Учебный курс

Модели жизненного цикла и методологии разработки корпоративных систем

Лекция 3

Модели жизненного цикла корпоративных систем

Лекции читает

кандидат технических наук, доцент

Зыков Сергей Викторович

Содержание

Модели ЖЦ ПО: Инкремент(аль)ная

- **Особенности**:
  - Разбивка ПО на последовательные **релизы** (каждый цикл разработки дает работоспособный продукт)
- **Преимущества:**
  - **Работающий продукт на каждом шаге разработки**
  - Плавный ввод новой функциональности у клиента
  - Легкость сопровождения за счет «прямолинейного расширения» основных модулей

Модели ЖЦ ПО: Инкрементная

- **Недостатки:**
  - Требует **наращиваемого программного решения**

(не годится для ПО, требующего сразу полной функциональности)

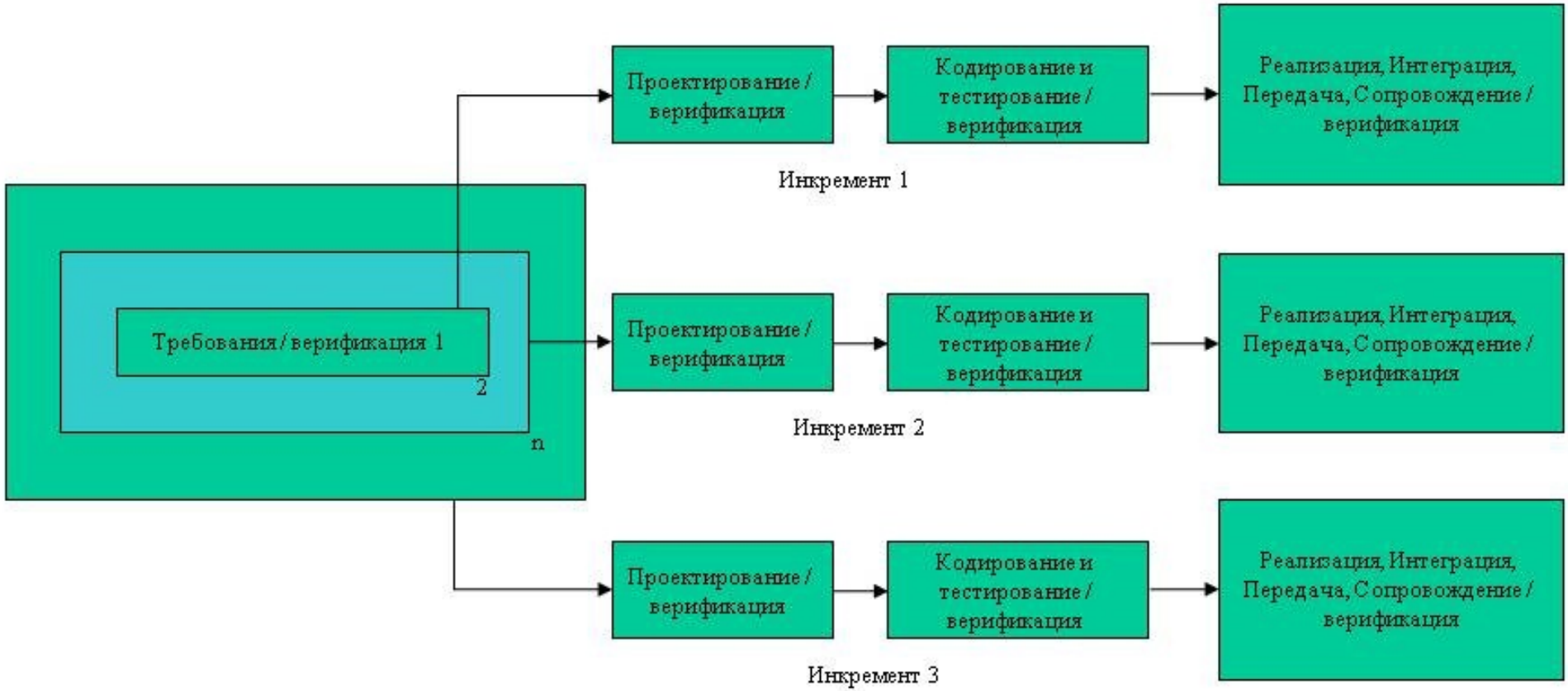
  - Продукт должен «масштабироваться» по архитектуре
  - ПО должно предусматривать **стабильный путь апгрейда**
  - **Не подходит для продуктов, которые быстро выходят**

