Сбор и анализ требований к программному продукту

Версия 1.03

Автор: Химонин Юрий
program manager компании Acronis

Опубликовано: июнь 2009

Оглавление

Предисловие

Данная статья адресована менеджерам проектов и бизнес аналитикам, которые занимаются сбором и анализом требований к системе с последующим проектированием на их основе.

Цель статьи – предложить готовый подход (методологию) по сбору и анализу требований с последующим проектированием системы на их основе. Методики изначально рассматриваются в рамках итерационного или циклического циклов разработки, в отличие от методик ведущих издателей, которые в исходном виде могут быть использованы только в водопадной или каскадной моделях. Благодаря этому, приведенная методология идеально подходит компаниям, использующим или планирующим использовать гибкие модели разработки (*Agile*).

Особое внимание уделяется процессу формирования данных, входящих в устав будущего продукта, так как для большинства проектов фиксация содержания, срока и бюджета происходит именно в уставе (для проектов на заказ — в контракте) и не может быть пересмотрена в дальнейшем. Для выигрыша тендера или удержания клиента, срок, выделенный на этот этап, не превышает 2–3 недель, а потому для обеспечения высокой точности планирования сроков и бюджета нужно иметь идеально выверенную методику. Такую, как в данной статье.

Статья является компиляцией уже существующих подходов описанных в рамках PMBOK (PMI), Software Requirements Book (Microsoft), а также современных инициатив, которые используются в компаниях Acronis, Microsoft и Borland. Она направлена только на проекты по разработке программного обеспечения, а потому менее универсальна и более кратка и конкретна, нежели методология, представленная в рамках PMBOK. В уже зарекомендовавшие себя процессы автор интегрировал такие артефакты как пользовательские сценарии, навел порядок с такими неоднозначными элементами как «варианты использования» и систематизировал все типы требований.

Этапы сбора и анализа требований

Процесс работы с требованиями к продукту можно разделить на 4 этапа:

- Определение концепции продукта.
- Сбор требований.
- Анализ требований.
- Проектирование системы

Определение концепции продукта

На этапе определения концепции продукта, проводится работа с его инвестором, целью которой является выработка единого видения будущего продукта. По окончанию этого этапа производится вывод о том, будет ли этот продукт разрабатываться или нет.

Сбор требований

На этапе сбора требований основная работа ведется с заказчиком системы и её будущими пользователями. Цель этапа — точно определить функции продукта и способы его интеграции в существующие процессы.

Качественное выполнение работ на этом этапе гарантирует то, что будущий продукт будет соответствовать ожиданиям заказчика. Четкая расстановка приоритетов обеспечивает реализацию наиболее востребованной функциональности и исключение второстепенной/невостребованной функциональности, что сэкономит бюджет и сроки.

Анализ требований

На этапе анализа требований проходит структуризация уже собранных ранее требований. Цель этапа — предоставить четкий список не дублируемых требований к системе, которые должны быть выделены из избыточных и частично дублирующихся сценариев и пользовательских историй, которые были полученных на предыдущем этапе.

Правильно сгруппированные требования помогут обойтись минимальным количеством функционала для удовлетворения максимально большего количества целей, а это, в свою очередь, поможет сэкономить бюджет и не даст расползтись рамкам проекта.

Проектирование системы

Целью всех предыдущих этапов был сбор информации о том, кому и зачем необходим будущий продукт. Этап проектирования — это первый этап, на котором группа разработки принимает проектные решения о том, какую функциональность будет нести продукт, чтобы удовлетворить пользователей.

Результатом этого этапа является законченное техническое задание к продукту. Оно должно содержать полное описание поведения будущего продукта и не содержать неоднозначностей и вопросов.

На основе технического задания начинается моделирование работы продукта с конечными пользователями (используя макеты пользовательского интерфейса, к примеру) и производится тестирование технического задания. Это позволяет увеличить качество продукта и снизить его стоимость, так как стоимость внесения изменений в техническое задание всегда меньше, чем в конечный продукт.

Интеграция в жизненный цикл разработки продукта

Этап определения концепции продукта обычно выделяется в отдельный проект или является первым этапом в разработке продукта. Я рекомендую выделять его именно в отдельный проект, так как это дает возможность заранее определить время, которое вы хотите затратить на выработку концепции и одновременно не требует от вас определения сроков и бюджета конечного продукта на столь ранней стадии.

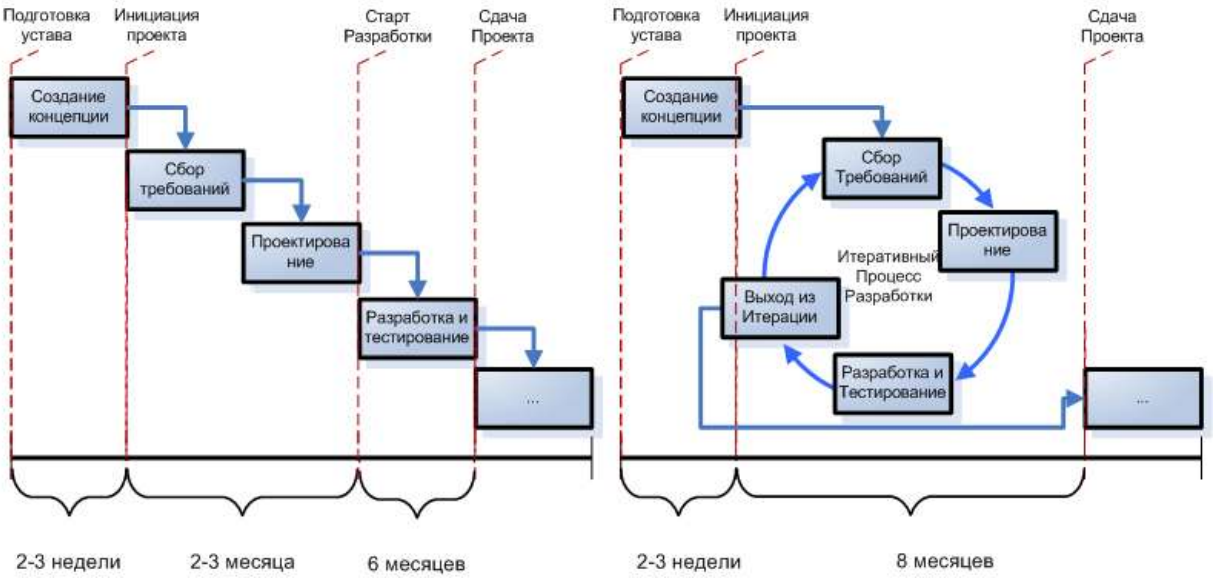

Рисунок 1. Место требований в жизненном цикле проекта, на примере каскадного и итеративного процессов

На этапе определения концепции считается, что идея уже родилась, её лишь надо развить и формализовать, а потому этот процесс может занимать от недели до месяца. Для проектов, с общей продолжительностью 6–12 месяцев, это срок обычно около 2–3 недель. После определения концепции проекта уже можно **экспертно оценить** необходимые ресурсы для выполнения проекта и срока, за который он будет выполнен, а также его прибыльность. **Точность оценки** полностью зависит от опыта людей, дающих оценку, но, как правило, не **превышает 50%**. Срок формирования концепции особенно критичен для продуктов, разрабатываемых под заказ, так как подписание контракта, как правило, происходит по завершении этой стадии.

Вы должны быть готовы к тому, что не каждый продукт способен принести вам деньги и даже окупить разработку, а потому, после определения концепции, вы должны трезво оценить шансы продукта на успех, оценить хватит ли вам ресурсов на его разра-

ботку и продвижение, сравнить его ликвидность с другими проектами в портфеле. В заключении вам надо принять решение — стоит ли заниматься разработкой продукта или нет. Необходимость принятия этого решения — еще одна причина выделять этот процесс в отдельный проект, так как решение о нецелесообразности инициации разработки продукта дается намного легче, чем решение о закрытии проекта (даже на столь ранней стадии). Желание продолжать работу над проектом любым способом, как правило, бывает сильнее здравого смысла и может привести к нежелательным последствиям.

Первый этап, с которого следует начинать разработку продукта — это сбор требований. Он может выполняться сразу и полностью (в рамках каскадного процесса) или же последовательно, по частям (в циклических и итерационных процессах).

Сразу за сбором требований идет их анализ. Эти два процесса могут выполняться одним человеком последовательно или параллельно группой людей (Группа, занимающаяся сбором требований, передает их аналитикам, которые сразу анализируют полученные данные).

Время, необходимое для сбора требований, может значительно отличаться для разных проектов. Например, для интеграционных проектов — этот процесс может занимать до половины человеко-часов от общей работы над проектом, для проекта автоматизации отдельных бизнес процессов только 5–7%. Продолжительность анализа требований, как правило, занимает в несколько раз меньше времени, чем сбор требований и в большей степени зависит от требований к документообороту, принятому в компании.

Проектирование — процесс предшествующий разработке продукта. Для проектов, использующих итерационную/циклическую модель разработки, с самого начало должно производиться высокоуровневое проектирование системы — определение высокоуровневой архитектуры, а в начале каждой итерации должна происходить её детализация.

Влияние качества работ на характеристики конечного продукта

Каждый из четырех вышеперечисленных этапов проводится на разных уровнях, а потому имеет различную степень влияния на характеристики будущего продукта. Вследствие этого невозможно линейно дать оценку, какой этап более важен, а какой менее. Но этот вопрос неизбежно встанет перед вами при планировании временных затрат.

Таблица 1. Влияние качества работ на характеристики конечного продукта.

Этап	Характеристики продукта
Разработка концепции	• Ожидаемая прибыль, срок вывода на рынок и бюджет продукта. • Дальнейшие перспективы развития продукта.
Сбор требований	• Качество — соответствие продукта ожиданиям заказчика. • Срок вывода на рынок и ориентировочный бюджет продукта.
Анализ требований	• Эффективность разработки. • Возможность корректировать функционал продукта в процессе реализации, с целью его удешевления или охвата новых рынков /пользователей.
Проектирование системы	• Эффективность разработки — позволяет уменьшить количество вносимых изменений в уже готовый продукт. • Качество реализации. • Риски проекта (возможность их снижения). • Эффективность разработки большими и/или распределенными командами.

В таблице выше приводится влияние каждого из этапов работы с требованиями на характеристики конечного продукта. В зависимости от того, какие характеристики продукта для вас важнее, вы сможете определить, на какой этап потратить больше времени, а на какой этап меньше.

Сферы ответственностей: потребности клиентов и предложенные решения

Все требования в продукте подразделяются на две основных категории: *потребности* (пользователей) и *решения* (функции), которые будут реализованы в продукте для удовлетворения потребностей.

Если в процессе работы над продуктом требуется изменить решение, то достаточно утверждения нового решения проектной командой. Если же изменениям подвергается потребность, то это обязательно должно быть обговорено с клиентом продукта.

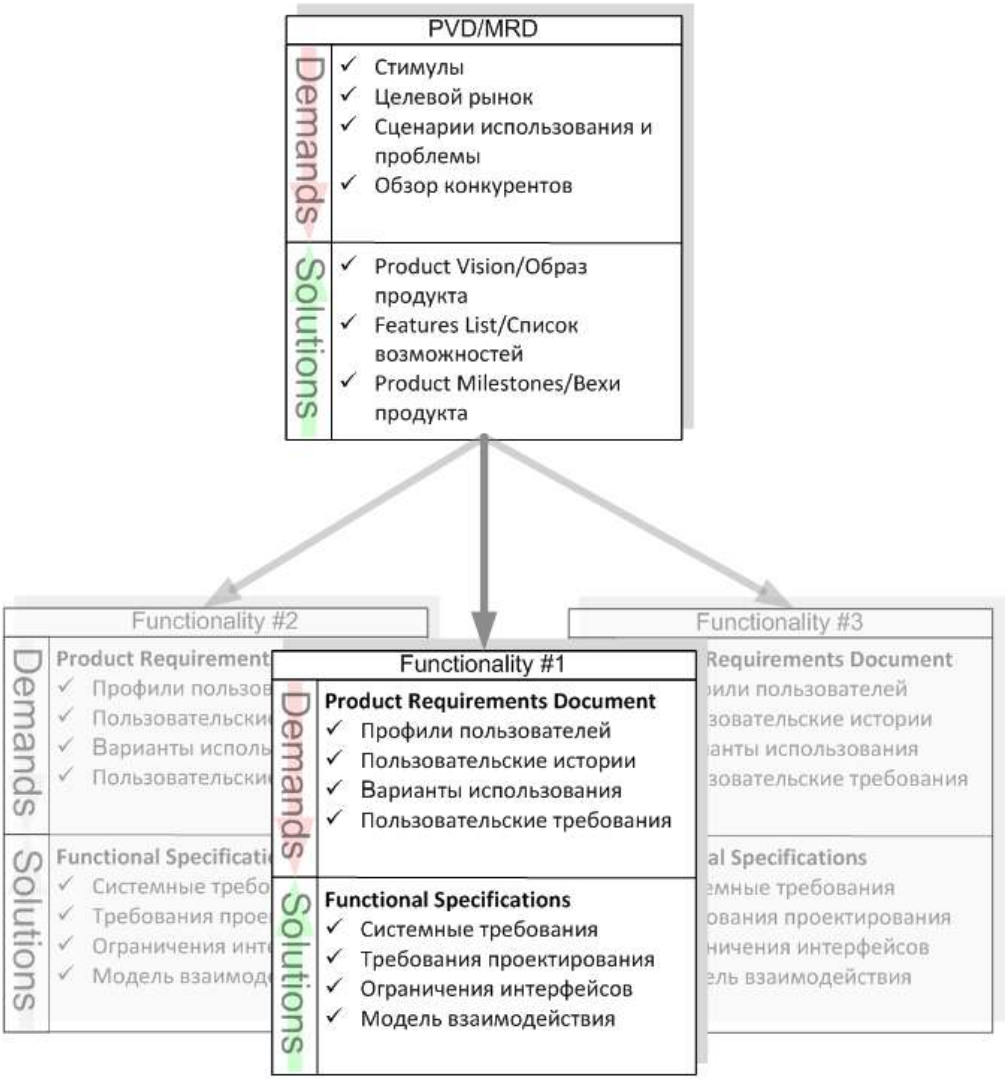

Рисунок 2. Потребности и решения в требованиях продукта.

