CASE технологии

Лекция 1

Предмет курса

- Предназначение CASE
- Виды CASE технологий
- Языки моделирования в CASE технологиях
- Виды методологий проектирования программных систем и их реализация в CASE технологиях

Структура курса

- Модуль 1. Моделирование бизнес-процессов предприятия. Методология SADT / IDEF (IDEF0, IDEF3, DFD)

- Модуль 2. Моделирование в UML 2.

Предназначение CASE

- Характеристики современных крупных проектов разработки ИС:
  - сложность описания (масштаб, детали)
  - много компонентов - много взаимодействий
  - отсутствие прямых аналогов
  - интеграция новых и унаследованных систем
  - несколько аппаратных платформ
  - несколько групп разработчиков – разных стилей
  - длительное время разработки

Сложность описания проекта

- Модель больших размеров
- Модель должна быть непротиворечивой и адекватной
- Требования пользователей могут уточняться/изменяться

Сложность описания проекта

- Есть графические методики (SADT/IDEF, UML, другие), но!

Модель на бумаге +

реализация в коде

= хорошо и возможно для малых проектов

- Для больших проектов - чаще «трата времени/ людских ресурсов» на документацию

Сложность описания проекта

- Если модель «на бумаге»
  - как проверить ее на противоречия?
    - закодировать, исправить ошибки…
  - Как изменить модель по просьбе заказчика?
    - Перерисовать на бумаге, закодировать, исправить ошибки…
- такая модель - «для красоты», не для работы

Резюме: модель «на бумаге»

- неадекватная спецификация требований
- неспособность обнаруживать ошибки в проектных решениях
- низкое качество документации, снижающее эксплуатационные качества
- затяжной цикл и неудовлетворительные результаты тестирования

Определение CASE

CASE-средства - это особый класс программно-технологических систем, реализующих CASE-технологию создания и сопровождения ИС

CASE-технология – это **методология** проектирования ИС, и **набор инструментальных** средств для:

- **наглядного моделирования** предметной области
- **анализа модели** на всех этапах разработки и сопровождения ИС
- **разработки приложения** в соответствии с информационными потребностями пользователей

Аббревиатура CASE

- CASE - Computer Aided Software Engineering

  Первоначальное значение термина CASE ограничивалось вопросами автоматизации разработки только лишь программного обеспечения (ПО)

Определение CASE

- CASE-средства - это программные средства, поддерживающие процессы создания и сопровождения ИС, включая **анализ и формулировку требований, проектирование прикладного ПО** (приложений) и **баз данных**, **генерацию кода**, **тестирование, документирование, обеспечение качества**, **конфигурационное** управление и управление **проектом**, а также другие процессы.
- CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС

Факторы, повлиявшие на распространение CASE

- **подготовка аналитиков**, программистов, готовых к использованию методологий проектирования
- Рост **производительности** компьютеров (возможность использования графики)
- **Сетевые** технологии

Статистика

На 1996 год:

1000 фирм, специализирующихся на разработке ПО

- Из них около 500 использовали CASE
- Более чем в 1/3 проектов
- Из которых 85% окончилось успешно

Особенности использования CASE

- CASE-средства **не обязательно дают немедленный эффект**; он может быть получен только спустя какое-то время;
- реальные затраты на внедрение CASE-средств обычно **намного** превышают затраты на их приобретение;
- CASE-средства обеспечивают возможности для получения существенной выгоды **только после успешного завершения процесса их внедрения**

Успешное внедрение CASE

- высокий уровень технологической поддержки процессов разработки и сопровождения ПО
- положительное воздействие на некоторые или все из перечисленных факторов:
  - производительность,
  - качество продукции,
  - соблюдение стандартов,
  - документирование;
- приемлемый уровень отдачи от инвестиций в CASE-средства.

Два аспекта создания ИС

- Первый: как, сквозь какие фильтры рассматривается задача? Что видит разработчик в задаче? Какова структура задачи?

  - Проектирование

- Второй (когда задача уже понятна): как разработчик будет решать задачу? Каков будет процесс решения?

  - Разработка

Методологии проектирования

- Структурный анализ и проектирование
- Объектно-ориентированный анализ и проектирование

Методологии для первого аспекта

Используются спецификации в виде диаграмм или текстов

Методологии разработки ИС

- Базовое понятие: жизненный цикл (ЖЦ) программного обеспечения (ПО)

- Нормативная база: международный стандарт ISO/IEC 12207

Методологии для второго аспекта

Процессы ЖЦ (ISO/IEC 12207)

Всего 7 групп процессов:

- процессы *соглашения* — 2;
- процессы *организационного обеспечения проекта* — 5;
- процессы *проекта* — 7;
- *технические* процессы — 11;
- процессы *реализации* программных средств — 7;
- процессы *поддержки* программных средств — 8;
- процессы *повторного применения* программных средств — 3

Процессы соглашения

- Поставка
- Приобретение

Процессы организационного обеспечения проекта

- Процесс менеджмента модели жизненного цикла;
- Процесс менеджмента инфраструктуры;
- Процесс менеджмента портфеля проектов;
- Процесс менеджмента людских ресурсов;
- Процесс менеджмента качества.

