Методологии разработки ПО

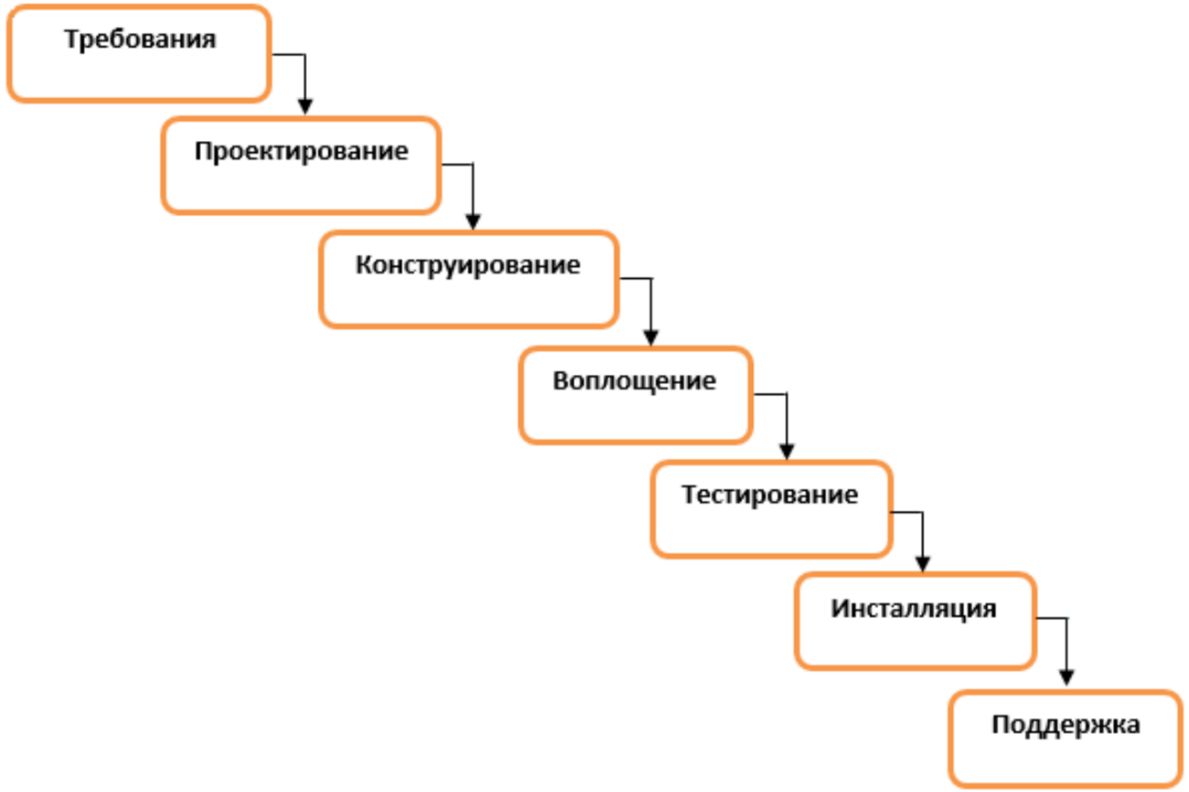
Лекция 3
Базовые методологии разработки ПО (ч.1)

Старичков Н.Ю., Крахмалёв Д.С., ФКН ВШЭ, 2021/2022 уч.год

КАСКАДНАЯ МОДЕЛЬ

КАСКАДНАЯ МОДЕЛЬ

- **Последовательное прохождение стадий**
- **Стадия не начинается, пока не закончена предыдущая**
- **Четкие:**
  - Сроки
  - Стоимость
  - Результат
- **Но есть и минусы**



Хороший пример №1

Необходимо разработать библиотеку для реализации длинной арифметики на языке C++

Заранее понятны и известны:

- требования (и они неизменны)
- технологии (в данном случае, алгоритмы)
- сроки
- ресурсы

Хороший пример №1

Необходимо разработать библиотеку для реализации длинной арифметики на языке C++

- мы можем работать постепенно реализуя функциональность
- выпуск будет в самом конце - когда мы все сделаем
- можем заложить необходимое время на проектирование и тесты
- можем сразу оценить сроки (по требованиям)

Хороший пример №2

Делаем свою компьютерную игру с аддонами

Заранее известны:

- требования (мы сразу знаем, чего хотим)
- сроки - и мы их можем зафиксировать
- можем запланировать ресурсы