I. Разработка концепции продукта

На этапе определения концепции продукта выясняются причины побудившие заказчика начать разработку продукта и определяются основные требования к нему. Далее командой аналитиков создается образ решения для продукта и ограничения текущей его версии.

Ведущую роль в разработке концепции ведет **бизнес аналитик** (или Program/Product Manager), занимающийся этим продуктом. Для определения потребностей заказчика/рынка на весь срок разработки концепции проводится интенсивная работа с **инвестором проекта**. Цель — выработка единого видения будущего продукта. Если это заказной продукт, то инвестором является конечный заказчик. Если продукт предназначен для открытого рынка, то ответственными за него являются учредители компании или совет директоров. Далее на основе изученных и систематизированных требований аналитик вместе с командой экспертов создает образ будущего продукта. В заключении, аналитик с экспертной командой определяют границы проекта, которые должны содержать ориентировочные сроки и бюджет продукта.

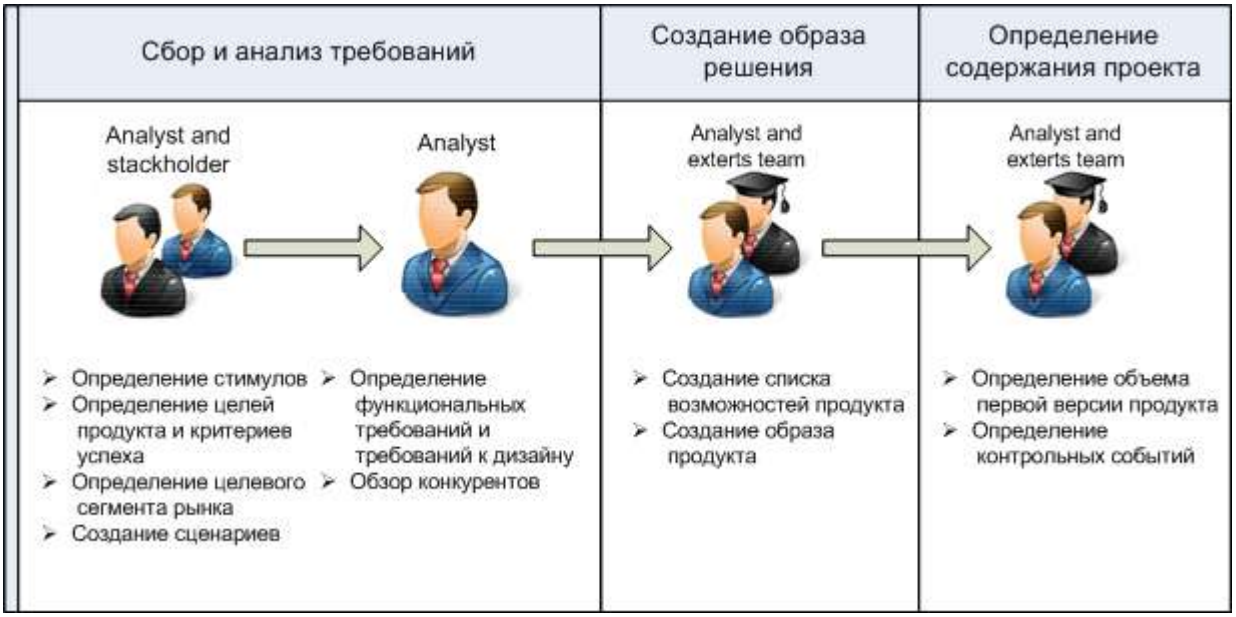

Рисунок 3. Этапы разработки концепции продукта.

Результатом разработки концепции является **Product Vision Document** (если продукт разрабатывается под заказ) или **Marketing Requirement Document** (если продукт предназначен для открытого рынка). Эти документы содержат подробную информацию о требованиях заказчика, возможностях продукта, которые должны удовлетворять эти требования, а также ориентировочные сроки его реализации и бюджет. Различием между PRD и MRD является то, что в MRD обычно более детально описаны целевые сегменты рынка и сделан детальный обзор конкурентов.

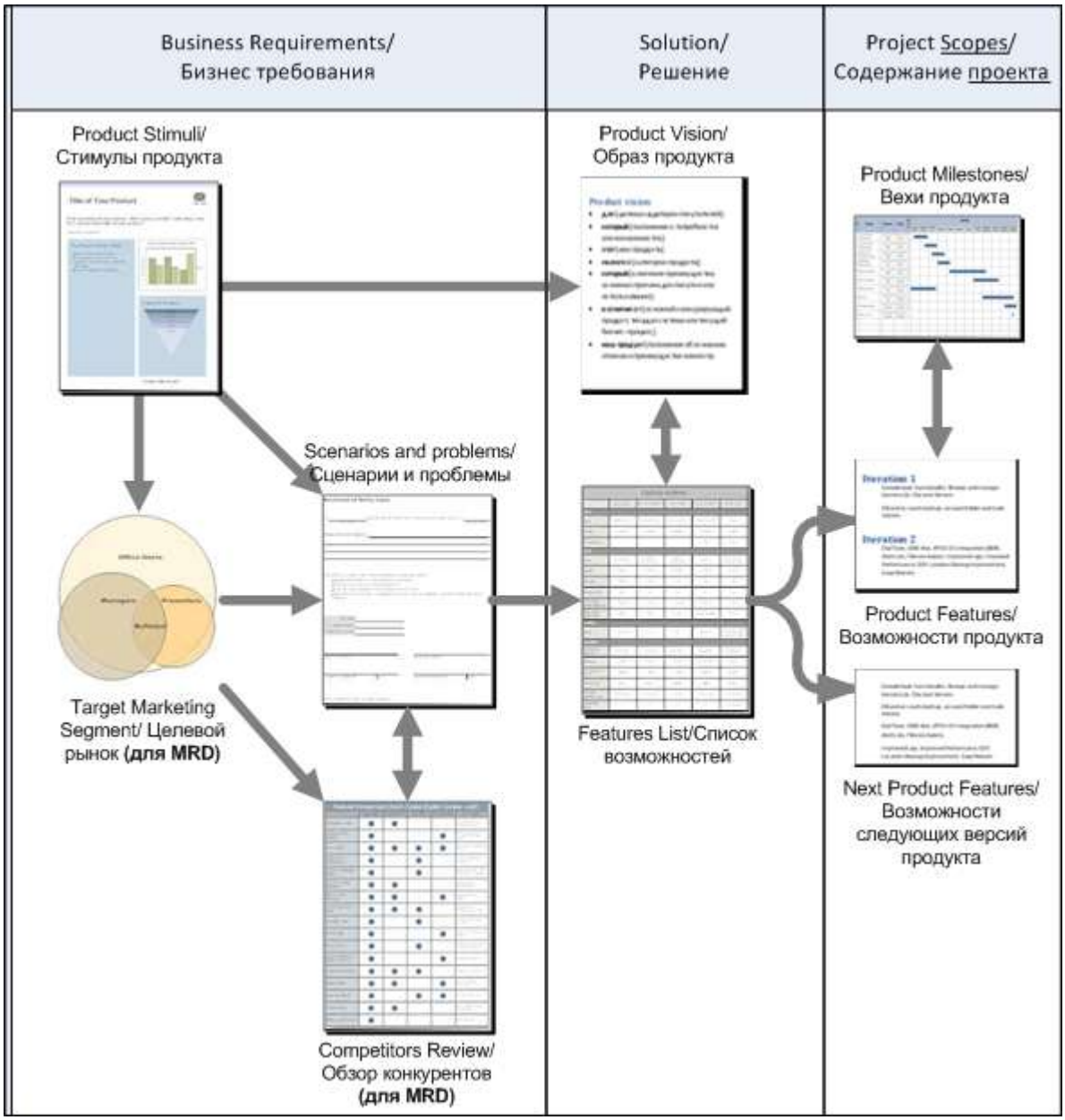

Рисунок 4. Содержание MRD или Product Vision Document.

По окончании разработки концепции продукта делается вывод о целесообразности разработки продукта. В случае принятия положительного решения, создается устав проекта по разработке продукта. PVD/MRD является входным документом устава проекта, описывающим содержание работ по проекту.

Сбор и анализ бизнес требований

Первым и самым важным этапом в разработке продукта является **сбор бизнес требований**. Цель этой работы — определить основные требования бизнеса (исходные данные, истинные цели, которым должен служить продукт и проблемы, которые нужно преодолеть).

Для продуктов под заказ и продуктов для открытого рынка процесс сбора бизнес требований существенно различается.

- **Продукт под заказ** — заказчик определен с самого начала, ему известны начальные предпосылки (стимулы) для инициации проекта и проблемы, которые продукт должен решать. В связи с этим, сбор требований начинается с определения исходных предпосылок, целей продукта и описания сценариев работы пользователей с будущим продуктом. Анализ конкурентных продуктов, которые могут удовлетворять схожие сценарии, делается в самом конце.

- **Продукт для открытого рынка** — сектор рынка с самого начала может быть не определен, а цели продукта основываются на конкурентном анализе.
В результате, сразу за определением исходных предпосылок (стимулов) идет обзор конкурентов, далее идет определение целевого сегмента рынка и потребностей его заказчиков и только после этого определяются цели продукта и критерии его успеха.

Таблица 2. Последовательность задач, выполняемых на этапе сбора бизнес требований.

Продукт под заказ	Продукт для открытого рынка
• Определение исходных стимулов • Определение целей продукта и критериев успеха • Определение потребностей клиента • Обзор конкурентов	• Определение исходных стимулов • Обзор конкурентов • Определение целевого сегмента рынка • Определение потребностей клиента • Определение целей продукта и критериев успеха

Определение стимулов (Initial prerequisites)

На данном этапе указываются основные причины, которые стимулируют принятие решения о создании этого продукта.

Как правило, причиной создания продукта может служить одна или несколько из нижеперечисленных проблем, благоприятных возможностей или требований бизнеса:

- Потребность рынка (например, банк авторизует проект миграции клиентского ПО на платформу Mac OS в ответ на возросшую рыночную долю этой ОС);

- Производственная необходимость (например, R&D отдел авторизует проект интеграции системы контроля версий с сервером статистики для повышения уровня контроля за достижениями сотрудников);

- Потребность заказчика (например, крупный сборщик компьютеров авторизует проект кастомизации существующего ПО под компьютеры собственного производства для достижения большей совместимости);

- Технический прогресс (например, производитель ПО авторизует проект разработки новой версии продукта, использующей новые возможности только что вышедшей ОС);

- Юридические ограничения или нормы (например, банк может авторизовать проект перехода продукта на новый метод шифрования, соответствующий крите-

риям современных норм безопасности, для работы с государственными заказчиками).

Определение целей продукта и критериев успеха

Следующей задачей является определение одной или нескольких целей, которые преследуют заинтересованные стороны при разработке продукта. Необходимо описать основные преимущества, которые предоставит продукт для бизнеса. Сделать это надо в измеряемом виде. Также нужно определить механизм, используя который, заинтересованные люди будут измерять успех продукта в конечном итоге.

Далее приведены основные цели, которые обычно преследуются при разработке программного обеспечения (Wiegers, 2002):

Финансовые:

- Освоить X% рынка за Y месяцев.
- Увеличить сектор рынка в стране X на Y% за Z месяцев.
- Достигнуть объема продаж X единиц или дохода, равного $Y, за Z месяцев.
- Получить X% прибыли или дохода по инвестициям в течение Y месяцев.
- Достигнуть положительного баланса по этому продукту в течение Y месяцев.
- Сэкономить $X в год, которые в настоящий момент расходуются на обслуживание системы.
- Уменьшить затраты на поддержку на X% за Z месяцев.
- Получить не более X звонков в службу обслуживания по каждой единице товара и Y звонков по гарантии каждой единицы товара в течение Z месяцев после выпуска товара.
- Увеличить валовую прибыль для существующего бизнеса с X до Y%.

Нефинансовые:

- Достигнуть показателя удовлетворения покупателей, равного, по крайней мере, X, в течение Y месяцев со времени выпуска товара.
- Увеличить производительность обработки транзакций на X% и снизить уровень ошибок данных до величины не более Y%.
- Достигнуть определенного времени для достижения доминирующего положения на рынке.
- Разработать надежную платформу для семьи связанных продуктов.
- Разработать специальную базовую технологическую основу для организации.
- Получить X положительных отзывов в отраслевых журналах к определенной дате.

- Добиться признания продукта лучшим по надежности в опубликованных обзорах продуктов к определенной дате.
- Соответствовать определенным федеральным и государственным постановлениям.
- Уменьшить время оборота до X часов на Y% звонков покупателей в службу поддержки.

Определение целевого сегмента рынка

Этот элемент является частью MRD и, как правило, требуется только для продуктов, ориентированных на широкий рынок. Принято сегментировать рынок следующим образом:

- **Рынок домашних пользователей** (может быть также разделен на обычных и продвинутых пользователей).
- **Рынок корпоративных пользователей.**
    1. **SMB** (Small and Medium Business) — компании, насчитывающие от 1 до 250 сотрудников. Также может быть разделен на Micro (или Soho) — 1–10 сотрудников, Small — 10–25 и Medium — 25–250.
    2. **Large** — компании, насчитывающие 250–2500 сотрудников.
    3. **Corporation** — корпорации с числом сотрудников более 2500.

