Программная инженерия

Жизненный цикл ПО

Определения

Классификация моделей ЖЦ

ЖИЗНЕННЫЙ ЦИКЛ ПО

Жизненный цикл ПО – 1/2

- Жизненный цикл
  - Развитие системы, продукта, услуги, проекта … начиная с разработки концепции и заканчивая прекращением применения
  - Период времени, который начинается с момента принятия решения о необходимости создания ПО и заканчивается в момент его полного изъятия из эксплуатации
- Жизненный цикл может быть описан с помощью модели жизненного цикла, состоящей из стадий. Количество и последовательность стадий не регламентируется и различна в разных типовых моделях

Жизненный цикл ПО – 2/2

- Обычно выделяют стадии (как минимум):
  - Спецификация требований
  - Разработка
  - Валидация
  - Развитие
- Другие стадии:
  - Внедрение
  - Эксплуатация

Типы моделей жизненного цикла

- «Водопадная» (waterfall) [линейная, каскадная]
- Итеративная
- Спиральная
- Инкрементная

WATERFALL МОДЕЛЬ

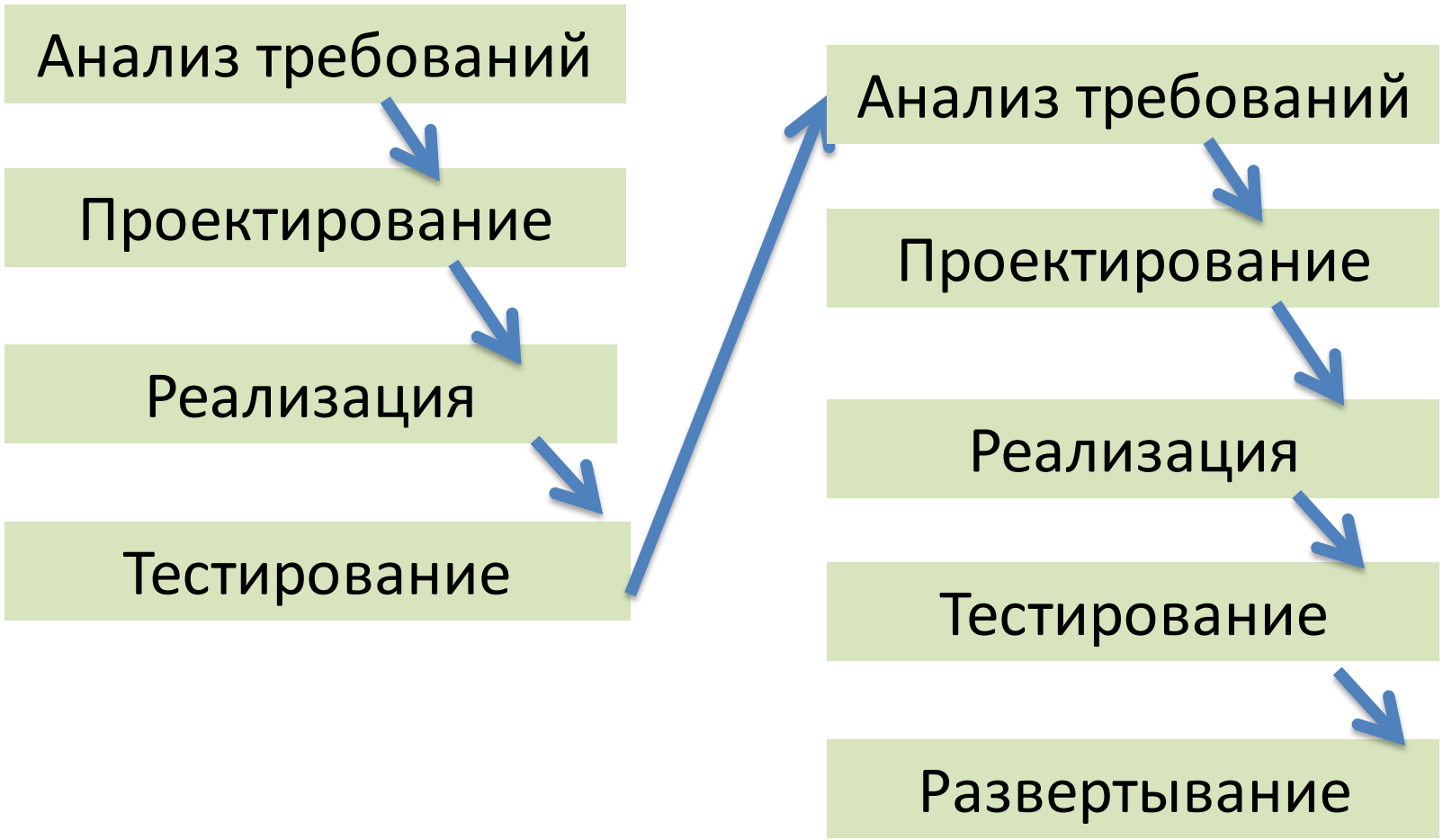
Waterfall – 1/2

Waterfall – 2/2

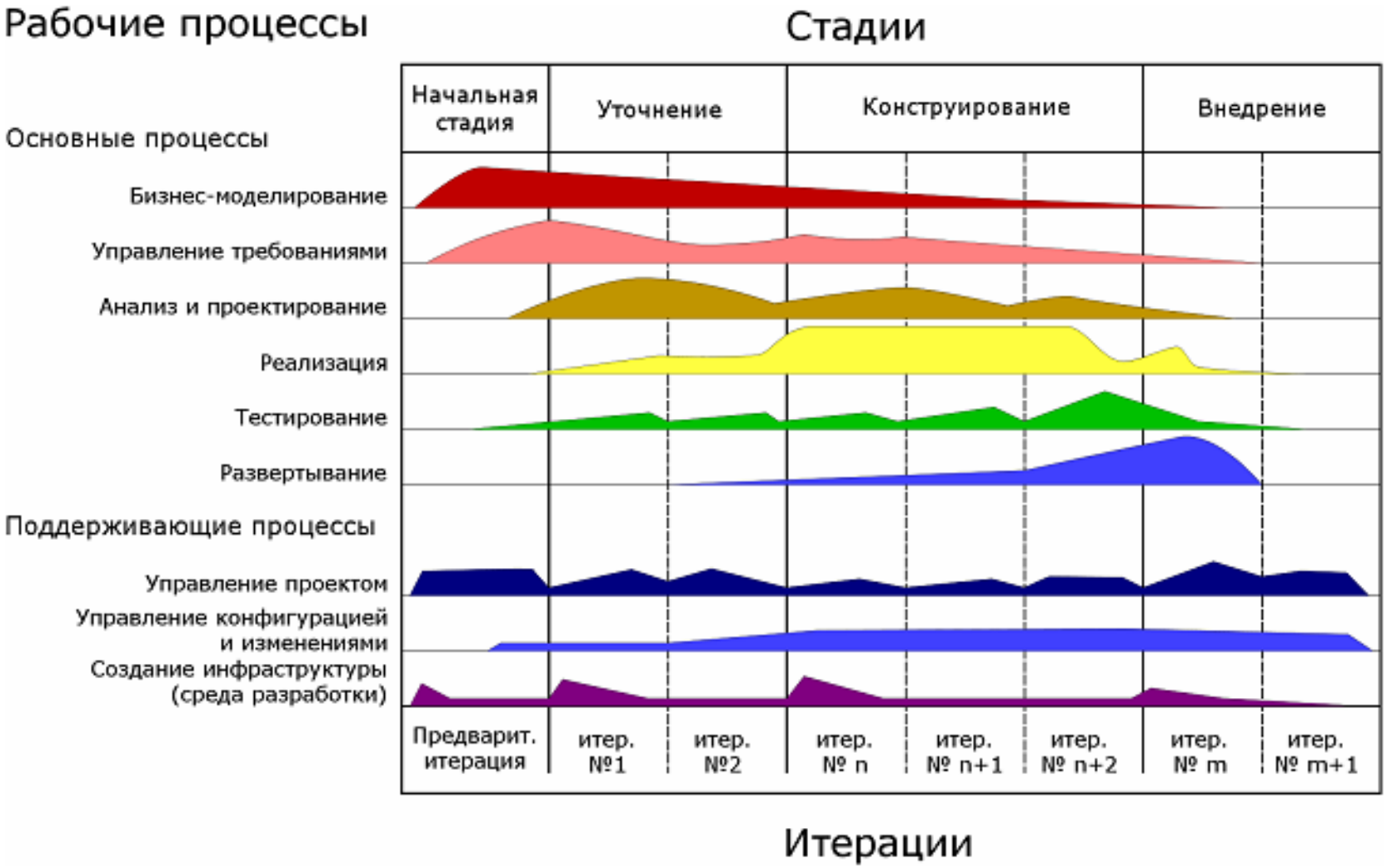
- Достоинства:
  - Формализованная, результатом каждой фазы является документ (утвержденный и подписанный);
  - Заказчик не вмешивается в большую часть работы
- Недостатки:
  - Линейность
  - Требования в дальнейшем не уточняются
  - Все архитектурные решения принимаются на ранней стадии