Хороший пример №2

Делаем свою компьютерную игру с аддонами

- У нас несколько выпусков - но мы заранее знаем, что и когда будет
- Можем заложить необходимое время на проектирование и тесты
- Можем работать, ожидая результат к конкретному сроку

Хороший пример №2

Делаем свою компьютерную игру с аддонами

- У нас несколько выпусков - но мы заранее знаем, что и когда будет
- Можем заложить необходимое время на проектирование и тесты
- Можем работать, ожидая результат к конкретному сроку

- не предполагается возможность параллельной работы над разными частями в базовом подходе, однако, это можно расширить

Плохой пример №1

Хотим делать мессенджер (да, очередной)

В чем проблемы:

- мы не можем заранее знать всю функциональность - она будет добавляться на ходу
- соответственно, выпускаемся мы регулярно и часто - и тяжело заранее сказать, сколько таких выпусков будет
  - отсюда еще следуют проблемы со сроками и/или качеством

Плохой пример №1

Хотим делать мессенджер

- Еще у нас появляется сложность с планированием ресурсов - придется как-то решать на лету
- В зависимости от числа задач, их сложности, их комбинации (и наличия ресурсов) - мы получаем, состояние проекта в какой-то фиксированный момент времени непредсказуемо

Плохой пример №2

Мы делаем софт для автоматизации бизнеса (в части налогового учета)

- У нас может быть хорошее понимание, что мы должны сделать
- Мы можем четко запланировать сроки и хорошо распределить ресурсы
- Можем заранее запланировать выпуск (выпуски)

Плохой пример №2

Мы делаем софт для автоматизации бизнеса (в части налогового учета)

- У нас может быть хорошее понимание, что мы должны сделать
- Мы можем четко запланировать сроки и хорошо распределить ресурсы
- Можем заранее запланировать выпуск (выпуски)

Только вот новые законы могут принять в любой момент

И времени на реакцию может быть совсем немного

V-МОДЕЛЬ

V-МОДЕЛЬ

- **Верификация**
  - Общая концепция
  - Требования бизнеса
  - Функциональные требования
  - Архитектура
  - Реализация
- **Валидация**
  - Приемо-сдаточное тестирование
  - Функциональное тестирование
  - Интеграционное тестирование
  - Модульное тестирование

Хороший пример №1

Допустим, мы делаем небольшую систему, но в области, которую не до конца понимаем - например, делаем генератор ПУДов

- Нам придется отразить все требования в тестах - а значит, ничего не упустим (а требования может составить специалист)
- Мы проверим все детали реализации, проектирования, концепции (выполнение требований)

Хороший пример №1

Допустим, мы делаем небольшую систему, но в области, которую не до конца понимаем - например, делаем генератор ПУДов

- Нам придется отразить все требования в тестах - а значит, ничего не упустим (а требования может составить специалист)
- Мы проверим все детали реализации, проектирования, концепции (выполнение требований)

Ключевое преимущество в том, что система тестирования будет развиваться вместе с развитием проекта

Хороший пример №2

Мы проводим исследование, например, пытаемся создать систему, которая будет по фотографии определять страну, в которой она сделана

- Фактически, подобная модель разработки позволяет нам в любой момент остановить проект, как только возникли непреодолимые сложности
- Мы на каждом этапе будем четко понимать, в каком статусе находимся

Хороший пример №2

Мы проводим исследование, например, пытаемся создать систему, которая будет по фотографии определять страну, в которой она сделана

- Фактически, подобная модель разработки позволяет нам в любой момент остановить проект, как только возникли непреодолимые сложности
- Мы на каждом этапе будем четко понимать, в каком статусе находимся

Можете попробовать применить к своей ВКР

Хороший пример №3

Допустим, мы делаем сложную, многокомпонентную систему

Например, автоматизируем деятельность вуза

- У нас есть четкий план и четкое понимание сроков
- Мы понимаем, кто, когда и что будет делать
- Мы поэтапно расширяем систему тестирования

Хороший пример №3

Допустим, мы делаем сложную, многокомпонентную систему

Например, автоматизируем деятельность вуза

- Мы можем успешно довести до результата сложный проект
- На каждом уровне все будет протестировано и отработано
- Более того, мы будем в каждый момент времени четко понимать, в каком состоянии находится проект (и, соответственно, нам будет проще управлять процессом)