Это разделение принято использовать в связи с тем, что требования, налагаемые пользователями, которые принадлежат к разным сегментам рынка, кардинально отличаются. Это касается как функциональности, способов администрирования, так и вопросов лицензирования и поддержки продукта. Практически невозможно создать продукт, который бы подошел одновременно домашним пользователям и в тоже время мог бы эксплуатироваться в крупной компании.

Для того чтобы правильно выбрать сегмент рынка, необходимо определить требования, налагаемые каждым из сегментов рынка с учетом предметной области продукта.

Таблица 3. Часто встречающиеся нефункциональные требования.

	Домашний пользователь	Малый и средний бизнес	Крупная компания
Администрирование	Не требуется.	Интуитивно понятное, не требующее спец. обучения. Интеграция с AD на уровне пользователей.	Удаленное или централизованное (в зависимости от пр. обл.). Интеграция с AD на уровне пользователей и прав. Централизованная или, как минимум, удаленная установка.
Интерфейс	Графический. Красивый, удобный и быстрый.	Графический.	• Графический • Командный • SDK
Требования к безопасности.	—	Защита паролем.	Защита сертификатом.
Отказоустойчивость.	В зависимости от прикладной области	В зависимости от прикладной области	В зависимости от прикладной области
Масштабируемость*	—	Поддержка до 250 компьютеров	Поддержка до 2000 компьютеров.
Лицензирование	Лицензии на: • одну копию • пользователя	Лицензии на: • одну копию • пользователя • группу	Использование специальных лицензий.
Поддержка	• Email • форум	• Email • Форум • телефон	• Email • Форум • Телефон • Выделенный инженер для разворачивания и поддержки продукта.
Способ продажи	• Интернет • магазин	• Интернет • магазин	Дилер

** Требует тестирования на указанном количестве продуктов.*

При выборе сегмента рынка следует определить требования, продиктованные каждым из сегментов к вашему продукту. Выбор целевого сегмента должен основываться на собственных возможностях: бюджете проекта, квалификации аналитиков и разработчиков, наличии возможностей для продвижения продукта.

Я бы не рекомендовал сразу претендовать на рынок крупных корпоративных пользователей. Как правило, путь к корпоративному сегменту у всех разработчиков ПО начинается с SMB или с рынка приложения для домашних пользователей. Многие компании потерпели неудачу при попытке выйти на рынок крупных корпоративных пользователей, не имея у себя в портфеле аналогичных продуктов для других сегментов рынка, даже когда сегмент рынка был свободен, и его можно было легко занять. Но подобные предположения, как правило, бывают ложными. Если в домашнем сегменте борьба может идти между десятками или даже сотнями продуктов, то в корпоративном сегменте их может быть всего несколько, и при этом рынок будет полностью занят.

Определение потребностей клиентов или рынка (от сценариев к требованиям)

Теперь, когда определен целевой сегмент для вашего продукта, следует определить кто, какие проблемы и в каких условиях будет решать при помощи будущего продукта.

Создание сценариев

Наиболее эффективным способом получения ответа на эти вопросы является определение сценариев работы пользователей с будущим продуктом. *Сценарий* — это совокупность **всех** процессов, в которых будет участвовать продукт, а также описание окружения, в котором его планируется использовать. Сценарий не должен являться описанием работы отдельного пользователя для достижения конкретной цели. Его ценность состоит в том, что он описывает способы взаимодействия с продуктом всех его пользователей одновременно на протяжении всего цикла эксплуатации продукта. Таким образом, сценарий гарантирует отсутствие взаимоисключающих требований к продукту и дает возможность легко убедиться, что никто и ничто не забыто. Для проверки сценария надо всего лишь проанализировать его выполнение всеми заинтересованными лицами (проиграть его).

Для продуктов под заказ сценарии использования продукта формируются самим заказчиком. Как правило — сколько заказчиков, столько и сценариев. Наиболее эффективным методом является живой диалог с заказчиком, в котором аналитик задает наводящие вопросы (пытается разговорить заказчика), а заказчик отвечает на них. Если личный контакт невозможен, как правило, помогают вопросники, содержащие «нужные» вопросы, по которым заказчику легче будет написать сценарий.

Для открытого рынка сначала определяются профили будущих клиентов продукта, а затем для каждого из них создается подробный сценарий его использования. Аналитик может описывать сценарии самостоятельно, используя информацию из личного опыта или открытых источников. Другой вариант, позволяющий достичь явного преимущества — найти клиентов или компании, подходящие под ранее определенные профили и получить сценарии непосредственно от них.

> *Важно не путать понятие клиента и пользователя. Клиентом может являться компания, в которой множество сотрудников будут являться пользователями системы. В таком случае, профиль клиента описывает характерные черты и проблемы компании, а профиль пользователя (описываемый позднее) — характеристики её сотрудников.*

Благодаря использованию сценариев акцент делается на реальные потребности конкретных пользователей системы и лишь потом определяется необходимый функционал продукта. Каждый сценарий содержит все процессы, в которых планируется использовать продукт, а потому исключается возможность того, что продукт будет удовлетворять только часть требований к системе. Также на этапе формирования сценариев очень легко определить те ситуации или группы пользователей, которые в принципе не смогут быть удовлетворены будущим продуктом в силу технических или других ограничений. Это избавит участников проекта от излишних ожиданий и позволит отказаться от реализации свободных функций (непривязанных ни к одному из сценариев, которые вы намерены удовлетворить).

Каждый сценарий должен содержать:

- Имя конкретного заказчика или его профиль (в качестве заголовка).
- Информацию обо всех типах пользователей, которые будут работать с продуктом.
- Все процессы, которые будут затрагивать продукт.
- Операционная среда, в которой будет использоваться продукт.
- Требования к дизайну: операционная система; приложения, с которыми интегрируется, форматы ввода вывода.
- Приоритет. Зависит от того, насколько важен этот заказчик или как много покупателей будут попадать под профиль клиента, описанного в сценарии.

Для приоритезации я рекомендую использовать метод MoSCoW (детальнее http://en.wikipedia.org/wiki/MoSCoW_Method). В нем используются четыре оценки — Must, Should, Could и Won't.

*Элементы, помеченные как **Must** обязаны быть включены в продукт. Проект не может считаться успешно завершенным, если хоть один подобный элемент (сценарий/возможность/требование) не включен в продукт.*

*Элементы **Should** также являются критическими для продукта, но, тем не менее, они могут быть исключены из текущей версии продукта по объективным причинам.*

*Элементы **Could** являются некритичными, но способны увеличить пользовательскую удовлетворенность.*

*Элементы **Won't** являются наименее критичными для продукта. Они могут считаться интересными и перспективными для будущих версий продукта, но точно не будут реализованы в текущей версии.*

Определение функциональных требований и требований к дизайну

Теперь, имея всю необходимую информацию от пользователей, следует заняться её систематизацией.

Сценарии избыточны. Они содержат большое количество повторений или дополнений. В процессе выделения требований должен быть создан список уникальных требований к продукту, которые не должны повторяться, но могут дополнять друг друга. Необходимо выделить два типа требований: *функциональные бизнес требования* и *требования к дизайну* (нефункциональные требования).

Функциональные бизнес требования описывают потребность бизнеса (заказчика/покупателя), которую должен удовлетворить будущий продукт. Основной вопрос, на который должны отвечать бизнес требования — зачем бизнесу та или иная функциональность (подробнее об этом читайте в примечании «Зачем? Что? и Как?»).

Требования к дизайну описывают операционную среду, в которой будет функционировать продукт, интерфейсы взаимодействия с пользователем или другими системами, форматы хранения и передачи данных, а также требования к надежности, производительности, обслуживанию и доступности системы. Эти требования в значительной степени влияют на средства разработки, архитектуру и используемые технологии при разра-

ботке продукта, а значит и на конечный бюджет и сроки продукта. Поэтому крайне важно как можно более точно определить важнейшие требования к дизайну именно на стадии определения концепции.

В процессе структурирования нужно выделить основные требования и дополнения к ним. Требования удобно отображать в виде дерева. Независимые требования являются требованиями первого уровня, а их дополнения дочерними элементами. Не стоит создавать требования — контейнеры, это усложнит работу в будущем.

Разбивая требования на отдельные элементы, основным критерием должна быть возможность реализации этих требований отдельно друг от друга (реализовать первое и не реализовать второе). Если требования столь сильно зависят друг от друга, что должны быть реализованы только вместе, лучше их объединить. Цель разделения требований на составные части — получение возможности принимать решения о целесообразности реализации каждого требования в отдельности и назначать им различные приоритеты, что в дальнейшем обеспечит гибкость в определении списка требований для первой и всех последующих итераций продукта или исключении из текущей версии продукта. Это критично для продуктов, бюджет и сроки которых строго определены и не могут быть пересмотрены.

Как только все требования структурированы, следует назначить им приоритет. Наилучшим методом является комбинирование унаследованного приоритета (от родительских сценариев) с приоритетом требования внутри сценария. Для этого вы должны приоритезировать требования в рамках сценариев по шкале MoSCoW и сложить с приоритетом сценария, в который они входят. Если требование входит в несколько сценариев, то берется максимальная оценка.

Таблица 4. Пример вычисления приоритета требования.

	Must	Should	Could	Would
Must	Must	Should	Could	Would
Should	Should	Could	Would	Would
Could	Could	Would	Would	Would
Would	Would	Would	Would	Would

Обзор конкурентов

Для выпуска продукта на рынок требуется придать ему уникальность, определить, чем он лучше других, почему именно его будут покупать? Такими мотивами может быть, например, большая функциональность, скорость, масштабируемость, удобство использования или более низкая цена, в конце концов. Поэтому для определения текущего положения на рынке крайне важно серьезно подойти к обзору конкурентов. Причем этот документ будет важен как при создании продукта для открытого рынка, так и для продукта на заказ.

Вторая цель обзора конкурентов — рассмотрение идей, реализованных в продукте. Цель проста — использовать наиболее удачные решения, реализованные в конкурирующих продуктах, и исключить неудачные. Так сказать, сделать работу над ошибками других.

Структура обзора конкурентов обычно следующая:

1. Конкурентное положение на рынке.
2. Список конкурентов (Резюме по каждому конкуренту).
3. Список проблем, которые призваны решать продукты.
4. Список возможностей продуктов.

В процессе создания обзора вам потребуется пройти этапы, описанные ниже.

1. Определить список конкурентов на рынке и выделить лидеров.
Вся дальнейшая работа проводится с лидерами на рынке. Определить лидеров можно, используя различные обзоры, результаты опросов или продаж. Учтите, что в числе лидеров может оказаться не только продукт с отличным функционалом, но и бесплатное приложение, предоставляющее необходимый минимум возможностей.

Если предметная область продуктов достаточно популярна, то в сети можно найти уже готовый обзор, который будет содержать необходимую для вас информацию. Обзор конкурентов, как правило, самый продолжительный этап определения концепции продукта, а потому я настоятельно рекомендую воспользоваться как можно большим количеством готовых материалов, которые помогут сэкономить ваше время.

2. Узнать цену и способ доставки конкурентов. Также нужно постараться достать демонстрационную версию продукта. Если это не удалось, то надо хотя бы найти маркетинговые материалы, содержащие список возможностей (как правило, описывается в документе DataSheet) и руководство пользователя.

3. Составить список проблем, которые решает каждый конкурент. Этот раздел обзора конкурентов особенно интересен при проектировании продукта для открытого рынка, так как благодаря ему аналитик сможет определить максимальное количество проблем, для решения которых пользователи захотят купить будущий продукт. Грамотная приоритезация поможет добиться максимальной востребованности продукта при фиксированных вложениях.

Составить список проблем можно, исследовав продукт самостоятельно, но более эффективный способ — просмотреть документацию по продукту. Качественные продукты содержат сценарии использования продукта в тех или иных ситуациях, а маркетинговые материалы — выгоды, которые сулит продукт при его использовании.

Все проблемы, для решения которых созданы продукты, должны отвечать на вопрос «зачем?».

Суммарную информацию о конкурентах желательно поместить в таблицу. В ней нужно указать, кто имеет возможность решать указанную проблему (сегмент рынка или профиль пользователей) и насколько важно иметь возможность её решать (обязательно (*essential*), полезно (*useful*), желательно (*desirable*)). В ячейке продукта напротив каждой проблемы нужно указать предоставляет ли конкурент эту возможность или нет (обычно помечаются как «+», «−» или «+/−»). Также полезно дополнять записи кратким описанием проблемы и заметками о конкурентах. Это особенно важно если проблема решается необычным методом или частично (позже, даже для автора обзора станет неясно, что же конкретно означает «+/−» в ячейке одного из конкурентов).

Таблица 5. Пример списка проблем.

Problem	Majority	Market Segment	Competitor 1	Competitor 2	Competitor 3	Competitor 4	Description	Competitors Notes
Increase Free Space	E	All	+	+	+	+	System volume size is growing all the time and free space will be over.	
Online Increase Free Space	E	S	−	+	+	−		
Improve existing volume performance	U	H	−	−	+/−	−	Ability to increase existing volume read/write speed.	C3 able to convert Simple-to-Striped, but add disk to Striped function is absent.

Target marketing segments: S — Server, H — Home, E — Essetial, U — Useful.

По завершении исследования проблем, которые решают продукты конкурентов, следует провести повторный анализ бизнес требований к своему продукту. Полученная информация о конкурентах поможет вам сделать бизнес требования к вашему продукту лучше.

> *Совет: Если вы занимаетесь проектированием новой версии продукта, то имеет смысл добавить предыдущую его версию в качестве конкурента. Это поможет вам увидеть текущее конкурентное положение продукта и различия между старой и новой его версиями.*

4. Составить список возможностей. Здесь нужно описать все важные возможности, которые были реализованы в конкурентных продуктах для удовлетворения проблем, описанных в предыдущем пункте. На основе этого списка можно узнать, как хорошо продукт решает заявленные проблемы, какие у него сильные и слабые стороны.