ИТЕРАТИВНАЯ МОДЕЛЬ

Итеративная модель – 1/5



Итеративная модель – 2/5



Rational Unified Process (RUP)

Итеративная модель – 3/5

- Достоинства:
  - Снижение проектных рисков и минимизация затрат на их устранение
  - Эффективная обратная связь с потребителем
    - Концентрация усилий на наиболее важных и критичных направлениях
    - Раннее обнаружение конфликтов между требованиями, моделями и реализацией проекта
  - Непрерывное итеративное тестирование, позволяющее оценить успешность всего проекта в целом

Итеративная модель – 4/5

- Достоинства (продолжение):
  - 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

Итеративная модель – 5/5

- Недостатки:
  - Неопределенность сроков завершения проекта при изменяющихся внешних условиях
  - Постоянная модификация архитектурной части проекта
  - Неэффективность и/или большой объем тестирования

СПИРАЛЬНАЯ МОДЕЛЬ

Спиральная модель – 1/5

Спиральная модель – 2/5

- 1986 год!
- Сочетает итеративность и этапность
- Виток спирали → создание фрагмента или версии ПО
  - Уточняются цели и характеристики проекта
  - Определяется качество
  - Планируются работы следующего витка
- Детализация проекта последовательно увеличивается → обоснованно выбирается вариант, доводимый до реализации

Спиральная модель – 3/5

- Каждый виток разбит на 4 сектора:
  - Оценка и разрешение рисков
  - Определение целей
  - Разработка и тестирование
  - Планирование
- На каждом витке спирали могут применяться разные модели процесса разработки ПО
- В конечном итоге на выходе получается готовый продукт (Точно-точно?)