Процессы проекта

- Процессы менеджмента проекта
  - процесс планирования проекта;
  - процесс управления и оценки проекта.
- Процессы поддержки проекта
  - процесс менеджмента решений;
  - процесс менеджмента рисков;
  - процесс менеджмента конфигурации;
  - процесс менеджмента информации;
  - процесс измерений.

Технические процессы

- Определение требований правообладателей
- Анализ системных требований
- Проектирование архитектуры системы
- Процесс реализации
- Процесс комплексирования системы
- Процесс квалификационного тестирования системы
- Процесс инсталляции программных средств
- Процесс поддержки приемки программных средств
- Процесс функционирования программных средств
- Процесс сопровождения программных средств
- Процесс изъятия из обращения программных средств

Процессы реализации программных средств

- Процесс анализа требований к программным средствам;
- Процесс проектирования архитектуры программных средств;
- Процесс детального проектирования программных средств;
- Процесс конструирования программных средств;
- Процесс комплексирования программных средств;
- Процесс квалификационного тестирования программных средств

Процессы поддержки программных средств

- Процесс менеджмента документации программных средств;
- Процесс менеджмента конфигурации программных средств;
- Процесс обеспечения гарантии качества программных средств;
- Процесс верификации программных средств;
- Процесс валидации программных средств;
- Процесс ревизии программных средств;
- Процесс аудита программных средств;
- Процесс решения проблем в программных средствах.

Процессы повторного применения программных средств

- Процесс проектирования доменов;
- Процесс менеджмента повторного применения активов;
- Процесс менеджмента повторного применения программ.

Этапы ЖЦ: Разработка

- работы по созданию ПО и его компонент в соответствии с заданными требованиями, включая:
  - оформление проектной и эксплуатационной документации,
  - подготовку материалов (тестов), необходимых для проверки работоспособности и соответствующего качества программных продуктов,
  - материалов, необходимых для организации обучения персонала и т.д.

  Разработка ПО включает в себя, как правило, анализ, проектирование и реализацию (программирование).

Этапы ЖЦ: Эксплуатация

- Эксплуатация включает в себя работы:
  - по внедрению компонентов ПО в эксплуатацию, в том числе конфигурирование базы данных и рабочих мест пользователей,
  - обеспечение эксплуатационной документацией,
  - проведение обучения персонала и т.д.,

  И, непосредственно, эксплуатацию, в том числе
  - локализацию проблем и устранение причин их возникновения,
  - модификацию ПО в рамках установленного регламента,
  - подготовку предложений по совершенствованию, развитию и модернизации системы

Этапы ЖЦ: Управление проектом

- Планирование и организация работ
  - выбор **методов** и инструментальных **средств** для реализации проекта,
  - определение методов описания **промежуточных состояний разработки**,
  - разработку **методов и средств испытаний ПО**,
  - обучение персонала и т.п.
- Создание коллективов разработчиков
- Контроль за сроками и качеством выполнения работ

Этапы ЖЦ: Управление проектом: качество

Обеспечение **качества** проекта связано с проблемами верификации, проверки и тестирования ПО.

Верификация - это процесс определения того, отвечает ли текущее состояние разработки, достигнутое на данном этапе, требованиям этого этапа.

Проверка позволяет оценить соответствие параметров разработки с исходными требованиями.

Проверка частично совпадает с **тестированием**, которое связано с **идентификацией различий между действительными и ожидаемыми результатами** и оценкой соответствия характеристик ПО исходным требованиям.

Этапы ЖЦ: Управление проектом: конфигурирование

В процессе реализации проекта важное место занимают вопросы идентификации, описания и контроля **конфигурации** отдельных компонентов и всей системы в целом

Модели процесса разработки ПО

- В стандарте нет жесткого описания моделей ЖЦ ПО
- Стандарт ISO/IEC 12207 описывает структуру процессов ЖЦ, но не конкретизирует деталей