На этом этапе вам придется работать либо с самим продуктом, либо с его документацией.

> *Я рекомендую делать как можно больше снимков экрана, для того чтобы их можно было использовать в процессе разработки продукта. Наличие снимков экрана поможет вашим дизайнерам быстро изучить предметную область и легко распознать удачные или неудачные идеи конкурентов, чтобы использовать эти знания при разработке вашего продукта.*

Результатом работы будет таблица со списком возможностей, которые предоставляют все продукты. В столбце продукта напротив каждой возможности должно быть указано предоставляет ли конкретный продукт эту возможность (обычно помечаются как «+» или «−»). Также полезно дополнять записи кратким описанием возможности и заметками о конкурентах.

Таблица 6. Пример списка возможностей.

Functionality/ Feature	Majority	Market Segment	Competitor 1	Competitor 2	Competitor 3	Competitor 4	Description	Competitors Notes
Резервное копирование	E	All	+	+	+	–		
Копирование файлов	U	All	+	+	+	–		
Копирование томов	E	All	+	+	–	–		Продукт 1 использует VSS
Восстановление	E	All	+	+	+	+		
BMR	E	All	–	+	–	–	Восстановление на голое железо.	Загрузочный CD продается за отдельные деньги
Восстановление файлов	U	All	+	+	+	+		Только из файлового архива

5. Обобщая всю полученную информацию, полезно **составить резюме по сильнейшим конкурентам**. Следует описать их преимущества и недостатки, выделить интересные идеи.

6. Если вы проектируете продукт для открытого рынка, то в заключении необходимо **описать конкурентное положение на рынке** *(несмотря на то, что этот пункт описывается в конце, он добавляется в начало обзора конкурентов)*. В нем нужно обозначить зрелость целевого рынка и выделить продукты, с которыми будет вестись наиболее ожесточенная борьба.

Далее необходимо перечислить наиболее перспективные пути развития продукта и его шансы на успех.

Создание образа решения

При работе над образом решения пришло время аналитикам определить очертания будущего продукта, который бы удовлетворял всем требованиям, описанным в предыдущем пункте.

На этом этапе определяются все основные характеристики, которыми должен обладать будущий продукт, чтобы достичь ранее поставленных целей. А на основе экспертной оценки уже могут быть определены ожидаемые сроки и бюджет продукта.

Образ решения будет служить доказательством того, что разработчики выбрали правильный путь (или напротив), и оказывает наибольшее влияние при выборе команды-разработчика для тендерных проектов.

Создание списка возможностей будущего продукта

Теперь, когда известно кому и зачем нужен продукт, следует определить какую функциональность он будет предоставлять для удовлетворения потребностей будущих пользователей. Задача упрощается тем, что после обзора конкурентов известны решения, которые уже предлагаются конкурентами, их слабые и сильные стороны.

Начать работу, как это ни странно, нужно с определения нефункциональных возможностей, которые должны удовлетворять ранее собранным требованиям к дизайну. Эти возможности имеют максимальное влияние на дизайн продукта, а потому должны быть четко определены на стадии создания концепции. Именно от списка нефункциональных возможностей будет зависеть сложность реализации основных функций продукта. Следовательно, они будут определять стоимость реализации и тестирования этих функций, ведь одно дело, к примеру, разрабатывать продукт под конкретную платформу (например, Microsoft Windows), и совсем другое, реализовывать функционал для кросс-платформенного продукта.

Нефункциональные возможности должны содержать:

- Имя возможности
- Приоритет возможности.
- Наличие этой функции у конкурентов (для каждого конкурента).
- Краткое описание возможности.
- Примечание для конкурентов.

Важно, что приоритет для нефункциональных возможностей распространяется только на тестирование продукта и исправление ошибок, но инфраструктура будет разрабатываться под *все нефункциональные возможности*, которые решено реализовывать. На более поздних этапах, манипулировать списком нефункциональных возможностей станет намного сложнее, чем функциями продукта.

Затем нужно добавить основные функции, которые требуется реализовать в продукте.

Каждый элемент списка возможностей должен содержать:

- Имя функции, которую требуется реализовать в продукте.
- Приоритет.
- Наличие у конкурентов (желательно для каждого из конкурентов)
- Краткое описание (если требуется).
- Отметки о конкурентах.
- Экспертную оценку необходимого времени и затрат, связанных с реализацией функциональности (формат оценки во многом зависит от модели управления рисками, принятыми в компании).

При экспертной оценке считается, что функционал продукта должен удовлетворять все нефункциональные возможности. Если одна или несколько нефункциональных возможностей сильно увеличивают стоимость функции, то следует сделать пометку об этом и позднее обсудить целесообразность поддержки этой (их) нефункциональной возможности продуктом или отдельными его функциями.

В результате список возможностей должен содержать все высокоуровневые возможности, которые будут реализованы в продукте, не будут реализованы, но уже есть в конкурирующих решениях, а также уникальные возможности еще никем не реализованных. Приоритет следует унаследовать от родительских требований, также как это делалось ранее.

Создание образа продукта

На протяжении всего жизненного цикла разработки «Образ продукта» должен являться дорожной картой, которой нужно пользоваться, чтобы не сбиться с пути. В силу этого он должен содержать самые главные данные и его описание не должно быть больше половины страницы. Для достижения этой цели лучше всего подходит следующий шаблон, который содержит все необходимые ключевые вопросы и тезисы *(Moore, Geoffrey A. 1991. Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers. New York: HarperBusiness)*:

- «для» — целевая аудитория покупателей;
- «который» — положение о потребностях или возможностях;
- «этот» — имя продукта;
- «является» — категория продукта;
- «который» — ключевое преимущество, основная причина для покупки или использования;
- «в отличие от» — основной конкурирующий продукт, текущая система или текущий бизнес-процесс;
- «наш продукт» — положение об основном отличии и преимуществе нового продукта.

Список возможностей должен полностью соответствовать образу решения. Если между образом решения и списком возможностей есть расхождения, вы должны это исправить (модернизировать список возможностей или образ решения).

Определение содержания проекта

Говоря об образе продукта, мы подразумеваем список направлений, по которым он должен развиваться, а с**одержание относится к определенному проекту или его итерации**, в которых реализуется меньше возможностей продукта, чем это возможно. Образ продукта будет изменяться относительно медленно. Содержание в свою очередь более динамично, так как менеджер проекта изменяет содержимое каждой версии в соответствии с графиком, бюджетом, ресурсами и критериями качества. **Задача планирования заключается в управлении содержанием определенного проекта** (разрабатываемого или расширяемого), как определенным подмножеством большого стратегического образа.

Определение объема первоначальной версии продукта

Объем первоначальной версии продукта зависит от целей, которые возлагаются на продукт. Как правило, объем первоначальной версии должен быть минимально необходимым для того, чтобы удовлетворить пользователей. Это делается для того, чтобы как можно раньше занять нишу на рынке или предоставить решение заказчикам.

Для продуктов, разрабатываемых под заказ, функционал строго определен заказчиком. Но для того чтобы наладить обратную связь с пользователем на раннем этапе, продукт часто разбивается на несколько версий (сначала выпускается первая версия продукта, затем следующая). Вместо этого я бы рекомендовал использование итеративного процесса разработки, в котором результатом определенных итераций должен быть продукт готовый к промышленному внедрению на ограниченном количестве рабочих мест (бета-тестирование).

Определение контрольных событий проекта

Определение контрольных событий — это обязательный этап процесса разработки проекта, который тесно переплетен со сбором и детализацией требований к проекту. Поэтому, не смотря на то, что он относится к дисциплине управления проектами, он все равно будет рассмотрен в рамках этой статьи.

После определения концепции проекта невозможно точно определить его срок, и бюджет может быть определен лишь на основе экспертной оценки, его точность обычно не превышает 50% (в лучшем случае). По завершении сбора и анализа требований, точность оценки, сделанной на их основе, уже может достигать 60–70%. И только после завершения проектирования опытный менеджер сможет определить бюджет с точностью до 80–90%.

Из этого следует, что точно определить сроки и бюджет проекта на ранних стадиях невозможно. Для проектов, разрабатываемых на основе водопадной модели, определить сроки и бюджет проекта можно завершив стадию проектирования, когда около 30% работы по проекту уже завершено. Для проектов, в основе которых стоит циклическая или итерационная модель — срок и бюджет не могут быть точно определены до самого конца.

Но, не смотря на это, часто инвесторам проекта нужен срок и бюджет проекта сразу после определения концепции. Для решения этой проблемы используется комбинирование экспертной оценки и приоритизации требований.

Суть решаемой проблемы в том, что у вас есть три величины: срок, бюджет и содержание/качество (стандартный треугольник проектного управления), при этом две из них (бюджет и срок) нужно зафиксировать на этапе, когда их еще нельзя точно определить.

В самом начале проекта нужно экспертно оценить сроки проекта и его бюджет, применить сведения о рисках проекта и заложить резерв (на данном этапе уместно использовать двукратное увеличение оценок). Для выполнения этой работы должны быть привлечены аналитики, эксперты предметной области и менеджер будущего проекта (или человек, имеющий опыт управления проектами). Результатом работы является оценка срока и бюджета проекта. Как говорилось ранее, точность этой оценки обычно не превышает

50%, так что чем большие резервы вы заложили, тем лучше и качественнее продукт вы получите в конечном итоге.

Далее происходит процесс определения контрольных событий. **Для итерационного или циклического проекта — контрольными событиями, как правило, являются выходы из итераций.** Каждая итерация ассоциируется с теми возможностями, которые вы планируете реализовать в её теле. Список возможностей формируется по формуле: не более 50% трудозатрат должно уйти на обязательный функционал (приоритет — Must), еще 25% на необходимый (приоритет — Should) и еще 25% на желательный (приоритет — Could). При выполнении работ во время итерации вы сначала реализуете обязательную функциональность, далее необходимую и, в самом конце, желательную. Если оценка затрат была неверна, то на выполнение обязательной функциональности у вас будет в 2 раза больше времени, чем было запланировано.

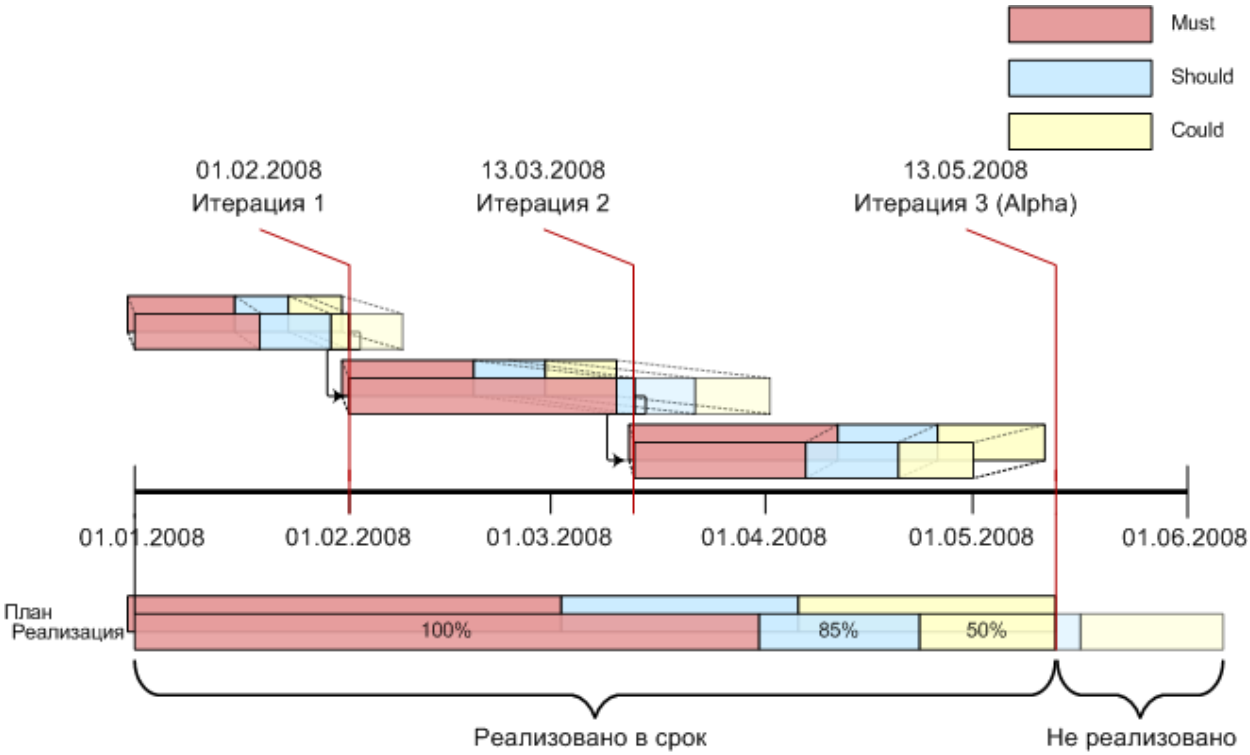

Рисунок 5. Пример выполнения работ в срок.

Благодаря этому подходу вы увеличите свои шансы на то, чтобы выпускать продукт в срок, с наиболее востребованным функционалом. Да, часть желаемых возможностей может быть не реализована при данном подходе, но главное — реализовано все, что на самом деле важно. Если же пользователю действительно важны второстепенные функции, вы всегда можете добавить еще одну итерацию и реализовать их (изменив сроки и бюджет проекта).

II. Сбор требований

Уже известны основные сценарии использования будущего продукта, определены бизнес требования и высокоуровневые возможности. Теперь нужно детализировать требования к продукту. Основная работа на этом этапе ведется с будущими пользователями продукта.