за рамки исходной концепции (при этом вырождается в build-and-fix)

Модели ЖЦ ПО: Инкремент(аль)ная

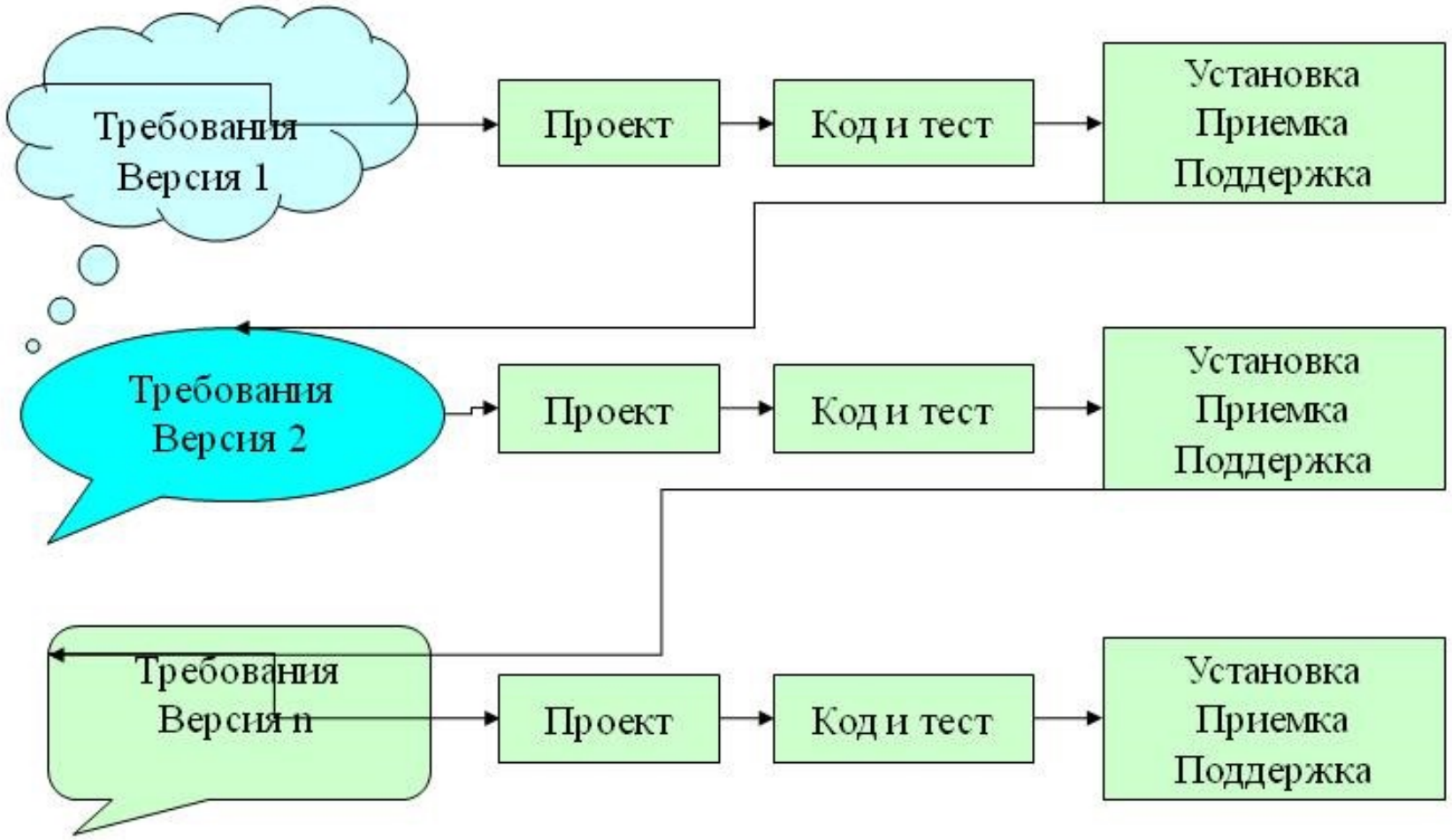
- Продукт поделен на подсистемы и поставляется в релизах (builds)
- Каждый релиз включает обеспечение операционного качества подсистемы
- С каждым новым релизом новая подсистема включается в предыдущий релиз