Плохой пример №1

Мы делаем систему для торговли биткоинами

- Понимаем, что хотим сделать
- Понимаем сроки
- Понимаем, как будем делать

Плохой пример №1

Мы делаем систему для торговли биткоинами

- Понимаем, что хотим сделать
- Понимаем сроки
- Понимаем, как будем делать

Правила игры поменялись - и нам приходится откатывать назад весь проект со всеми достигнутыми на текущий момент результатами

Плохой пример №2

Придумали стартап - будем нейросетями предсказывать новости

- Понимаем, что хотим сделать
- Допустим, даже понимаем, как хотим делать
- Понимаем требования
- Можем даже оценить сроки

Плохой пример №2

Придумали стартап - будем нейросетями предсказывать новости

- Понимаем, что хотим сделать
- Допустим, даже понимаем, как хотим делать
- Понимаем требования
- Можем даже оценить сроки

Проблема в том, что показать что-то мы можем только добравшись “до основания V” - т.е. до этапа реализации

Плохой пример №3 (самый плохой)

Допустим, мы делаем сложную, многокомпонентную систему

Например, автоматизируем деятельность вуза

- У нас есть четкий план и четкое понимание сроков
- Мы понимаем, кто, когда и что будет делать
- Мы поэтапно расширяем систему тестирования

Мы говорили про это, как про хороший пример

Все хорошо,
за исключением того,
что мы не подумали о возможности ошибки

Плохой пример №3 (самый плохой)

Допустим, мы делаем сложную, многокомпонентную систему

Например, автоматизируем деятельность вуза

- У нас есть четкий план и четкое понимание сроков
- Мы понимаем, кто, когда и что будет делать
- Мы поэтапно расширяем систему тестирования

На этапе сбора требований не учли, что образование может быть дистанционным - и все, надо все переделывать (или почти все)

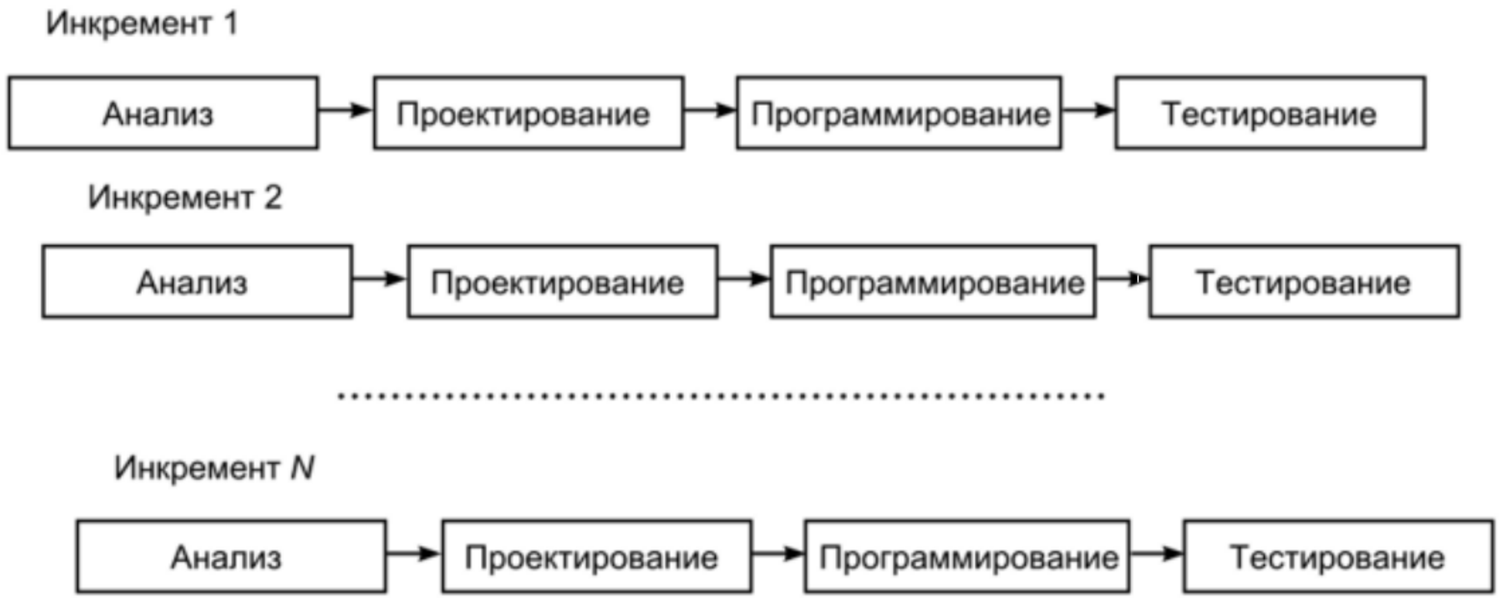
Примеры из не IT

- *Некоторые советские подводные лодки не могут плавать в тропических морях (вода слишком теплая; система была рассчитана на другую температуру забортной воды) (проектирование концепции)*
- *Разделили на этапе проектирования на модули - и не продумали, как модули будут взаимодействовать друг с другом в нештатной ситуации (например, современные автомобили - пример с аккумулятором) (проектирование архитектуры)*
- *Пример с доработкой “Оки” (детальное проектирование)*