При использовании итеративного процесса вся функциональность поделена на части (это описано ранее), поэтому сбор и анализ требований, а позднее проектирование системы ведется строго в рамках функциональности, которая должна быть реализована в текущей итерации.

Определение основных профилей пользователей

Для того чтобы продукт был удобен пользователям и делал «то, что надо», сначала надо определить, кто же им будет пользоваться.

На практике, даже для домашних продуктов очень сложно определить «среднего пользователя», чтобы на основе его потребностей проектировать продукт. Для корпоративных продуктов это попросту невозможно: директор будет пользоваться одной функциональностью, его секретарь другой, а бухгалтер третьей. По этой причине перед началом сбора требований должны быть определены основные профили пользователей.

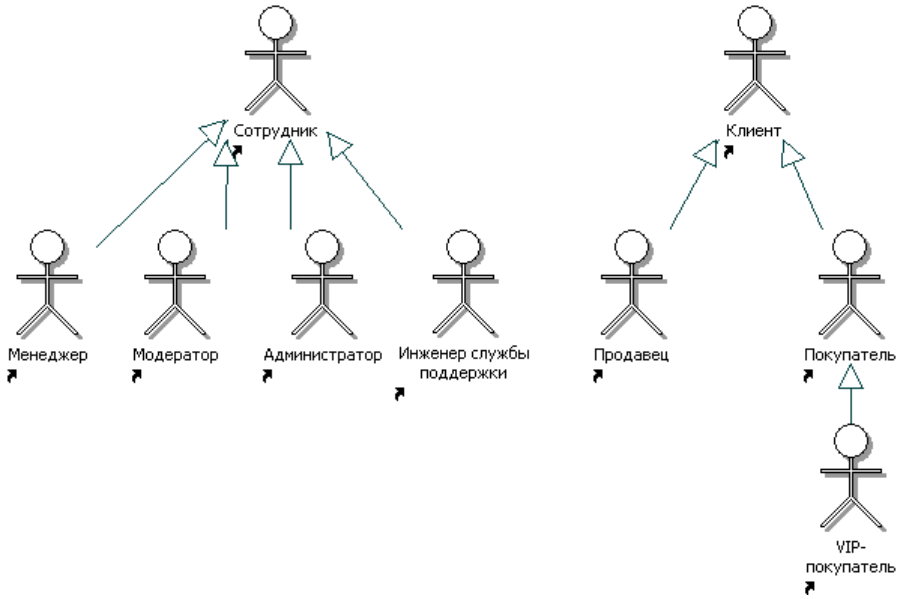

Рисунок 6. Пример диаграммы профилей пользователей продукта.

Профили пользователей явно или неявно уже должны содержаться в сценариях, поэтому все, что нужно сделать — это выделить их. Для домашних продуктов разделение обычно производится, исходя из уровня подготовленности пользователя и его интересов, для корпоративных продуктов основным критерием является специализация работника и круг его обязанностей.

Формирование инициативной группы

Когда определены профили пользователей продукта, следует найти людей, соответствующих этим профилям и желающих помочь вам в разработке продукта. Для проектов под заказ это достаточно простая задача — нужно выбрать одного или двух наиболее грамотных и инициативных людей для каждого профиля. В случае разработки проекта для открытого рынка вам придется искать таких людей среди коллег и знакомых. В последнее время эту задачу значительно облегчили социальные сети — блог на любую профильную тему сейчас можно найти практически на любом языке, правда, наладить контакт с его авторами не всегда возможно.

Работа может считаться выполненной, когда у вас есть как минимум один источник для каждого описанного выше профиля, и связь с источниками налажена.

Сбор пользовательских историй

Настала очередь начать общение с конечными пользователями и узнать для достижения каких целей будет использоваться будущий продукт. Лучшим методом для достижения этой цели является сбор пользовательских историй.

Пользовательская история — это вариант использования будущего продукта в конкретной ситуации с целью достижения измеримого результата. Каждая пользовательская история должна приносить пользу — не должно быть историй, которые выполняют действия ради действий.

Пользовательские истории могут содержать как сложные инструкции с ответвлениями, так и конкретные примеры. Если действия пользователей продиктованы скорее здравым смыслом, нежели инструкцией, то пользовательская история должна содержать пример реальной ситуации, а их систематизация и оптимизация должна быть оставлена на потом. Если же бизнес процессы, которые автоматизирует продукт, строго формализованы, то пользовательская история должна содержать алгоритмы действий продукта или людей, работающих с ней.

Важно помнить, что пользовательская история не описывает способ работы с конкретным продуктом и использует только термины предметной области. Она должна быть понятной и родной людям, которые знают процесс, автоматизируемый продуктом. Способ реализации всегда должен оставаться за кадром (он будет определен намного позже).

Структура пользовательской истории полностью соответствует стандартному варианту использования, за тем лишь исключением, что указание последовательности действий, которая будет выполняться в рамках варианта использований, является необязательной. Далее указаны поля вариантов использования (полужирным шрифтом помечены обязательные):

- **Идентификатор** («Уникальный номер» плюс «Имя»).
- Источник/Автор.
- Дата создания.

- Профиль пользователя (Действующее лицо/Актер в UML). *Если в рамках истории происходит взаимодействие между различными пользователями, то их следует указать через запятую, а в последовательности действий описать принадлежность пользователей к ним. Также удобно определить профиль по умолчанию — он будет иметься в виду, если поле не заполнено.*

- **Приоритет**.

- Частота использования (числа от 1 до 10 или текстовые значения: «почти никогда», «один раз», «редко», «время от времени», «часто», «очень часто», «каждый N минут/часов/дней»)

- **Родительское бизнес требование**.

- Предусловие. *Описание начального состояния. Может содержать причины/порядок действий, которые привели к проблеме, решаемой в пользовательской истории.*

- **Цель/ Результат.** *Цели, которые преследует пользователь в рамках пользовательской истории. По возможности следует указывать мотивы, которые побудили его (зачем это надо?) и ожидаемый результат.*

- Последовательность действий. Может быть указан в *качестве примера или содержать алгоритм взаимодействия с системой для формализованных процессов*.

  Важно! Последовательность действий в конечном продукте вероятнее всего будет отличаться от того, что будет указанно в варианте использования.

Основной акцент в описании пользовательской истории должен быть сделан на цели, которую хочет достичь её автор. В качестве последовательности действий можно указать, какого поведения ожидает пользователь от продукта при достижении поставленной цели, но ни в коем случае не стоит пытаться смоделировать поведение продукта для этой ситуации, так как на этом этапе это будет напрасная трата времени и сил (исключение составляют продукты, в основе которых положен стандарт).

Каждая пользовательская история имеет одно или больше родительских бизнес требований и ее главная цель описать наиболее удачные способы их удовлетворения. Именно на основе родительского требования устанавливается приоритет истории. Если один из пользователей просит реализовать историю, которая не подчинена бизнес требованию, нужно обговорить добавление соответствующего бизнес требования в продукт с его непосредственным инвестором. В случае положительного решения, нужно отразить изменение в подходящем сценарии.

Зачем? Что? И как?

Самая главная ошибка, которую делают при сборе требований к продуктам, это формирование образа продукта на основе ответов на вопрос «Что должен делать продукт» или еще хуже «Как должен работать продукт». Целью же сбора требований, является по-

лучение ответа на вопрос — «**Для чего** нужен продукт». А потому, единственный вопрос, который должен задаваться пользователю — это «Зачем?».

> *Пример: пользователь продукта начинает рассказывать, что ему нужно три мастера, которые бы выполняли определенные задачи, просит использовать определенные цветовые схемы и форматы данных. Как правило, такой случай заканчивается полным фиаско. В процессе ввода продукта в эксплуатацию окажется, что пользователь не учёл мнения других людей, которые будут работать с продуктом, не указал исчерпывающих требований к аппаратной платформе. И, как результат, первая версия продукта будет иметь ошибки в проектировании (например, клиентская часть продукта не будет поддерживать Mac OS X и, как следствие, сильно ограничит рынок сбыта), будет отсутствовать часть важнейшего функционала, без которого невозможно использовать её в рамках основных бизнес процессов. А значит, продукт не будет отвечать требованиям реального бизнеса.*

Кто виноват? Как правило, и вы и тот, кто предоставлял вам эту информацию. В случае, если вы разрабатываете продукт на заказ вы, возможно, сможете минимизировать свои убытки, заранее регламентировав полный список работ в договоре, но морального удовлетворения вам это не принесет. Если же продукт ориентирован на широкий рынок, то это будет означать, что вы не получите ожидаемую прибыль.

Какой вывод? Нужно очень четко контролировать поступающую от заказчика информацию. Требования должны быть продиктованы бизнесом и быть нацелены на достижение реальных и *измеряемых* результатов. Одним словом реализация требования должна приносить пользу, а не являться самоцелью. Если пользователь говорит, что неплохо было бы сделать «мастер того-то и того-то» нужно мгновенно среагировать и задать вопрос — «зачем этот мастер ему нужен». Как правило, он с радостью расскажет о реальных мотивах, которые и интересуют!

> *Пример: пользователь сказал, что ему нужна функция авто-ввода для поля «фамилия». Это требование отвечает на вопрос «что», а потому прежде чем его вносить, нужно задать вопрос: «зачем нужна эта функция?» ответом на этот вопрос, скорее всего, будет что-то вроде «Нужно максимально сильно ускорить процесс ввода персональных данных оператором о клиенте». Требование будет выглядеть так:*
>
> *Нужно максимально сильно ускорить процесс ввода персональных данных оператором о клиенте. Одним из возможных способом может быть реализация функции авто-ввода фамилии из базы уже имеющихся клиентов.*
>
> *Теперь команда проектирования пользовательского интерфейса знает реальное требование к будущему интерфейсу, а команда тестирования, знает критерии для его тестирования.*

III. Анализ требований

После завершения этапа сбора требований вы **уже располагаете всей необходимой информацией** о требованиях к будущей системе, но эта информация не систематизирована и часто дублируется. Для небольшого продукта это неважно, и он может обойтись без большинства стадий анализа, быстро перейдя к проектированию, но для крупного продукта это приведет к большому количеству ошибок в процессе проектирования и, как следствие, к увеличению бюджета и срока разработки. Чем больше продукт, тем более важной является стадия анализа.

Основной целью анализа требований является их систематизация и избавление от дублируемых данных. Это достигается за счет разделения пользовательских историй на отдельные пакеты по функциональному признаку и их иерархической структуризации.

По окончании этапа анализа требований, многостраничный документ, содержащий сотни пользовательских историй, будет разбит на части. Каждая часть будет освещать только необходимую функциональность, а в её основе будет стоять диаграмма вариантов использования, на основе которой можно будет легко увидеть все требуемые функции системы. Описание вариантов использования не будут дублироваться, а лишь дополнять друг друга.

Выделение пользовательских историй в отдельные пакеты

Первым этапом анализа требования является выделение пользовательских историй в отдельные пакеты требований. Для специализированных систем управления требованиями вроде Borland Caliber RM или Rational Request Pro, в которых все требования хранятся в базе данных, и представлены древовидным списком, пакетами являются элементы первого уровня, которые будут играть роль контейнера для всех остальных требований. При использовании текстовых документов пакетами будут являться отдельные файлы, возможно с общим индексом.

Главная цель формирования пакетов — упростить доступ к нужным данным, за счет того, что все варианты использования относящихся к определенной функциональности можно будет увидеть на одной странице (оглавление или диаграмма вариантов использования). В случае использования текстовых документов, пакеты также существенно упростят автору процесс последующего редактирования — не нужно будет блокировать весь документ на время редактирования, а пользователям документа будет легче узнать об изменениях (как правило, для этого используется секция «История изменений» в начале каждого документа). Для того чтобы достичь поставленной цели требуется добиться того, чтобы в один пакет входило 20–30 пользовательских историй.

Пакеты формируются из пользовательских историй, которые описывают схожую деятельность или способ достижения схожего результата. Как правило, всего они подчинены одному бизнес требованию.

Для средних/крупных продуктов часто создается дополнительный пакет, в который включено множество непохожих историй, которые не имеют схожих признаков, но должны быть рассмотрены.

Работу над этапом анализа можно считать законченной, когда вы выделили все пользовательские истории в отдельные пакеты. Каждый пакет имеет понятное название, а описание содержит критерии, на основе которых в него попали или не попали пользовательские истории.

Два подхода: Варианты использования или спецификации требований

Существует два основных метода проектирования — проектирование на основе вариантов использования и проектирование на основе требований.

Проектирование на базе вариантов использования считается более эффективным, так как этот метод позволяет не терять связь с пользовательскими историями и прекрасно иллюстрирует требуемое поведение системы в целом, а, следовательно, гарантирует востребованность всего функционала, который будет создан (очень расточительно и болезненно для разработчиков писать код, которым никогда не удастся воспользоваться). Но все же есть условия, при которых аналитик может пренебречь этим методом в пользу проектирования на базе требований:

- Проектирование на основе требований следует предпочесть, если необходимо сократить затраты на стадию анализа до минимума, а количество пользовательских историй в пакете не велико (их содержимое можно просмотреть не дольше чем за 10 минут)
- Команда разработки не умеет или не хочет работать с вариантами использования и требует предоставление требований к системе в классическом виде.

Далее описаны работы, которые нужно произвести в рамках выбранного вами метода.

A. Варианты использования. Структурирование пользовательских историй

Как правило, у всех пользовательских историй в пакете есть одна или несколько главных целей, и есть история, которая описывает наиболее простой способ их достижения. Такая история называется — базовой, а описание её действия — базовый путь. Остальные истории описывают альтернативный способ достижения результата или содержат дополнительные действия для достижения специфического результата и по большому счету являются дополнениями к базовой истории.

Пользовательские истории это разновидность стандартных вариантов использования, с той лишь особенностью, что они описывают взаимодействие не с реальной, а с гипотетической системой. Это свойство пользовательских историй будет использовано в этой главе и все манипуляции в процессе структурирования будут производиться

в соответствии с правилами и методами, разработанными для стандартных вариантов использования.