Модели ЖЦ ПО: Инкремент(аль)ная



Каждый релиз включает детальное проектирование, реализацию, интеграцию, тестирование и передачу

Модели ЖЦ ПО: Эволюционная модель



Модели ЖЦ ПО: Итеративный подход

- Пример – спиральная модель, разработанная Боэмом Boehm [1987]
- Модель с акцентом на задачи (активности)
  - Работает с изменением задач и итерациями
  - Фокус на **управление риском**
- Расширяет каждую активность каскадной модели в **цикл**
- Каждый цикл состоит из 4 этапов
  - Определить цели, альтернативы и указать ограничения
  - Оценить альтернативы, идентифицировать риски и указать пути их снижения
  - Реализовать и проверить текущий цикл
  - Спланировать следующий цикл

Модели ЖЦ ПО: синхростабилизации или Microsoft

- Работа членов команды постоянно синхронизируется
  - Фаза планирования
    - Формулируется видение
    - Готовится документ спецификаций
    - График работ и формирование команды
  - Фаза разработки
    - Первая треть функций (критические функции, разделяемые компоненты)
    - Вторая треть функций
    - Последняя треть функций (наименее критичные функции)
  - Фаза стабилизации
    - Внутреннее тестирование
    - Внешнее тестирование
    - Подготовка релиза
- Сейчас заменяется на MSF

Модели ЖЦ ПО: синхростабилизации или Microsoft

- **Особенности**:
  - 3-4 инкрементных версии ПО, включающих:
  - Синхронизацию (проверка, сборка, тестирование)
  - Стабилизацию(устранение ошибок, найденных тестами)
  - «Заморозку» - работающий «срез» ПО
- **Преимущества:**
  - **«частое и раннее» тестирование** (и выявление ошибок)
  - Постоянная интероперабельность (модули тестируются в сборе => всегда есть работающая версия ПО => связи между модулями легко тестировать)
  - Ранняя коррекция проекта (полная «сборка» ПО первых версий позволяет выявить недочеты проекта до полно-масштабной реализации и снизить стоимость редизайна)

Модели ЖЦ ПО: синхростабилизации или Microsoft

- **Недостатки:**
  - Подходит не для всех типов ПО (скажем, только для поддерживающих автоматизацию тестирования)
  - Необходимо уделять время синхростабилизации (а не только проектированию)
  - **Нужны частые циклы сборки/тестирования** (еженедельно или ежемесячно)
  - **Редко используется вне корпорации Microsoft**

Модели ЖЦ ПО: спиральная

- **Особенности**:
  - Расширение преимуществ быстрого прототипирования на весь ЖЦ ПО
  - В основе – водопадная модель и **анализ рисков**
  - Анализ рисков в начале каждой фазы (выявление и разрешение наиболее серьезных рисков проекта)
  - Завершение проекта при невозможности устранить риски
  - Возможно несколько шагов прототипирования и неограниченое количество итераций