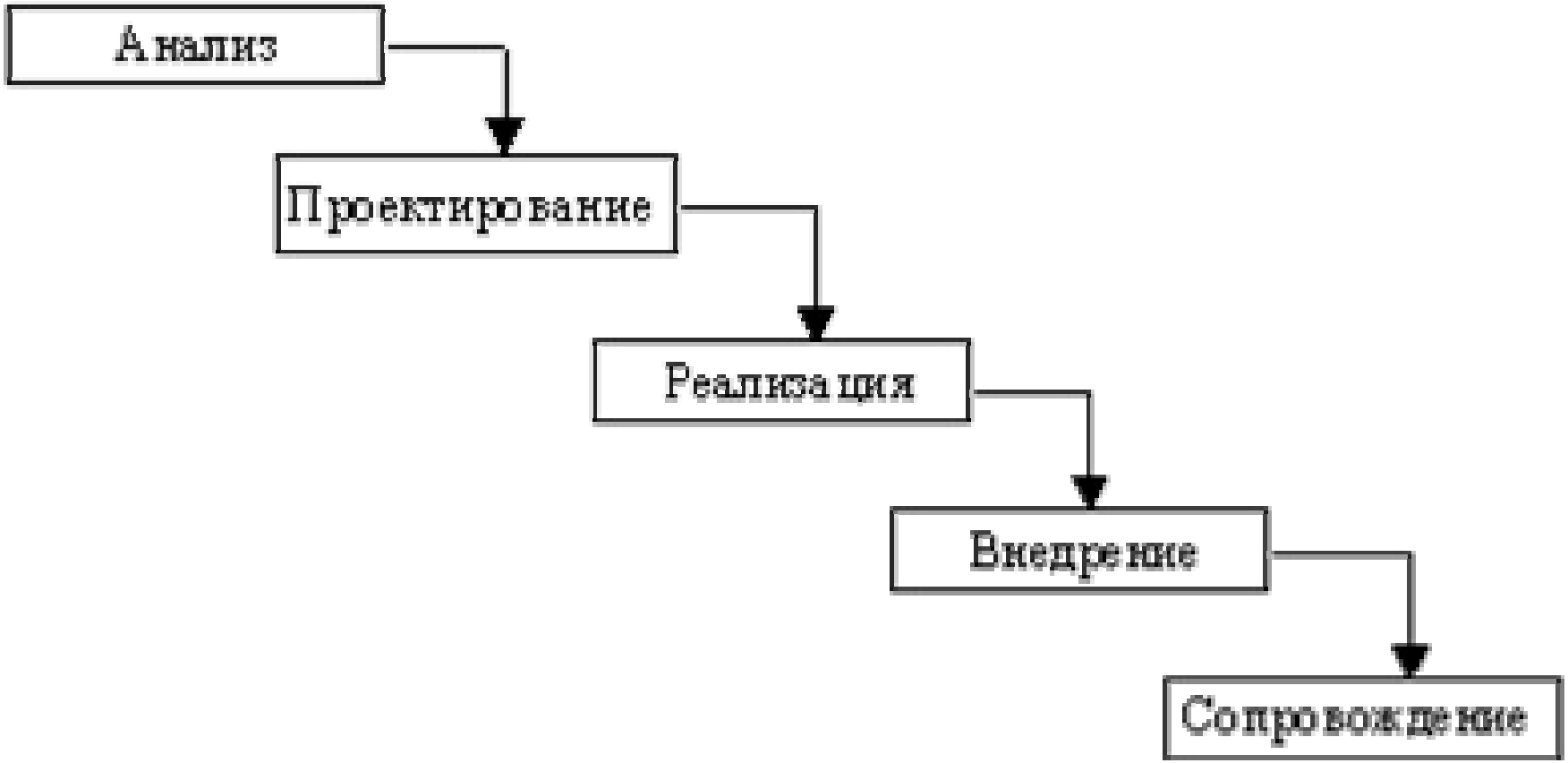
Модели процесса разработки ПО

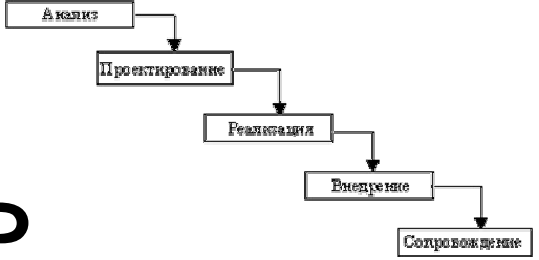
- каскадная модель (70-80-е г.г.)
- спиральная модель (конец 80х г.г.):
  - Методология RAD (Rapid Application Development)
- итеративная модель (90-е г.г.)
  - Методология RUP (Rational Unified Process)
- V (VEE) модель (конец 80х г.г.)
- Dual VEE модель (90-е г.г.)

Каскадная модель (waterfall, водопад)

- разбиение всей разработки на этапы, причем переход с одного этапа на следующий происходит только после того, как будет полностью завершена работа на текущем.
- Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков

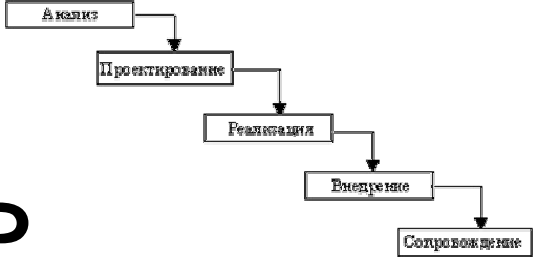
Каскадная модель





Каскадная модель

- Достоинства:
  - на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
  - выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты

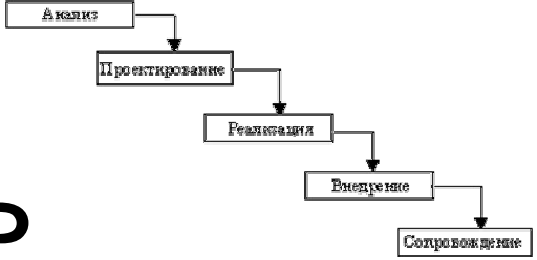


Каскадная модель

- Применение:

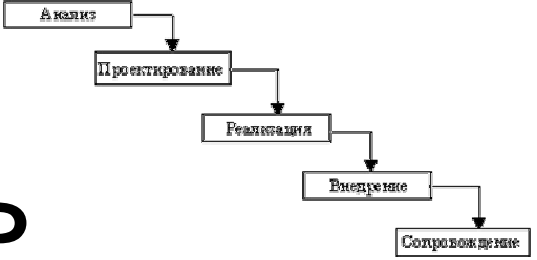
ИС, для которых в самом начале разработки можно достаточно точно и полно сформулировать все требования, с тем чтобы предоставить разработчикам свободу реализовать их как можно лучше с технической точки зрения.

В эту категорию попадают сложные расчетные системы, системы реального времени и другие подобные задачи



