□бществ□ ст□р□нник□в гибких мет□д□л□гий разраб□тки пр□граммн□г□ □беспечения, □писывается пр□цесс п□дг□т□вки треб□ваний к разрабатываем□й системе, к□т□рый п□зв□ляет эк□н□мить время, избавляет □т не□бх□дим□сти в переделках и ведет к с□зданию б□лее с□вершенных пр□грамм. Лучший сп□с□б с□здать пр□граммн□е □беспечение, максимальн□ п□лн□ уд□влетв□ряющее п□требн□стям п□льз□вателей, — начать с п□льз□вательских ист□рий. Эт□ пр□стые, п□нятные и краткие □писания функци□нальн□сти, к□т□рая представляет дел□вую ценн□сть для реальных п□льз□вателей. В книге прив□дятся п□др□бные рек□мендации □тн□сительн□ т□г□, как следует писать п□льз□вательские ист□рии и включать их в жизненные циклы разраб□тки пр□екта. Вы узнаете, чт□ так□е х□р□шие п□льз□вательские ист□рии и чт□ делает ист□рии пл□хими. Вы п□знак□митесь с практическими мет□дами сб□ра ист□рий, п□зв□ляющими д□биться х□р□ших результат□в даже т□гда, к□гда в□зм□жн□сть неп□средственн□г□ □бщения с п□льз□вателями □тсутствует.

ISBN 978-5-8459-1795-9

в прод же

ГИБКОЕ ТЕСТИРОВАНИЕ
ПРАКТИЧЕСКОЕ РУКОВОДСТВО ДЛЯ ТЕСТИРОВЩИКОВ ПО И ГИБКИХ КОМАНД

Л йз Криспин
Дж нет Грегори

www.williamspublishing.com

В настоящей книге дается исчерпывающее определение гибкого тестирования и показана роль тестировщиков в реальных гибких командах. Подробно рассматривается использование квадрантов гибкого тестирования для идентификации потребностей в тестировании, анализируются требования к тестировщикам и предлагаются советы по построению набора инструментальных средств, помогающего проводить тестирование наиболее эффективно. В книге описана итерация гибкой разработки программного обеспечения с точки зрения тестировщика, а также объясняются семь ключевых факторов успеха гибкого тестирования. Читатели узнают, как вовлечь тестировщиков в гибкую разработку, каким образом произвести переход от традиционной циклической к гибкой разработке, как обеспечить полное выполнение всех действий по тестированию в течение коротких итераций, а также каким образом организовать управление процессом разработки с помощью тестов.

ISBN 978-5-8459-1625-9

в прод же

ГИБКАЯ РАЗРАБОТКА ПРИЛОЖЕНИЙ НА JAVA
С ПОМОЩЬЮ SPRING, HIBERNATE И ECLIPSE

Анил Хемраджани

www.williamspublishing.com

В этой книге основное внимание уделено разработке и в меньшей степени инфраструктуре. Другими словами, больше внимания уделено технологиям разработки приложений, таким как Spring, Hibernate и Eclipse, а не программным продуктам, таким как серверы приложений или базы данных. Все, что представлено в этой книге, опробовано в реальных приложениях, которые успешно работают (некоторые в кластеризуемой среде сервера приложений). Одна из задач этой книги заключается в краткости и конкретности, поэтому автор решил практически полностью сосредоточиться на разработке хорошо масштабируемого приложения. В данной книге, кроме технологий Spring, Hibernate и Eclipse, также описаны альтернативные и конкурирующие технологии.