Модели ЖЦ ПО: спиральная

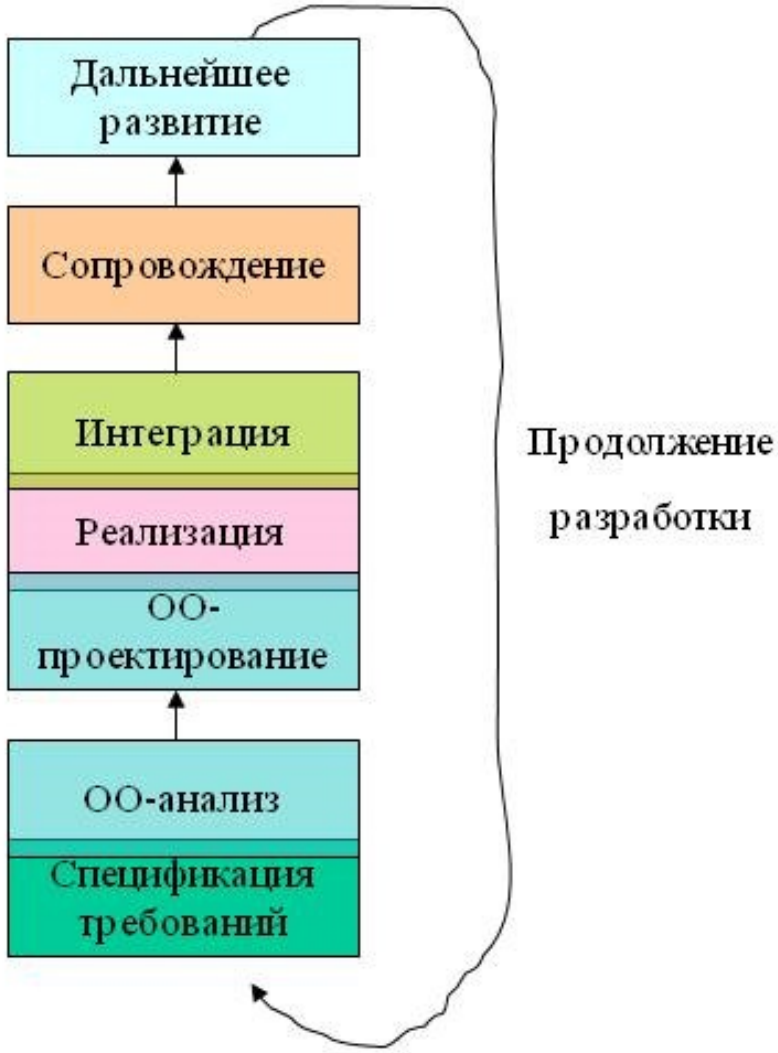
- **Преимущества:**
  - Возможность повторного использования

  (за счет анализа и оценки альтернатив)
  - Обоснование тестирования (за счет анализа рисков)
  - «Бесшовный» переход к сопровождению (благодаря цикличности в разработке ПО до сдачи)
- **Недостатки:**
  - Только для внутренних проектов (т.к. требует предварительной оценки требований и рисков)
  - Только для больших проектов (оценка рисков затратна)
  - **Требует опыта оценки рисков**

Модели ЖЦ ПО: ОО-модель (на прим. *фонтанной*)

- **Особенности**:
  - Интенсивное взаимодействие между фазами ЖЦ ПО
  - Явная итеративная смена фаз ЖЦ ПО
  - **Перекрытие фаз** (ООА и спецификация требований, …)
  - Фазы OOD обычно включают фазы ООА (напр., анализ сценариев = use-case, объектное моделирование)
  - Возможен возврат к предыдущим фазам ЖЦ ПО (стрелки обозначают итеративность)

Модели ЖЦ ПО: ОО-модель



Модели ЖЦ ПО: ОО-модель (на прим. *фонтанной*)

- **Преимущества:**
  - Хорошо подходит для проектирования ОО-приложений
- **Недостатки:**
  - **Слабые ограничения процесса проектирования** могут (при плохой дисциплине проектирования) привести **к вырождению в Build-and-fix**

Модели ЖЦ ПО: сравнительный анализ и выводы

Модель ЖЦ	Преимущества	Недостатки
Build-and-Fix	Хороша для небольших, не требующих сопровождения проектов	Абсолютно непригодна для нетривиальных проектов
Водопадная	Четкая дисциплина проекта, документно-управляемая	ПО может не соответствовать требованиям клиента
Быстрого прототипирования	Обеспечивает соответствие ПО требованиям клиента Максимально ранний возврат	Вызывает соблазн повторного использования кода, который следует заново реализовать
Инкрементная	инвестиций; способствует сопровождаемости Удовлетворяет будущим потребностям	Требует открытой архитектуры; может выродиться в Build-and-fix
Синхронизации и стабилизации	клиента; обеспечивает интеграцию компонент	Не получила широкого применения вне Microsoft
Спиральная	Объединяет хар-ки всех перечисленных выше моделей	Пригодна лишь для крупных внутренних проектов; разработчики должны владеть управлением рисками
ОО-модель	Обеспечивает итерацию внутри фаз и параллелизм между фазами	Может выродиться в CABTAB