ИНКРЕМЕНТНАЯ МОДЕЛЬ

ИНКРЕМЕНТНАЯ МОДЕЛЬ

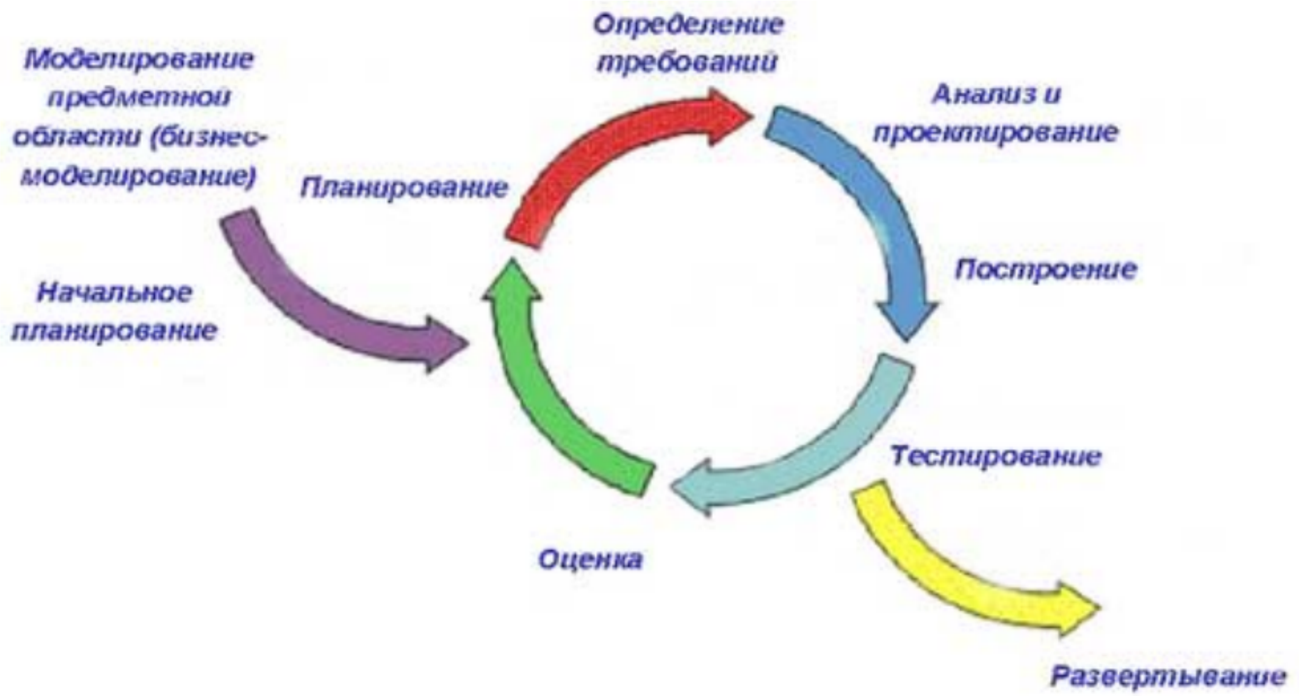
- **Первая версия - базовая**
- **Далее - дополнительные возможности**
- **На каждом этапе:**
  - Определение требований
  - Проектирование
  - Реализация
  - Внедрение
  - Тестирование



ИТЕРАЦИОННАЯ МОДЕЛЬ

ИТЕРАЦИОННАЯ МОДЕЛЬ

- Каждый этап - база для следующего (определение требований)
- Важный момент - каждая версия полностью работоспособна
- Проводится анализ (опционально - сбор обратной связи) по каждой выпущенной версии для формирования требований и планов для следующей версии



Тонкий пример

- *Допустим, мы разрабатываем операционную систему*

Тонкий пример

- *Допустим, мы разрабатываем операционную систему*
  - *Хорошо понимаем требования к каждой итерации*
  - *Можем четко планировать работу над итерацией*
  - *Проводим анализ и другие стадии*