ISBN 978-5-8459-1375-3 **в продаже**

ПРОГРАММИРОВАНИЕ.
ПРИНЦИПЫ И ПРАКТИКА С ИСПОЛЬЗОВАНИЕМ C++

ВТОРОЕ ИЗДАНИЕ

Бьярне Страуструп

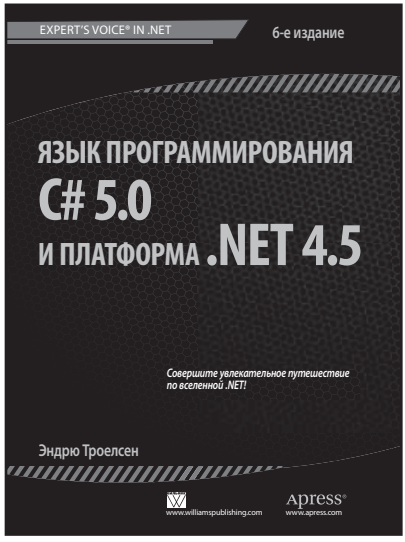
www.williamspublishing.com

Эта книга — учебник по программированию. Несмотря на то что его автор — создатель языка C++, книга не посвящена этому языку; он играет в большей степени иллюстративную роль. Книга задумана как вводный курс по программированию с примерами программных решений на языке C++ и описывает широкий круг понятий и приемов программирования, необходимых для того, чтобы стать профессиональным программистом.
В первую очередь книга адресована начинающим программистам, но она будет полезна и профессионалам, которые найдут в ней много новой информации, а главное, смогут узнать точку зрения создателя языка C++ на современные методы программирования.

ISBN 978-5-8459-1949-6 **в продаже**

ЯЗЫК ПРОГРАММИРОВАНИЯ C# 5.0 И ПЛАТФОРМА .NET 4.5
6-Е ИЗДАНИЕ

Эндрю Троелсен



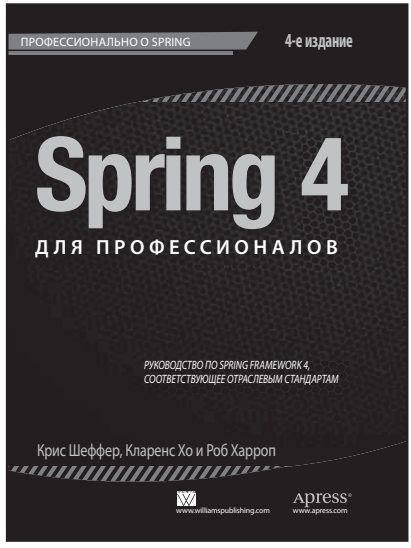
www.williamspublishing.com

Новое издание этой книги было полностью пересмотрено и переписано с учетом последних изменений в спецификации языка C# и дополнений платформы .NET Framework. Отдельные главы посвящены важным новым средствам, которые превращают .NET Framework 4.5 в самое передовое решение для корпоративных приложений. Помимо этого, рассмотрены все ключевые возможности языка C#, как старые, так и новые, что позволило обрести популярность предыдущим изданиям этой книги (материал покрывает все темы, начиная с обобщений и кончая pLINQ). Основное назначение книги — служить исчерпывающим руководством по языку программирования C# и ключевым аспектам платформы .NET (сборкам, удаленному взаимодействию, Windows Forms, Web Forms, ADO.NET, веб-службам XML и т.д.).

ISBN 978-5-8459-1957-1 **в продаже**

SPRING 4
ДЛЯ ПРОФЕССИОНАЛОВ

Крис Шеффер,
Кларенс Хо,
Роб Харроп



www.williamspublishing.com

Книга рассчитана на опытных Java-разработчиков, которые изучают Spring с самого начала или обладают поверхностным представлением о Spring Framework. Она ориентирована на тех, кто занимается или только планирует заняться разработкой корпоративных Java-приложений.

Вы изучите основы и ключевые темы, связанные с платформой Spring. Авторы поделятся с вами собственным реальным опытом в области удаленной обработки, использования Hibernate и работы с EJB. Помимо основ вы научитесь применять Spring Framework для построения разнообразных уровней или частей корпоративного Java-приложения: транзакций, веб-уровня и уровня презентаций, развертывания и многого другого.

Многочисленные примеры помогут вам в освоении технологий и приемов, рассмотренных в этой книге, а также в организации их совместной работы.

Прочитав эту книгу, вы научитесь с помощью Spring создавать сложные приложения от начала и до конца

ISBN 978-5-8459-1992-2

в продаже