Спиральная модель – 4/5

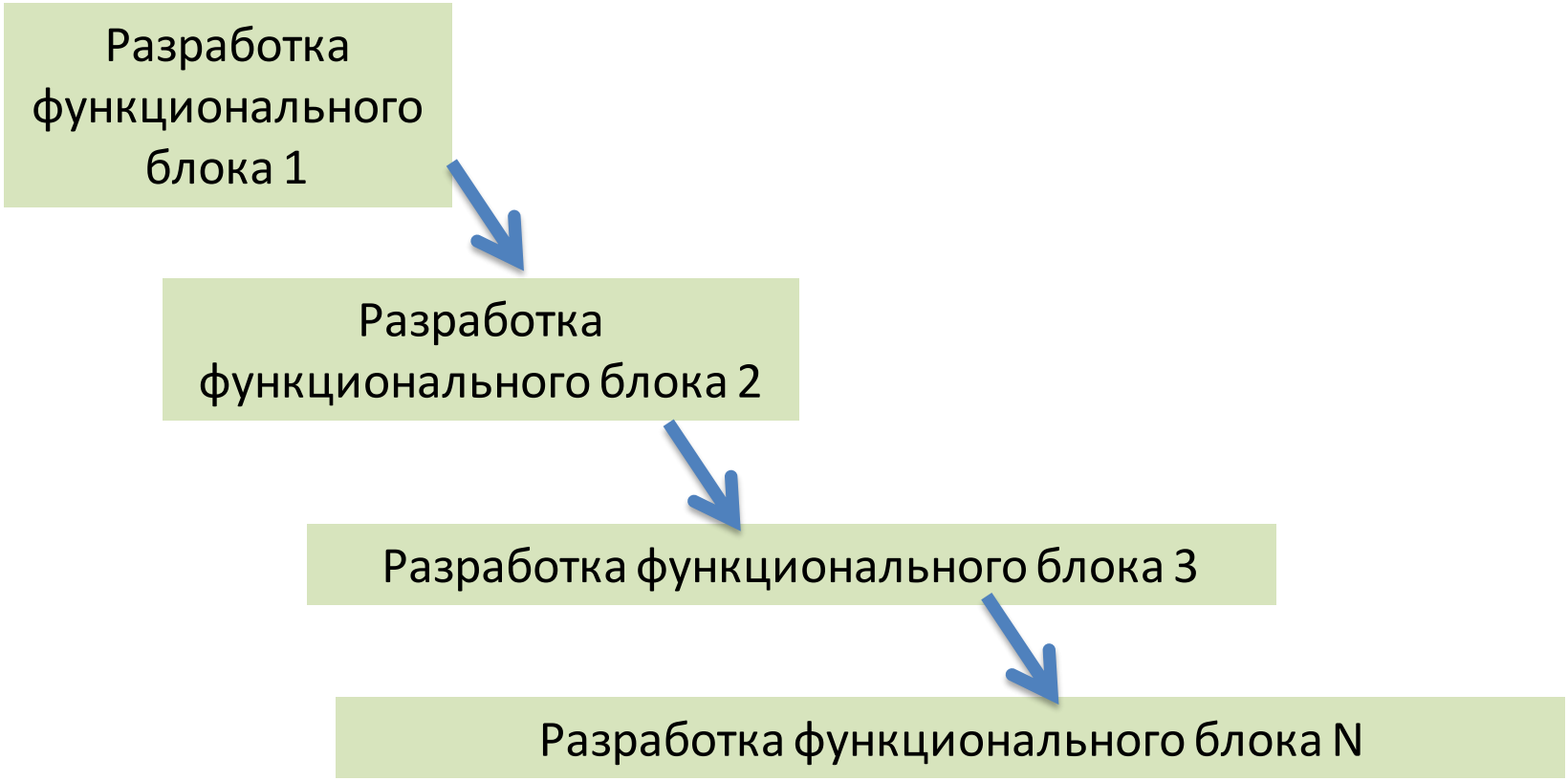
- Главная задача – как можно быстрее показать заказчику (пользователям) работоспособный продукт (прототип, версию), запуская процесс уточнения и дополнения требований
- Основная проблема спирального цикла – определение момента перехода на следующий этап
- Вариант решения:
  - Ввод временных ограничений на каждый из этапов жизненного цикла
  - Тогда переход осуществляется в заданный момент планом, даже если не вся запланированная работа закончена (NB!)
  - План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков (как всегда…)

Спиральная модель – 5/5

- Достоинства:
  - Явная оценка рисков
  - Быстрое получение работающих прототипов
- Недостатки:
  - Опасность принятия неверных архитектурных решений
  - Постоянное вмешательство заказчика может деструктивно влиять на работу разработчиков
  - Накопление недоработок в случае неверной оценки длительности этапов

ИНКРЕМЕНТНАЯ МОДЕЛЬ

Инкрементная модель – 1/2



Инкрементная модель – 2/2

- Достоинства:
  - Быстрое получение частичной функциональности
  - Возможность корректировки последовательности разработки блоков
- Недостатки:
  - Разработка каждого последующего блока может влиять на функционирование предыдущих
  - Сложность обеспечения взаимодействия между блоками

SDLC vs. SPLC

Классификация: процессов ЖЦ, моделей ЖЦ

Работа с требованиями

Практические соображения

SWEBOK – LIFE CYCLES (KA 8.2)

SDLC vs. SPLC

- SDLC – software development life cycle:
  - Процессы, нацеленные на превращение требований в готовый к поставке продукт
- SPLC – software product life cycle = Процессы SDLC +

  + развертывание, поддержку (maintenance, support), развитие, вывод из эксплуатации, etc.