Поиск базовых вариантов использования

Главным критерием определения базового варианта использования является наличие общих со всеми остальными вариантами использования действий (сродни базовому классу в ООП). Базовым вариантом использования может быть существующая пользовательская история, которая имеет результат (приносит пользу), или абстрактный набор действий, который создан лишь для выделения общих шагов. Для определения базового варианта использования нет четкого правила, часто в одном и том же пакете есть возможность создать абстрактный вариант использования или взять за базу одну простейшую/несколько более сложных историй — и все эти решения будут верными.

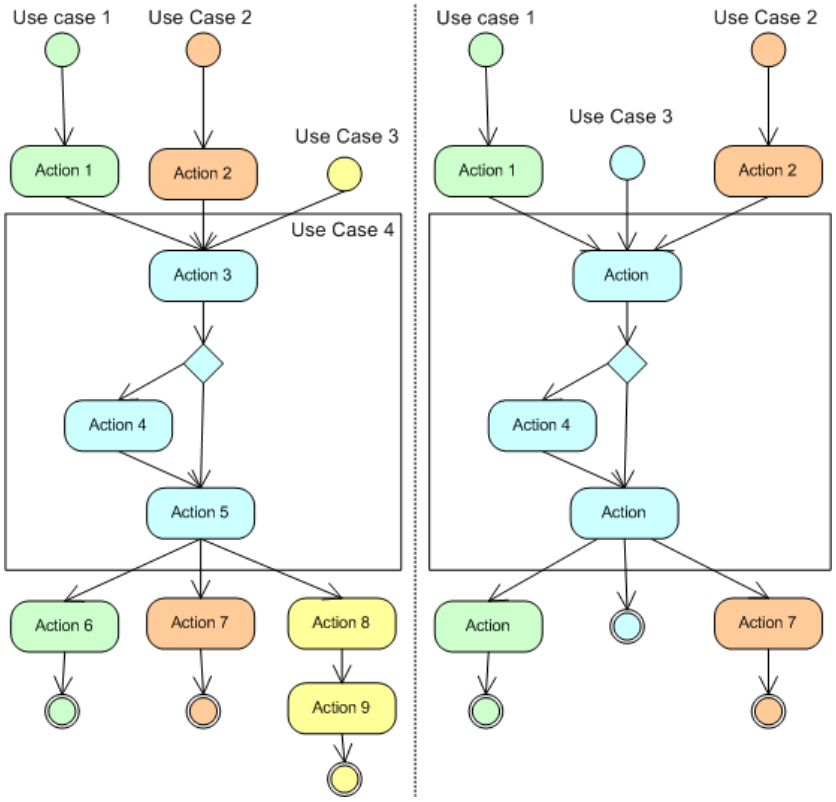

Рисунок 7. Пример выделения базового варианта использования. На рисунке слева — абстрактный (use case 4), на рисунке справа — реальный (use case 3).

Составление дерева пользовательских историй на диаграмме вариантов использований.

Теперь, когда есть один или несколько базовых вариантов использования, пора описать его связи с другими вариантами использований. Лучшим способом для этого является использование диаграммы вариантов использования. Этот вид диаграмм входит в стандарт UML и уже давно получил широкое распространение в сфере проектирования ПО, а поэтому у вас не должно возникнуть проблем с поиском редактора. На рынке представлены, как бесплатные решения с базовым набором функций, так и промышленные решения, самыми распространенными из них являются Rational Rose, Borland Together, Microsoft Visio.

Дополнительную информацию о способах работы с диаграммами вариантов использования вы можете найти, например, здесь: http://ru.wikipedia.org/wiki/UML.

В процессе проектирования на диаграмме вариантов использования очень важно видеть приоритет каждого элемента, очень удобно для этого использовать цветовое обозначение. Как правило, используются зеленый (must), желтый (should), голубой (could), красный (won't), также желательно, чтобы цвета были хорошо различимы и в черно-белой палитре (при печати на черно-белом принтере).

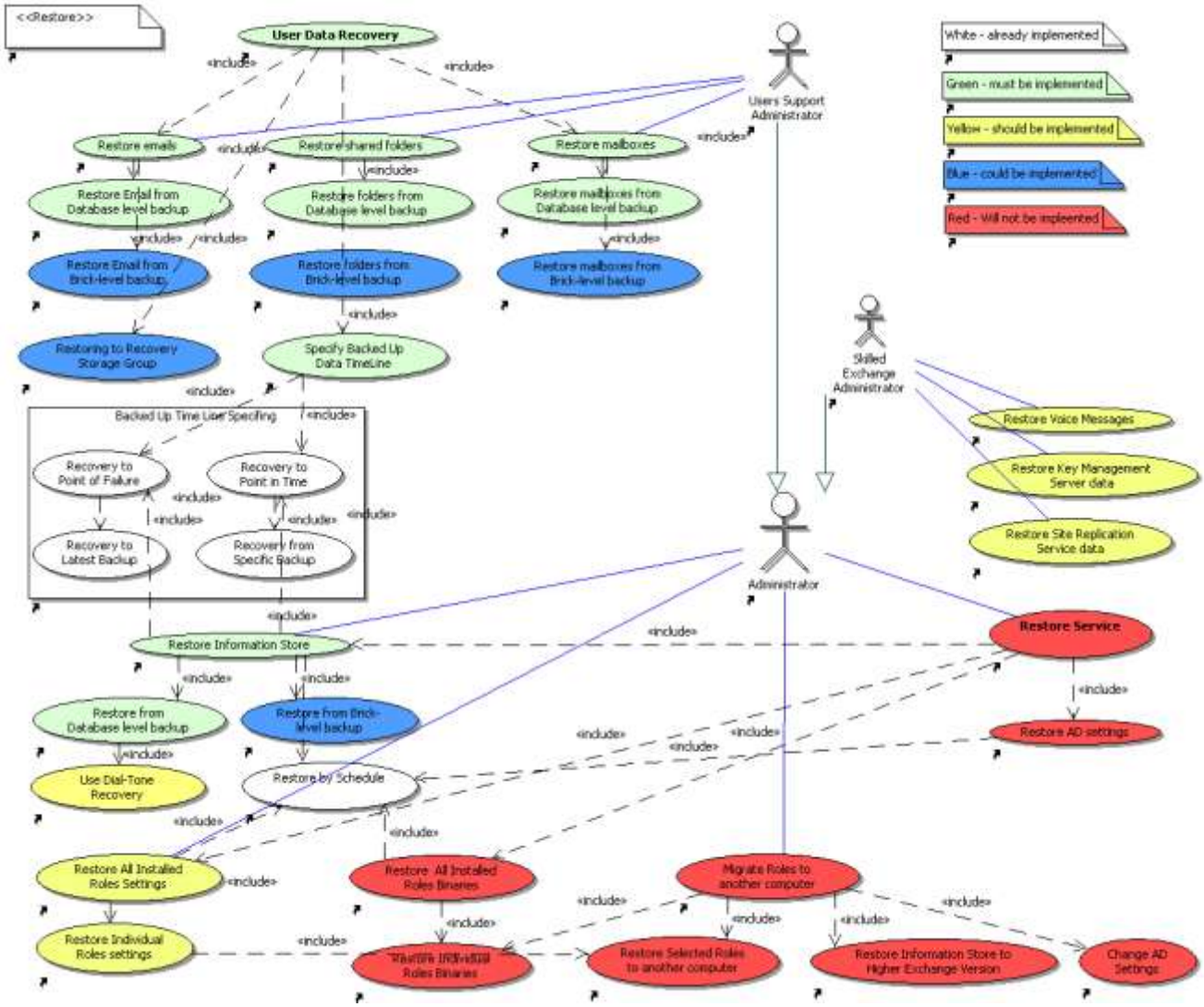

Рисунок 8. Пример UML диаграммы вариантов использования с цветовой маркировкой приоритетов.

В конечном итоге у вас должна получиться одна или несколько диаграмм, содержащих все варианты использования системы и их связи между собой.

Акцентирование на различительных признаках в теле пользовательских историй

После того, как все отношения между вариантами использования определены, следует отредактировать их тело.

На этапе сбора пользовательских историй можно было заметить, что поток действий большинства пользовательских историй содержит очень много повторений. На этапе

определения зависимостей общие части были определены, и на их основе была определена структура элементов на диаграмме вариантов использования. Теперь следует отредактировать тело вариантов использования, с целью исключения общих частей из их предусловий, постусловий и потоков действий.

Все варианты использования, за исключением базовых, являются расширением к другим вариантам использования. Поэтому в теле варианта использования надо указывать ссылку на вариант(ы) использования, который он расширяет. В предусловиях, постусловиях и потоке действий должны быть указаны только действия, уникальные именно для этого варианта использования.

Цель этой работы, повысить информативность описания вариантов использования и избавиться от дублирования данных в пакете. В итоге, каждый вариант использования должен содержать только уникальную информацию.

Поиск взаимоисключающих пользовательских историй с целью их исключения

Взаимоисключающие требования достаточно частое явление для крупного продукта, так как у него много пользователей, а значит много мнений. Для того чтобы противоречащие требования не порождали ошибки в процессе проектирования или еще хуже, в процессе разработки, большинство из них должны быть выявлены еще на этапе анализа.

Если считать, что при сборе требований аналитик не дал возможности пользователям принимать проектные решения, и, в результате, пользовательские истории содержат только «правильную» информацию (описаны мотивы пользователей), то у вас будет два типа взаимоисключающих требований представленных ниже.

1. Ошибка в пользовательской истории. Автор допустил ошибку в описании истории, указав неправильное поведение системы. Как правило, основным признаком таких историй является невозможность их интеграции с другими элементам, но для уверенности нужно проконсультироваться с автором или специалистом в предметной области, прежде чем удалить.

2. Разные взгляды на одну и ту же проблему. Несколько авторов имеют разные мнения о том, как должна работать система. Это проблема может иметь два решения:

- Можно выбрать «более правильную» точку зрения и оставить только её. Такое решение, как правило, можно принять на основе приоритетов историй.
- Определить дополнительные условия, при которых обе конфликтующие истории будут выполняться в рамках одной системы. Это решение может значительно усложнить систему, но в большинстве случаев, оно сделает её более гибкой (она будет ложиться не только на процессы директора, но и подчиненных).

Дробление вариантов использования

Для того чтобы иметь максимальную гибкость в процессе проектирования функциональности необходимо разбить варианты использования на неделимые элементы. Признаком того, что вариант использования требуется разбить на части, является то, что он описывает решение сразу нескольких проблем, которые могли бы быть решены по отдельности.

Результатом этого процесса должна быть диаграмма вариантов использования, каждый элемент которой описывает свою уникальную цель.

Преобразование диаграммы вариантов использования в список пользовательских требований

Диаграмма вариантов использования является отличным инструментов для проектирования, ознакомления и дальнейшего модернизирования системы, но есть процессы, где её использование не представляется возможным или, как минимум, затруднительным (в нашем случае одним из таких процессов будет планирование). В связи с этим необходимо преобразовать диаграмму вариантов использования в нумерованный список. Большинство промышленных средств проектирования представляют возможность сделать это автоматически, если же выбранный вами инструмент не содержит эту функциональность, то вам следует произвести преобразование самостоятельно, применяя следующие правила:

- Все элементы, которые не являются включениями или расширениями,, должны стать главными узлами списка.
- Элементы, являющиеся расширениями, должны стать дочерними элементами списка по отношению к тем, кого они расширяют. Если элемент имеет несколько родителей, то следует воспользоваться средствам трассировки (рекомендуется) или скопировать элемент в каждую ветку (с пометкой, что это копия).

Далее работы будут вестись как со списком, так и с диаграммой, поэтому оба этих представления следует поддерживать в актуальном состоянии.

Б. Спецификация требований

Для нормальной разработки нужен список требований, который однозначно идентифицирует потребности пользователей и не имеет множественных повторений, которые присутствуют с избытком в пользовательских историях (если не произвести структурирование историй, описанное в предыдущем подходе). Кроме этого, для более гибкого проектирования необходим как можно более детализированный (раздробленный) список.

Извлечение требований из пользовательских историй

На этом этапе нужно выделить все уникальные результаты пользовательских историй в отдельные требования. Если результат истории оказался уникальным, то вы должны преобразовать его в требование (просто скопировать его тело, используя повелительное наклонение). Если же похожий результат уже был вынесен в требование в рамках другой пользовательской истории, то надо проанализировать их возможные отличия и, если они есть — модернизировать существующее требование или создать дополнительное дочернее требование/ограничение. Приоритет требования должен быть унаследован от родительской истории с максимальным приоритетом.

Кроме результата пользовательской истории следует проанализировать её поток выполнения. Он может содержать уточнения/пожелания к поведению продукта, которые

не плохо было бы учесть. Пожеланиям следует назначать приоритет ниже, чем приоритет родительской истории.

В результате этой работы должен быть получен древовидный список требований. Каждое требование списка должно быть самостоятельным и не являться контейнером (структурной единицей для хранения дочерних элементов). Требования должны быть приоритезированы, причем дочернее требование не может иметь больший приоритет, чем родительское требование (которое оно дополняет).

Дробление требований

Для того чтобы иметь максимальную гибкость в процессе проектирования функциональности необходимо разбить требования на неделимые элементы.

Результатом этого процесса должен быть список требований, каждый элемент которого должен являться:

- Самостоятельным требованием — может расширять, а следовательно и зависеть от родительского требования, но не должно быть зависимо от дочерних требований или требований того же уровня.

  Это означает, что реализация требования не должна требовать реализации каких-либо дочерних требований или требований того же уровня. Если требования связаны настолько сильно, что могут быть реализованы только вместе — их нужно объединить.