Тонкий пример

- *Допустим, мы разрабатываем операционную систему*
  - *Хорошо понимаем требования к каждой итерации*
  - *Можем четко планировать работу над итерацией*
  - *Проводим анализ и другие стадии*

Но тут важный момент - насколько можно называть это инкрементной разработкой с таким длинным пробегом

Тонкий пример

- *Допустим, мы разрабатываем операционную систему*
  - *Хорошо понимаем требования к каждой итерации*
  - *Можем четко планировать работу над итерацией*
  - *Проводим анализ и другие стадии*

Давайте добавим, что мы выпускаем промежуточные версии
Например, обновления

Тонкий пример -> плохой пример

- *Допустим, мы разрабатываем операционную систему*
  - *Хорошо понимаем требования к каждой итерации*
  - *Можем четко планировать работу над итерацией*
  - *Проводим анализ и другие стадии*

Давайте добавим, что мы выпускаем промежуточные версии
Например, обновления
Обновления, зависящие от исправления ошибок - плохой пример

Тонкий пример -> хороший пример

- *Допустим, мы разрабатываем операционную систему*
  - *Хорошо понимаем требования к каждой итерации*
  - *Можем четко планировать работу над итерацией*
  - *Проводим анализ и другие стадии*

Давайте добавим, что мы выпускаем промежуточные версии
Например, обновления
Регулярные обновления с новыми возможностями - хороший пример

Тонкий пример №2

- *Допустим, мы разрабатываем систему автоматизации магазина*

Тонкий пример №2

- *Допустим, мы разрабатываем систему автоматизации магазина*
  - *Часто появляются какие-то новые пожелания*
  - *Обновляемся как придется, очень часто*

Тонкий пример №2

- *Допустим, мы разрабатываем систему автоматизации магазина*
  - *Часто появляются какие-то новые пожелания*
  - *Обновляемся как придется, очень часто*

Проблема в том, как именно мы организуем работу

Тонкий пример №2 -> плохой пример

- *Допустим, мы разрабатываем систему автоматизации магазина*
  - *Часто появляются какие-то новые пожелания*
  - *Обновляемся как придется, очень часто*

Проблема в том, как именно мы организуем работу
“Мальчик на побегушках”

Тонкий пример №2 -> хороший пример

- *Допустим, мы разрабатываем систему автоматизации магазина*
  - *Часто появляются какие-то новые пожелания*
  - *Обновляемся как придется, очень часто*

Проблема в том, как именно мы организуем работу
Заранее запланировали расписание всех этапов, работаем “по плану”

Плохой пример №1

Допустим, мы делаем библиотеку для длинной арифметики

- *Снова у нас есть отличные требования*
- *Понятные сроки*
- *Понятные технологии/алгоритмы*

Плохой пример №1

Допустим, мы делаем библиотеку для длинной арифметики

- *Снова у нас есть отличные требования*
- *Понятные сроки*
- *Понятные технологии/алгоритмы*

Но только зачем нам тут придумывать итерации?

Плохой пример №1

Допустим, мы делаем библиотеку для длинной арифметики

- *Снова у нас есть отличные требования*
- *Понятные сроки*
- *Понятные технологии/алгоритмы*

Но только зачем нам тут придумывать итерации?

Как уже обсуждалось, тут достаточно каскадной модели

Плохой пример №1

Допустим, мы делаем библиотеку для длинной арифметики

- *Снова у нас есть отличные требования*
- *Понятные сроки*
- *Понятные технологии/алгоритмы*

Но только зачем нам тут придумывать итерации?

Как уже обсуждалось, тут достаточно каскадной модели

Не нужно усложнять и создавать лишнее сущности

Плохой пример №2

Допустим, теперь мы делаем многокомпонентную систему - например, пишем ПО для управления автомобилем

- *Понятны требования*
- *Понятны технологии*
- *Понятны сроки*

Плохой пример №2

Допустим, теперь мы делаем многокомпонентную систему - например, пишем ПО для управления автомобилем

- *Понятны требования*
- *Понятны технологии*
- *Понятны сроки*

Проблема в том, что система очень разнородная - и тяжело привести все команды к работе в едином “ритме”

Плохой пример №2

Допустим, теперь мы делаем многокомпонентную систему - например, пишем ПО для управления автомобилем