  + поддерживающие процессы относящиеся к продукту: управление конфигурациями, качеством, etc.
  - SPLC может включать несколько SDLC как отражение операций по развитию и улучшению ПО

Типы процессов – 1/2

- Основные (primary) – процессы разработки, использования (operation) и поддержки
- Вспомогательные (supporting) – процессы, направленные на обеспечение выполнения основных процессов. Включают:
  - Управление конфигурациями
  - Управление качеством
  - Верификацию
  - Валидацию

Типы процессов – 2/2

- Организационные (organizational) – процессы, обеспечивающие программную инженерию. Включают:
  - Обучение (training)
  - Анализ и измерения (process measurement analysis)
  - Управление инфраструктурой
  - Управление повторным использованием и описаниями (portfolio and reuse management)
  - Улучшение организационных процессов
  - Управление жизненным циклом
- Межпроектные (cross-project) – такие процессы, как повторное использование, создание и поддержка линеек продуктов, управление доменами (domain engineering). Включают в себя взаимодействие более чем с одним проектом в организации

Виды моделей ЖЦ – 1/2

- Линейные (linear) – разработка ПО выполняется последовательно с использованием обратных связей или итераций где это нужно с последующей интеграцией, тестированием и поставкой отдельного продукта
- Итеративные (iterative) – ПО разрабатывается итерациями, расширяющими функциональность
- Гибкие (agile) – включают частые демонстрации работающего ПО представителям заказчика и/или пользователя, которые направляют последующую разработку ПО, организованную в виде коротких итеративных циклов, добавляющих небольшие «кусочки» функциональности к уже используемому ПО
- Инкрементные, итеративные и гибкие модели позволяют [быстро] создавать и предоставлять пользователю частично функциональное ПО

Виды моделей ЖЦ – 2/2

- Линейные модели SDLC также называют as предсказывающими (прогностическими, predictive)
- Итеративные и гибкие модели SDLC также называют адаптивными
- Процессы поддержки (maintenance) входящие в SPLC могут быть организованы с использованием различных моделей SDLC в зависимости от организуемого процесса

Модели ЖЦ и требования – 1/2

- Модели ЖЦ отличаются своим отношением к работе с требованиями
- Линейные модели обычно предполагают разработку полного описания требований (до предела возможного) на этапе инициализации и планирования
  - Затем требования к ПО строго контролируются
  - Изменения в требования вносятся на основании документально оформленных запросов (change requests), обрабатываемых официально и регламентированно

Модели ЖЦ и требования – 2/2

- Инкрементные модели порождают последовательно вносимые в [работающий] продукт добавления. При этом требования создаются к каждому добавлению. При этом:
  - Требования могут строго контролироваться (как в линейной модели)
  - Возможна некоторая гибкость, заключающаяся в пересмотре требований к продукту во время его развития
  - Гибкие модели на начальном этапе могут определить область применения ПО и высокоуровневые требования

Однако! Гибкие модели были созданы для облегчения развития (смены) требований в процессе разработки

Практические соображения – 1/2

- Процессы ЖЦ часто:
  - Чередуются – разбиваются на более короткие и выполняются по очереди с использованием результатов друг друга
  - Перекрываются – процесс начинается не дожидаясь [полного] завершения другого (по идее, предшествующего)
  - Выполняются одновременно
- Описание процессов ЖЦ как отдельных, имеющих строго заданные критерии начала и окончания, установленные ограничения и интерфейсов следует понимать как идеализацию, которую нужно адаптировать к реальной ситуации в конкретной предметной области и организации

Практические соображения – 2/2

- Процессы ЖЦ (такие как управление конфигурациями, разработка, тестирование) могут быть адаптированы для облегчения использования, поддержки, сопровождения, миграции и вывода ПО из эксплуатации
- Факторы, влияющие на «подстройку» (tailoring) процессов ЖЦ:
  - Соответствие стандартам, регулирующим документам (directives), политикам
  - Потребности (не требования! demands) заказчика
  - Организационная зрелость и компетентность (organizational maturity and competencies)
  - Критичность [качества] ПО
  - Сущность проекта: модификация существующего продукта, разработка нового, etc.
  - Предметная область (например, аэерокосмические системы ср. управление отелем)