- Неделимыми требованиями — в противоположность предыдущему критерию, требование не должно описывать сразу несколько проблем, которые можно решать порознь.

Благодаря выполненной работе, приобретается возможность запланировать реализацию наиболее значимых требований на начало разработки, а низкоприоритетных - на конец. Как результат, в случае отставания от графика выполнения в продукте уже будут реализованы наиболее приоритетные требования, и вы будете иметь возможность пожертвовать низкоприоритетными требованиями с целью сокращения отставания.

Экспертиза требований к дизайну

На этапе создания концепции продукта были описаны основные сценарии, в которых он должен использоваться, и на их основе был создан список требований к дизайну. Теперь на этапе сбора требований, кроме детализации требований к функционалу продукта (пользовательских историй), есть необходимость детализировать нефункциональные требования, проведя экспертизу требований к дизайну.

Этот процесс необходим в связи с тем, что текущие требования к дизайну были составлены в условиях сильного дефицита точной информации (присущему этапу создания концепции) и, как следствие, содержали множество предположений и неточностей. Взяв за основу пользовательские истории можно исправить большинство неточностей, а также детализировать существующие требования или добавить недостающие.

Экспертиза операционной среды продукта

Первым делом требуется определить все ли пользовательские истории могут быть удовлетворены требованиями к операционной среде продукта и не избыточен ли существующий список требований. Одновременно с этим, список системных требований должен быть заново приоритезирован. Для этого нужно произвести ревизию каждой пользовательской истории и выделить системные требования. Приоритет требования должен быть унаследован от пользовательской истории с максимальным приоритетом.

В результате ревизии у вас могут появиться новые системные требования, модифицироваться существующие или быть найдены лишние (требования в которых нет реальной необходимости). История всех изменений должна быть очень подробно описана, чтобы ею можно было воспользоваться на стадии утверждения требований.

Экспертиза нефункциональных требований

Список требований к дизайну, созданный на этапе создания концепции, уже содержит требования к надежности, производительности, обслуживанию и доступности системы. Теперь следует определить конкретные случаи, описанные в пользовательских историях, на которые распространяются указанные требования.

Каждое нефункциональное требование должно содержать метрики и параметры качества, которые должны быть достигнуты в каждой конкретной ситуации. Обычно, они оформлены, в виде правила, которое распространяется на большинство случаев. Правило может быть дополнено списком случаев, которые являются исключениями из этого правила и имеют свои собственные критерии качества.

Утверждение изменений требований к дизайну.

Требования к дизайну очень сильно влияют на архитектуру продукта, а необдуманные и частые изменения в архитектуре продукта могут сильно увеличить бюджет и сроки его реализации.

Потому в процессе принятия изменений требований к дизайну должны принимать все ответственные за продукт лица, а сам список изменений нужно постараться свести к минимуму.

Если невозможно реализовать отдельные требования в рамках текущей архитектуры, нужно просмотреть все пользовательские истории, которые эти требования затрагивают и определить можно ли их удовлетворить другим способом.

Список пользовательских историй, которые не будут реализованы в продукте (из-за ограничений архитектуры), должен быть рассмотрен и одобрен. Как правило, возможность удаления пользовательских историй напрямую зависит от их приоритета.

IV. Проектирование

Формирование детального описания будущего продукта — основная задача аналитика в процессе проектирования. Ранее, на этапах сбора и анализа определялись требования пользователей к системе — мотивы, по которым они будут её использовать. Теперь нужно определить, каким именно способом требования пользователей будут удовлетворены. Только с этого момента команда разработки продукта получает право принимать проектные решения — решать, какая конкретно функциональность будет реализована (Отвечать на вопрос — «что?»).

В процессе проектирования группой разработки продукта должно быть создано «Техническое задание» (Functional Specification), на основе которого будет производиться разработка и тестирование продукта. Это документ должен содержать следующие элементы:

- Требования к продукту уровня системы.
- Модель взаимодействия с пользователем:
    - Диаграммы вариантов использования продукта.
    - Потоки выполнения вариантов использования.
    - Ограничения интерфейсов.
    - GUI макеты.
    - CLI/ API спецификации.
- Архитектура продукта.
- Техническая информация.

На основе хорошего технического задания может быть определен исчерпывающий список работ и определены сроки его реализации. Чем выше качество технического задания, тем ниже риски и выше качество конечного продукта.

Определение функциональных требований к продукту уровня системы

В основу функциональных требований следует взять список функций продукта, который был определен на этапе создания концепции. Этот список является поверхностным и, возможно, содержит неточности, поэтому первым делом нужно уменьшить список неверных предположений, подкрепив все имеющиеся предположения фактами. Для этого понадобится выполнить исследования и/или привлечь специалистов предметной области.

Далее нужно произвести детализацию требований, описав **все**, что следует реализовать для удовлетворения пользовательских требований.

При описании требований важно абстрагироваться от пользовательского интерфейса, так как иначе вы сильно ущемите в возможностях ваших дизайнеров при его проектировании.

Функциональные требования рекомендуется структурировать в виде дерева, в котором дочерние требования расширяют функционал родительского. Кроме этого, можно создать диаграмму вариантов использования, на которой отобразить все взаимозависимости в продукте — это сильно облегчит процесс проектирования системы и сведет к минимуму ошибки в нем.

Каждое функциональное требование должно иметь родительское пользовательское требование и учитывать все требования к дизайну. Приоритет требований должен быть унаследован от родительского требования с максимальным приоритетом.

Работа может считаться законченной, когда удовлетворены все цели, содержащиеся в пользовательских историях (текущей итерации).

Ключевой фигурой в определении функциональных требований должен быть ведущий разработчик или менеджер проекта. Также, по возможности, стоит привлекать специалистов предметной области и конечных разработчиков, которые будут реализовывать функционал. Аналитик должен следить, чтобы в процессе обсуждения функционала, пользовательские требования трактовались без искажений, а предлагаемый функционал полностью им отвечал.

Как только список функций продукта определен, можно начинать работу по детализации архитектуры продукта, проводить исследования и назначать разработчикам задачи, не касающиеся пользовательского интерфейса.

Создание модели взаимодействия с пользователем

После того, как определен набор функций, который должен предоставлять продукт, нужно решить, каким образом пользователь сможет им воспользоваться. Для этого, на этапе создания диаграммы вариантов использования все функции продукта будут структурированы, и определен способ доступа к ним (прямой, расширение, включение). Далее, на этапе описания потоков выполнения будет смоделировано поведение продукта при взаимодействии пользователя с большим количеством функций и подразумевающим нелинейную реакцию системы (как правило, реализуются при помощи мастеров). В заключении должны быть определены непосредственные интерфейсы взаимодействия с системой; для GUI (графический пользовательский интерфейс) — это макеты пользовательского интерфейса; для CLI (Command Line Interface) и API — спецификация интерфейса.

Создание диаграммы вариантов использования продукта

Согласно законам когнитивной психологии, максимальное количество элементов, которое может эффективно восприниматься человеком, не превышает 7. При этом, даже очень маленький продукт, как правило, имеет более 10–20 функций. Поэтому, для того чтобы будущий продукт был удобен, его функции нужно должным образом структурировать. Структурирование проводится по общим признакам работы функций, по целям, которые пользователь намерен достичь или относительно контекста, в рамках которого про-

исходит выполнение функций. Часто, графический пользовательский интерфейс дублирует элементы доступа к функциональности, применяя к ним сразу несколько группировок (главное меню имеет группировку по признакам работы, выпадающее меню — контекстно-зависимую группировку, а мастера помогают в достижении отдельных целей).

Вторая проблема, которую нужно решить при проектировании пользовательского интерфейса — предоставить доступ к функционалу таким образом, чтобы пользователи смогли эффективно выполнить все последовательности операций, описанные в пользовательских историях.

Наиболее эффективным инструментом для решения вышеописанных проблем является создание диаграммы вариантов использования продукта. Если на этапе определения функциональных требований к продукту уже была создана диаграмма вариантов использования — то её можно взять за основу.

Основными элементами диаграммы вариантов использования должны служить функции системы. Очень удобным может оказаться цветовая маркировка частоты использования функций системы.

Описание сложных потоков выполнения

Удовлетворенность пользователя от работы с продуктом значительно зависит от простоты доступа к необходимому функционалу и легкости достижения конечного результата, описанного в пользовательских историях.

Действия, которые подразумевают взаимодействие системы с пользователем, называются потоками выполнения. Для описания сложных потоков, как правило, используются диаграммы активности, конечные автоматы, диаграммы последовательностей или их текстовые аналоги (нумерованные списки с условиями и ссылками).

Диаграммы активности (activity diagram) акцентируют внимание на совершаемых действиях и прекрасно подходят для описания алгоритма работы мастеров. В качестве активностей, как правило, используются страницы мастера. У активностей может быть несколько точек входа и выхода, каждую из которых нужно снабдить комментариями — для точки входа нужно указать предусловия, для точки выхода — постусловия.

Конечные автоматы (state machine) делают акцент на состояниях системы и возможных переходах между ними. Диаграммы этого типа удобно использовать при описании системы, которая имеет небольшое количество состояний и сложные условия перехода между этими состояниями.

Диаграммы последовательностей (sequences diagrams) акцентируют внимание на совершаемых действиях и объектах, которые их производят. В отличии от диаграмм активности они способны отлично отобразить параллельную работу сразу нескольких объектов для достижения общей цели. Их очень удобно использовать для иллюстрирования работы сложной среды, в которой задействуется множество компонентов.

Описание ограничений для интерфейсов

Для продуктов, предоставляющих сразу несколько интерфейсов, основными источниками ограничений, являются ограничения используемых технологий, ориентация разных интерфейсов на разные профили пользователей или желание сэкономить.

В качестве ограничений может быть заявлена недоступность отдельных функций продукта или только частичное покрытие требований к дизайну (поддержка только части ОС, невозможность работать удаленно, необходимость использования незаявленных языков разработки и др.). Все ограничения должны быть документированы и одобрены заинтересованными лицами еще до начала работ по созданию GUI макетов и CLI/API спецификаций.

Ограничение интерфейсов, как правило, включается в требования к дизайну (Design Requirements)

Создание GUI макетов

Практика показывает, что в процессе создания программного продукта, который имеет графический пользовательских интерфейс (GUI), каждое окно меняется или дорабатывается, по крайней мере — 3–4 раза (среднее кол-во пакетов изменений на продукт с 10 окнами — около 60) и чем позже эти изменения будут производиться, тем дороже обойдутся. Поэтому, для оптимизации затрат и уменьшения сроков разработки продукта, необходимо заниматься проектированием пользовательского интерфейса до его реализации в продукте.

Для проектирования GUI необходима среда разработки макетов. Со стороны GUI инженера основными требованиями к ней являются:

- Возможность использования уже имеющихся стандартных элементов пользовательского интерфейса (окна, компоненты) .
- Возможность быстрого создания новых элементов пользовательского интерфейса.
- Легкость наполнения компонентов «примерными» данными.
- Возможность быстрого внесения изменений в макет (за пару минут).
- Возможность сохранения макетов в доступном для чтения формате (PDF, HTML).

Для целевой аудитории основными требованиями к среде разработки макетов являются:

- Возможность делать комментарии к любым элементам макета.
- Возможность быстрого перехода к любой страничке интерфейса (без необходимости вводить начальные данные и др. действий, которые могут понадобиться в реальном продукте).
- Возможность демонстрации на проекторе.

- Возможность просмотреть все элементы интерфейса от начала до конца, вне зависимости от их взаимодействия между собой.

Как правило, очень велик соблазн выполнить всю работу в интегрированной среде разработки, чтобы потом воспользоваться уже готовыми моделями в самом продукте. Но уже на первых этапах это станет неудобным: на добавление случайных данных, нестандартных компонентов и компиляцию будет уходить много времени, простые изменения нельзя будет сделать прямо во время обсуждения нужного окна (а даже если можно, на возвращение к нему опять уйдет время) и т.д. В конечном итоге все те преимущества, которые дает ваша среда разработки, будут перечеркнуты недостатками. Поэтому, при разработке даже небольших продуктов, рекомендуется пользоваться специализированными программными пакетами для проектирования пользовательского интерфейса.
В качестве примера такого продукта можно привести совместное использование Microsoft Visio и плагина “Export to PDF”.

Обзор UI конкурентов

Вне зависимости от того, разрабатываете ли вы продукт под конкретного заказчика или для открытого рынка, обзор UI конкурентов будет очень важной фазой.

В отличие от обзора конкурентов, который делался на этапе разработки концепции, сейчас в фокус должны попасть детали реализации продуктов. Как правило, документ «Обзор конкурентов для GUI инженеров» имеет следующую структуру:

- Список конкурентов.
- Для каждого продукта:
    1. Список главных функций продукта
    2. Артефакты продукта (сущности, которыми манипулирует пользователь).
    3. Основные потоки выполнению + снимки экранов. Также может содержать CLI/API синтаксис функций.
    4. Резюме: хорошие идеи, которые надо использовать и ошибки, которых нужно избежать.

Для GUI инженеров этот документ является главным источником информации, из него они смогут подчеркнуть хорошие идеи, отметить плохие решения (которые желательно не повторять) и лучше понять предметную область продукта. В результате, есть шанс, что вы не создадите очередной «велосипед» десятилетней давности, а если и сделаете это, то абсолютно осознано и в кротчайший срок.

Проектирование CLI/API спецификаций

Хорошая спецификация к CLI/API создается так, чтобы её можно было использовать и в качестве сопроводительной документации к продукту. Следовательно, пользователями этого документа являются:

- Разработчики, которые должны реализовать нужную функциональность.
- Отдел тестирования, который должен её проверить .

- Конечные пользователи, которые будут пользоваться продуктом (после того, как он пройдет через технического писателя).