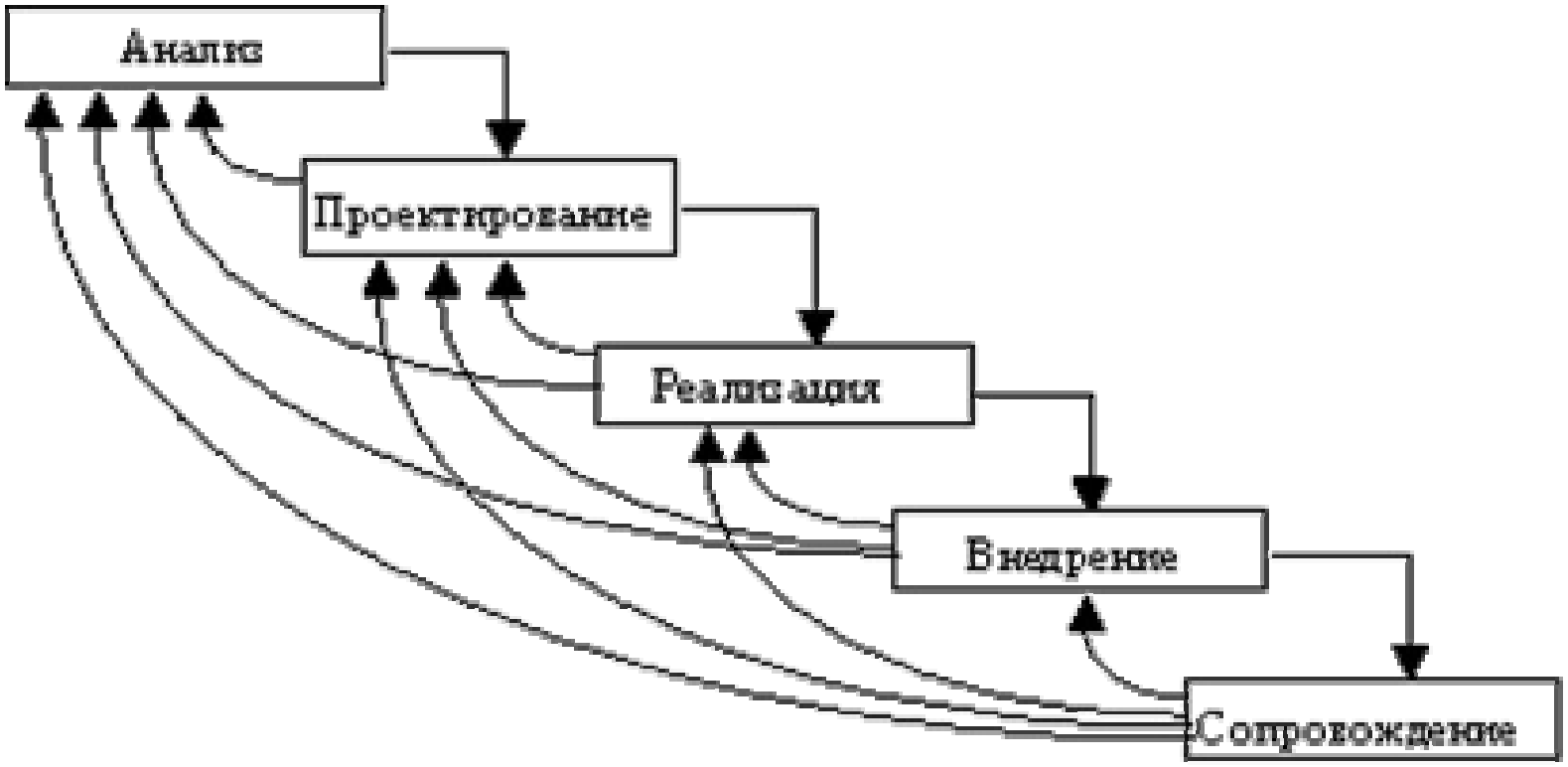
Каскадная модель

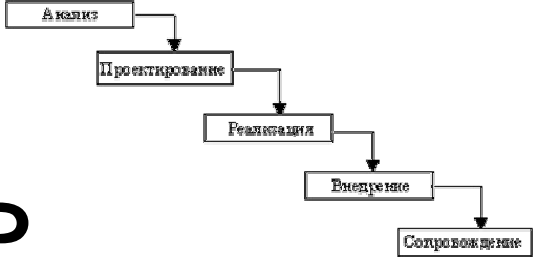
- Недостатки:
  - реальный процесс создания ПО никогда полностью не укладывался в такую жесткую схему. В процессе создания ПО постоянно возникала потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений.



Каскадная модель

- Недостатки:
  - Реальный процесс разработки выглядит так:

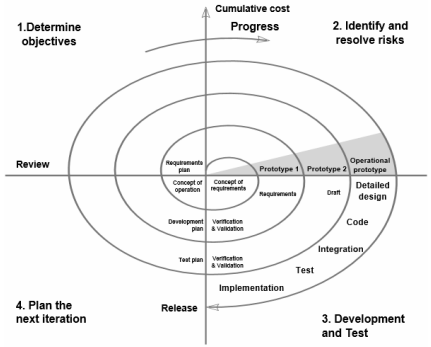




Каскадная модель

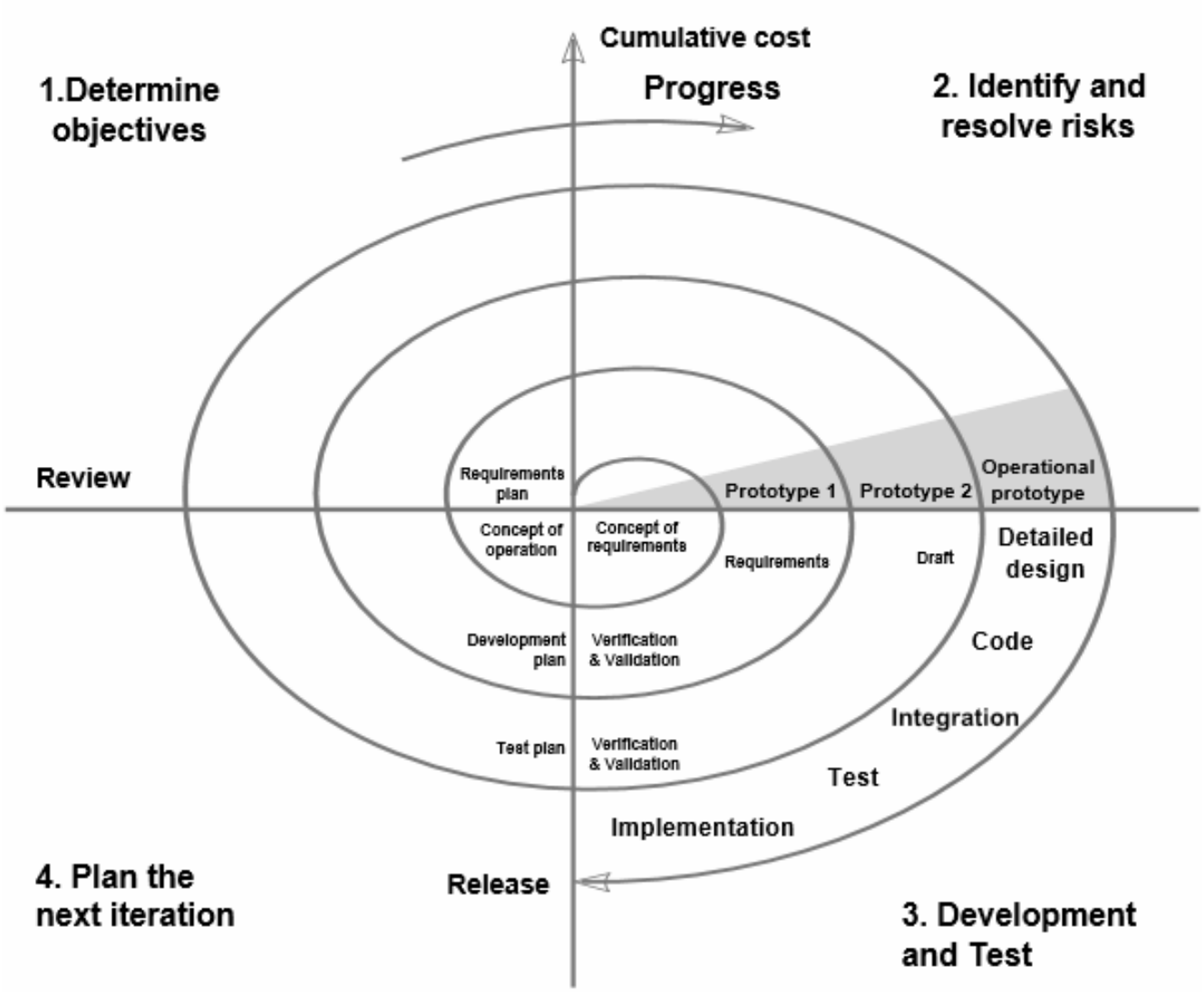
- Недостатки:
  - существенное запаздывание с получением результатов
  - Согласование результатов с пользователями производится только в точках, планируемых после завершения каждого этапа работ
  - Требования к ИС "заморожены" в виде технического задания на все время ее создания.
- Замечания можно внести только полного завершения работ
- Неточности в требованиях ведут к непригодной системе
- Модели (как функциональные, так и информационные) автоматизируемого объекта могут устареть одновременно с их утверждением.

Спиральная модель

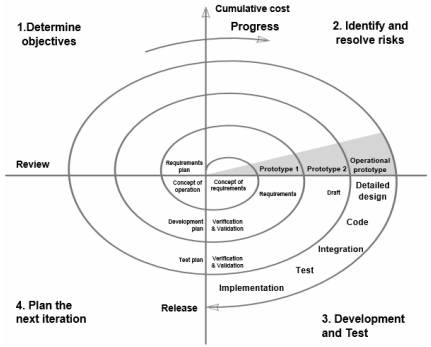


- упор на начальные этапы ЖЦ: **анализ** и **проектирование**
- реализуемость технических решений проверяется путем создания **прототипов**

Спиральная модель

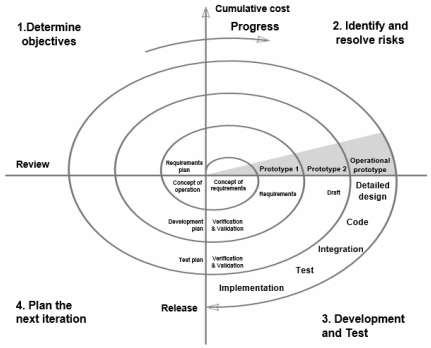


Спиральная модель



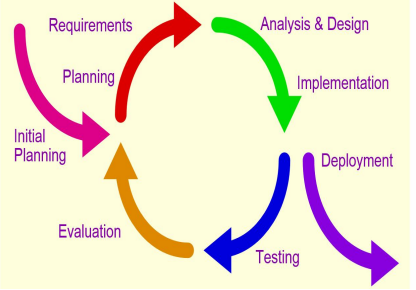
- Достоинства:
  - неполное завершение работ на каждом этапе позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем
  - Итеративность работ - недостающую работу можно будет выполнить на следующей итерации. Главная же задача - как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований

Спиральная модель



- Недостатки:
  - Сложность в определении момента перехода на следующий этап - применяются временные ограничения на каждый из этапов жизненного цикла Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена.
  - План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков

Итеративная модель



- выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы.
- Проект в каждой фазе развития проходит повторяющийся цикл **PDCA** (*plan-do-check-act cycle*):