Как и любой другой интерфейс приложения, CLI или API должны быть спроектированы на базе пользовательских историй с учетом ограничений для данного интерфейса. Также, перед структурированием функций, должен быть определен стиль интерфейса — базовый синтаксис, способ передачи параметров, модель и др.

Структурирование функций обычно выполняется либо архитектором компании, либо отделом пользовательских интерфейсов (если они не специализируются только на GUI).

Обычно спецификация CLI/API состоит из:

- Список функций продукта.
- Описания каждой функции:
    1. Синтаксис
    2. Текстовое описание
    3. Описание всех параметров с перечислением всех возможных значений
    4. Результат, если имеется
    5. Пример использования функции

Несмотря на то, что автор находит много ошибок уже на стадии написания примеров, редко удается опубликовать CLI или API спецификацию без ошибок, поэтому следует, как можно раньше, отдавать этот документ на тестирование отделу QA, чтобы отдел разработки получил его с минимальным количеством ошибок.

Описание архитектуры продукта

Архитектура продукта — это описание базовых компонентов и способа их взаимодействия между собой. Определением архитектуры должны заниматься наиболее опытные члены команды разработки (архитекторы, лидеры).

На этапе создания образа продукта (в рамках определения архитектуры высокого уровня) уже должна была быть выбрана модель выполнения продукта и технологии, которые планировалось использовать. Теперь, зная детальные требования к продукту, команде архитекторов нужно убедиться, что их первоначальный выбор был правильный, и в случае несоответствий, принять решение о корректировке архитектуры или требований (введение ограничений). Всегда следует находить баланс между приемлемостью вводимых ограничений и последствиями (сроками/стоимостью) её корректировки.

Львиная доля работы архитекторов сводится к описанию внешних (а в крупных продуктах и внутренних) интерфейсов. Их задача — всегда видеть весь проект (или даже группу проектов) целиком, обеспечивая его целостность на высоком уровне и предотвращая написания «велосипедов».

Таким образом, в структуру описания архитектуры продукта, как правило, входят:

- **Модель развертывания продукта** (Deployment diagram) — диаграмма, содержащая информацию о развертывании компонентов продукта и способе их взаимодействия.
- **Описание низкоуровневых компонентов продукта —** диаграмма взаимодействия всех компонентов продукта, которые нужно будет реализовать/включить в продукт.
- **Описание интерфейсов продукта** в виде диаграмм или на языке программирования.
- **Потоки выполнения** наиболее сложных процессов.
- **Базовая структура баз данных**, при их наличии.

Добавление технической информации

Часто функциональные и нефункциональные требования требуют дополнительного пояснения — информации о предметной области или деталях реализации. Подобная информация чаще всего дополняет сразу несколько требований или даже проектов, а потому удобно держать её в отдельном хранилище, доступном аналитикам, разработчикам, инженерам контроля качества, службы поддержки, а также продавцам. Самой удобной средой для этого является Wiki (локально установленная в компании). В случае использования системы управления требованиями, основанной на БД, вы потеряете целостность, вынеся часть данных за её пределы (не сможете отслеживать изменения зависимых элементов), но взамен вы получите очень простой для поддержки и наполнения инструмент, доступный всем сотрудникам.

Постановка задач по системным требованиям

Функциональные требования должны точно описывать, **что** должно быть реализовано в продукте. С точки зрения технического задания — функциональные требования являются результатом выполнения задач, поэтому каждое функциональное требование можно представить в качестве задачи (также можно объединять сразу несколько требований в одну задачу, или разделить одно требование на несколько задач).

При этом функциональные требования не имеют заточки под конкретного исполнителя. Более того, техническое задание может создаваться, не имея даже представления о команде, которая его будет реализовывать (аутсорсинг). В результате, в нем полностью отсутствует ответ на вопрос — «**как** реализовать нужную функциональность». В этом случае, хорошей практикой является первоначальное определение исполнителя задачи с последующим добавлением необходимой информации в соответствии с компетентностью исполнителя.

Иерархия компетентности исполнителей на примере разработчиков:

Ведущий разработчик (Lead Developer) — может делегировать выполнение задачи своим подчиненным, при этом сам несет ответственность за детализацию технического задания. Для человека, играющего роль менеджера

проектов это означает, что функциональных требований достаточно для постановки задачи ведущему разработчуку.

Старший разработчик (Senior Developer) — максимально компетентен в своей области. Для постановки задачи для него также достаточно функциональных требований.

Разработчик (Developer) — имеет среднюю компетентность. Для него рекомендуется дополнять функциональные требования, пожеланиями или рекомендациями.

Младший разработчик (Junior developer) — имеет низкую компетенцию. Для него рекомендуется описывать алгоритм реализации с наибольшей степенью детализации.

Общая структура требований

Структура требований при использовании средств ориентированных на документ

Маркетинговые требования (Product Marketing Requirements, также PVD — Product Vision Document) *— Отдельный документ:*

- Бизнес требования (Business Requirements):
  - Стимулы (Product Stimuli).
  - Целевой рынок (Target Marketing Segment) — для MRD.
  - Сценарии использования и проблемы (Scenarios and problems).
  - Обзор конкурентов (Competitors Review) — для MRD, как правило, оформлен как отдельный документ.
- Решение (Solution):
  - Образ продукта (Product Vision).
  - Список возможностей (Features List).
- Суть проекта (Project Scopes) — мероприятия по созданию следующей версии продукта).
  - Возможности продукта (Product Features).
  - Вехи продукта (Product Milestones).

Требования к продукту (Product Requirements Document) *— Множество документов, каждый из которых описывает необходимую функциональность:*

- Профили пользователей (User profiles).
- Пользовательские истории (User Stories).
- Варианты использования (Use cases) — систематизированные пользовательские истории.
- Пользовательские требования (User requirements) — разделяются на функциональные и нефункциональные.

Функциональная спецификация (Functional Specification) *— Множество документов, каждый из которых описывает необходимую функциональность:*

- Системные требования (System requirements).
- Требования проектирования (Design Requirements) — Архитектура продукта (Product Architecture).

- Ограничения интерфейсов (Interfaces scopes) — часто является расширением требований к проектированию.
- Модель взаимодействия (Interaction Model):
  - Варианты использования (Use case diagram).
  - Активность (Activities).
  - Макеты графического пользовательского интерфейса (GUI mock-ups).
  - Спецификация интерфейса командной строки (CLI specification).
  - Спецификация интерфейса прикладного программирования (API specification).
- Техническая информация (Technical Notes).

Структура требований при использовании баз данных

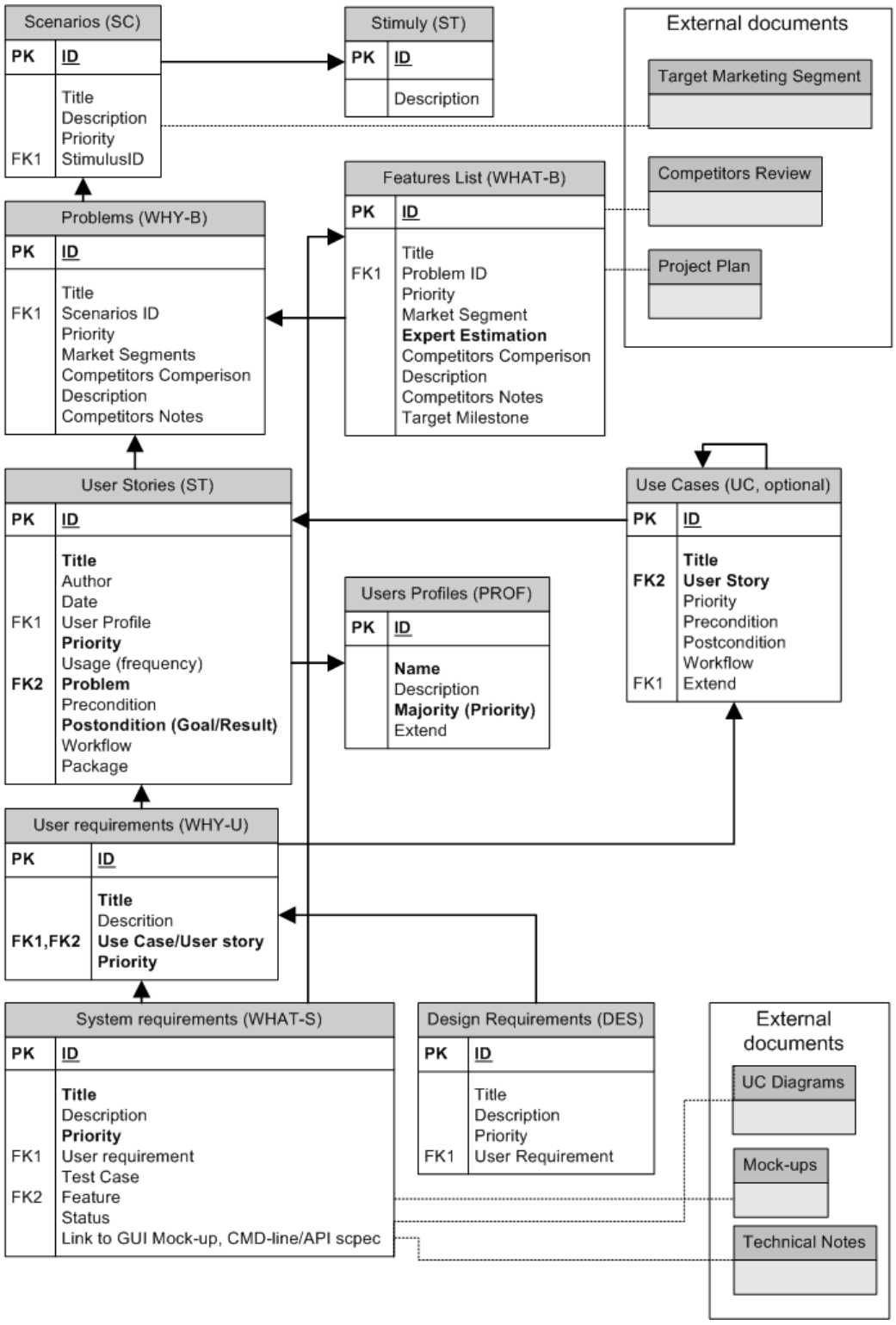

Рисунок 9. Артефакты жизненного цикла разработки продукта.

План работ по сбору и анализу требований к продукту

В данном примере приведен проект общей продолжительностью 9 месяцев. На создание концепции в нем отводится 1 месяц (3 недели на определение концепции и 1 неделя на её утверждение и доработку в случае необходимости). Еще 2 месяца отводится на процесс доведения продукта до бета-качества и проведения бета-тестирования (по месяцу на каждую стадию).

Итого на разработку остается 6 месяцев: 4 итерации по 1,5 месяца.

Для того чтобы процесс разработки был наиболее эффективным, в начале новой итерации техническое задание должно быть уже готово. Для того чтобы этого достичь, процессы сбора и анализа требований должны начинаться на 2–4 недели раньше, а проектирование должно производиться либо в самом начале итерации либо на стадии исправления ошибок в предыдущей итерации.

Таблица 7. Определение концепции продукта

Имя задачи	Исполнитель	Длит. (дни)
Определение бизнес требований		8.5
Определение стимулов	Analyst & Stakeholders	0.5
Определение целей продукта и критериев успеха	Analyst & Stakeholders	0.5
Определение целевого сегмента	Analyst & Stakeholders	0.5
Определение потребностей клиентов или рынка		3
Создание сценариев	*Analyst & Stakeholders*	*2*
Определение требований	*Analyst*	*1*
Обзор конкурентов	Analyst	4
Создание образа решения		2.5
Определение возможностей продукта	Project Team	2
Определение образа продукта	Project Team	0.5
Определение содержания проекта		3
Определение объема первоначальной версии продукта	Project Team	1
Определение контрольных событий проекта	Project Team	2
Всего		**14**

Таблица 8. Сбор и анализ требований

Имя задачи	Исполнитель	Длит. (дни)
Определение основных профилей пользователей (выполняется только в первой итерации)	Analyst	1
Формирование инициативной группы (выполняется только в первой итерации)	Analyst	2
Сбор пользовательских историй	Analyst & IG	5
Анализ требований (проектирование на основе вариантов использования)		2.5
Выделение пользовательских историй в пакеты	Analyst	0.5

Поиск базовых вариантов использования	Analyst	0.5
Составление диаграммы вариантов использования	Analyst	1
Преобразование историй в пользовательские требования	Analyst	0.5
Анализ требований (проектирование на основе требований)		1.5
Выделение пользовательских историй в пакеты	Analyst	0.5
Извлечение требование из пользовательских историй	Analyst	0.5
Дробление требований	Analyst	0.5
Экспертиза требований к дизайну		4
Экспертиза операционной среды продукта	Analyst	1
Экспертиза нефункциональных требований	Analyst	1
Утверждение новых требований к дизайну	Project Team	2
Всего		**14.5**

Таблица 9. Проектирование

Имя задачи	Исполнитель	Длит. (дни)
Определение системных требований	FG	5
Разработка/обновление высокоуровневой архитектуры	Architect	5
UI Development (GUI)	GUI engineer	10
CLI/API spec	Architect	3
Всего		**11**

Запланированные дополнения и изменения к статье

В случае интереса к статье со стороны читателей автор планирует обновлять статью раз в несколько месяцев в соответствии с планом ниже:

- Более тесная интеграция с IEEE 1074 — трансляция процессов, описанных в статье на стандарт.
- Применение SWAT анализа в процессе определения образа продукта.
- Добавление примеров.
- Описание структуры и процессов создания документов, необходимых маркетингу (DataSheet, Product Benefits, Release Notes).
- Описание сбора пользовательских историй с применением сторонних специалистов.

Примечание

Автор с благодарностью примет любые исправления и комментарии, относящиеся к данной статье. Пожалуйста, присылайте Ваши сообщения по электронной почте: hiex@mail.ru.