Планирование →

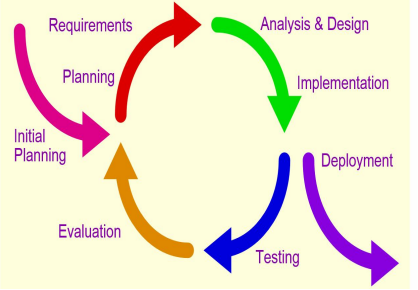
Реализация →

Проверка →

Оценка

Итеративная модель

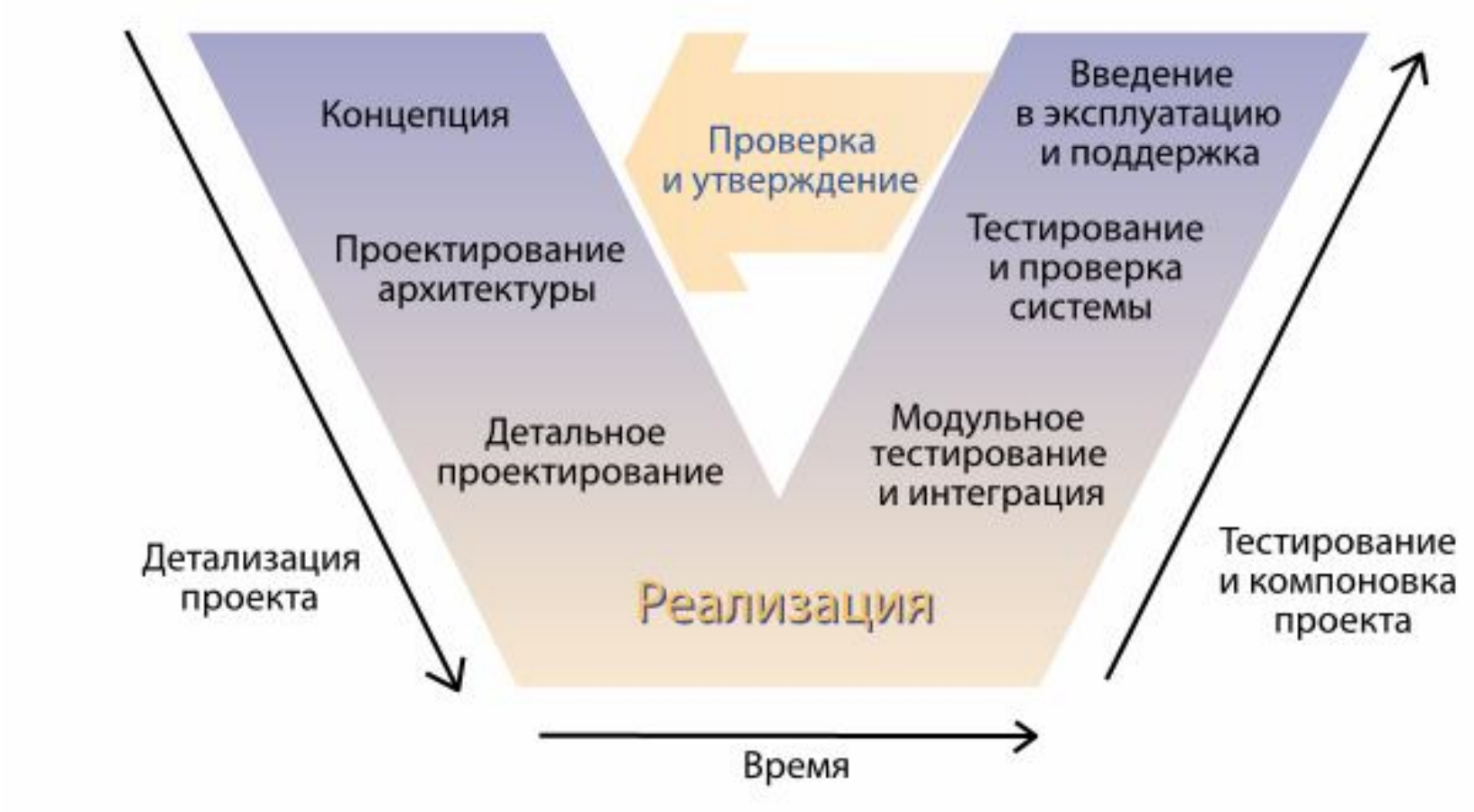
Итеративная модель



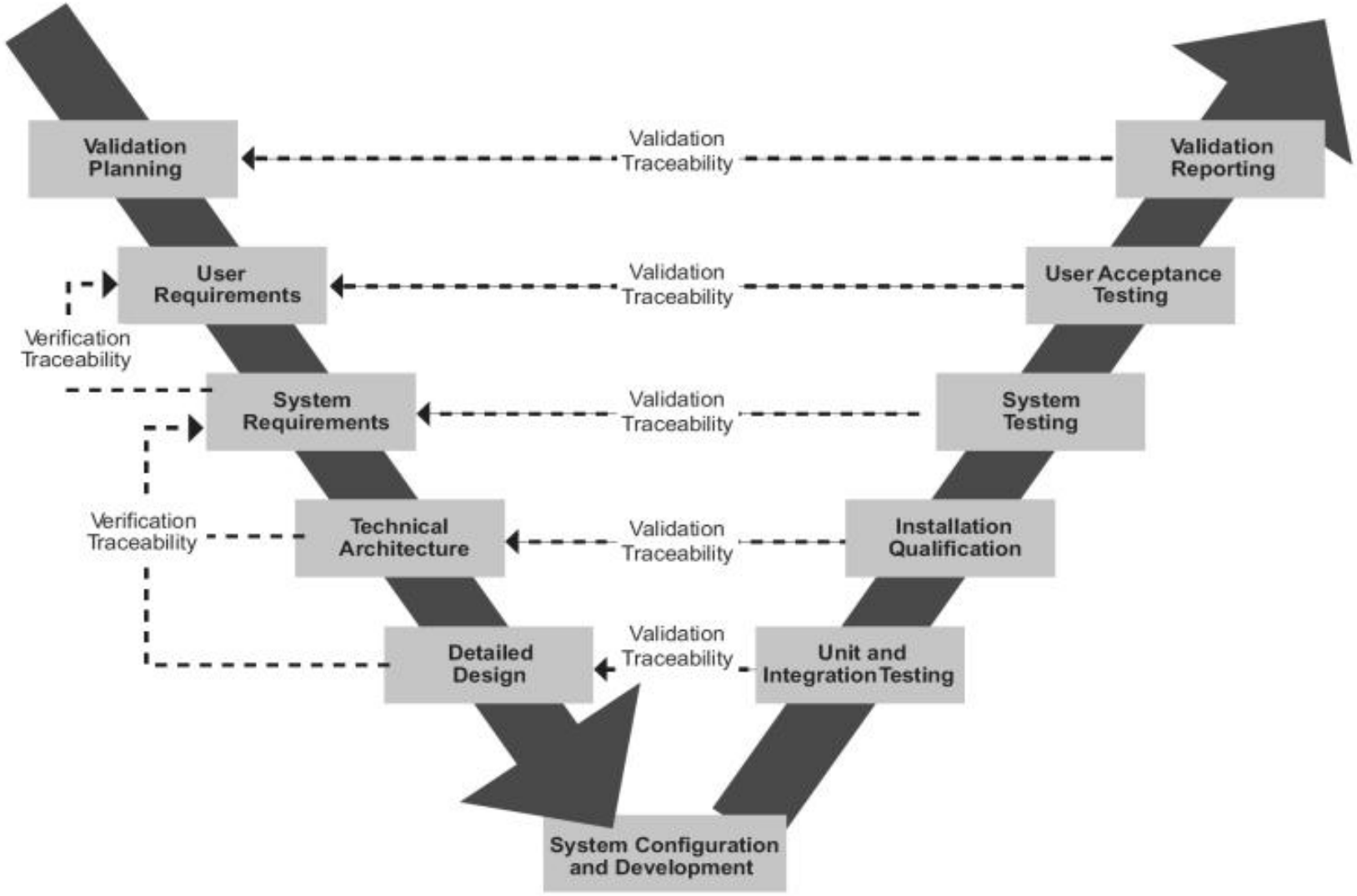
Достоинства

- Ошибки на ранних стадиях проекта не так страшны;
- Эффективная обратная связь проектной команды с заказчиками;
- Акцент усилий на наиболее важные и критичные направления проекта;
- Непрерывное итеративное тестирование, позволяющее оценить успешность всего проекта в целом;
- Раннее обнаружение конфликтов между требованиями, моделями и реализацией проекта;
- Более равномерная загрузка участников проекта;
- Эффективное использование накопленного опыта;
- Реальная оценка текущего состояния проекта и, как следствие, большая уверенность заказчиков и непосредственных участников в его успешном завершении;
- Затраты распределяются по всему проекту, а не группируются в его конце

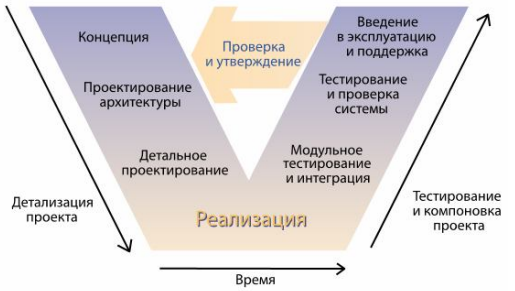
VEE модель


Концепция
Проектирование архитектуры
Детальное проектирование
Проверка и утверждение
Введение в эксплуатацию и поддержка
Тестирование и проверка системы
Модульное тестирование и интеграция
Детализация проекта
Реализация
Тестирование и компоновка проекта
Время

VEE модель детально



VEE модель



Вариант **каскадной** модели,в которой

задачи разработки идут сверху вниз по левой стороне буквы V,

задачи тестирования — вверх по правой стороне буквы V.

Внутри V проводятся горизонтальные линии, показывающие, как результаты каждой из фаз разработки влияют на развитие системы тестирования на каждой из фаз тестирования.

VEE модель

Основные принципы VEE модели:

1) детализация проекта возрастает при движении слева направо, одновременно с течением времени.

2) итерации в проекте производятся по горизонтали, между левой и правой сторонами буквы.

VEE модель

Достоинства:

- Особое значение придается планированию, направленному на верификацию и аттестацию разрабатываемого продукта на ранних стадиях его разработки.
- аттестация и верификация всех внешних и внутренних полученных данных, а не только программ
- Сначала - определение требований, потом - разработка проекта системы

VEE модель

Недостатки

- внесение требования динамических изменений на разных этапах жизненного цикла не предусмотрено
- Тестирование требований в жизненном цикле происходит слишком поздно
- Нет анализа рисков

Dual VEE модель

Многократное использование V-модели для разработки системы и подсистем ПО

Резюме: что узнали на лекции?

- Что такое CASE-технологии
- Для чего нужны CASE-средства
- Наиболее известные методологии проектирования: структурная и ООП
- Познакомились с каскадной, спиральной, итеративной, V, Dual VEE моделями процесса разработки ПО
- Выяснили их достоинства и недостатки