- *Понятны требования*
- *Понятны технологии*
- *Понятны сроки*

Проблема в том, что система очень разнородная - и тяжело привести все команды к работе в едином “ритме”
Соответственно, получим по итогу или простой команд, либо срыв сроков

Плохой пример №2

Допустим, теперь мы делаем многокомпонентную систему - например, пишем ПО для управления автомобилем

- *Понятны требования*
- *Понятны технологии*
- *Понятны сроки*

Проблема в том, что система очень разнородная - и тяжело привести все команды к работе в едином “ритме”
Соответственно, получим по итогу или простой команд, либо срыв сроков

Тут можно разрешить командам начинать итерации в своем темпе, с общим сроком на конец итерации

Плохой пример №3

Допустим, мы делаем софт для автоматизации игровых магазинов

- *Понимаем требования*
- *Понимаем технологии*

Плохой пример №3

Допустим, мы делаем софт для автоматизации игровых магазинов

- *Понимаем требования*
- *Понимаем технологии*

Тут проблема в том, что требования могут меняться быстрее, чем мы успеваем завершить итерацию

Плохой пример №3

Допустим, мы делаем софт для автоматизации игровых магазинов

- *Понимаем требования*
- *Понимаем технологии*

Тут проблема в том, что требования могут меняться быстрее, чем мы успеваем завершить итерацию
Соответственно, мы будем всегда в роли “догоняющего” или пострадает качество

Хороший пример №1

И снова мессенджер...

Хороший пример №1

И снова мессенджер…

- *Мы можем хорошо продумать функциональность и четко разбить на этапы*
- *Быстро получим MVP - и выйдем на рынок*
- *Будем работать в фиксированном темпе*

Хороший пример №1

И снова мессенджер…

- *Мы можем хорошо продумать функциональность и четко разбить на этапы*
- *Быстро получим MVP - и выйдем на рынок*
- *Будем работать в фиксированном темпе*

В целом подобные проекты отлично ложатся на инкрементную модель
Когда нужно и быстро сделать минимум, и стабильно развиваться

Хороший пример №2

Делаем систему автоматизации большого предприятия

Хороший пример №2

Делаем систему автоматизации большого предприятия

- *Более-менее понятны требования*
- *Система большая, и заказчик хочет получать ее частями, а не через 3 года “все и сразу” - для него это вопрос денег*

Хороший пример №2

Делаем систему автоматизации большого предприятия

- *Более-менее понятны требования*
- *Система большая, и заказчик хочет получать ее частями, а не через 3 года “все и сразу” - для него это вопрос денег*
- *Система сложная не только в разработке, но и во внедрении*

Хороший пример №2

Делаем систему автоматизации большого предприятия

- *Более-менее понятны требования*
- *Система большая, и заказчик хочет получать ее частями, а не через 3 года “все и сразу” - для него это вопрос денег*
- *Система сложная не только в разработке, но и во внедрении*

Инкрементная модель хороша тем, что мы можем в конце каждой итерации делать внедрение данной части - не будет “снежного кома”

Хороший пример №2

Делаем систему автоматизации большого предприятия

- *Более-менее понятны требования*
- *Система большая, и заказчик хочет получать ее частями, а не через 3 года “все и сразу” - для него это вопрос денег*
- *Система сложная не только в разработке, но и во внедрении*

Инкрементная модель хороша тем, что мы можем в конце каждой итерации делать внедрение данной части - не будет “снежного кома”

Плюс по итогам внедрения мы можем вносить корректировки в планы следующей итерации

Хороший пример №3

Допустим, делаем тиражное решение для автоматизации документооборота

- *Понимаем требования*
- *Понимаем, что нужно сделать*
- *Хотим выпустить “полный”, готовый продукт*

Причем тут вообще итерации в таком случае?

Хороший пример №3

Допустим, делаем тиражное решение для автоматизации документооборота

- *Понимаем требования*
- *Понимаем, что нужно сделать*
- *Хотим выпустить “полный”, готовый продукт*

Проект сложный, и мы боимся, что не все учли

Хороший пример №3

Допустим, делаем тиражное решение для автоматизации документооборота

- *Понимаем требования*
- *Понимаем, что нужно сделать*
- *Хотим выпустить “полный”, готовый продукт*

Проект сложный, и мы боимся, что не все учли
Могут быть какие-то ошибки и проблемы у пользователей

Хороший пример №3

Допустим, делаем тиражное решение для автоматизации документооборота

- *Понимаем требования*
- *Понимаем, что нужно сделать*
- *Хотим выпустить “полный”, готовый продукт*

Проект сложный, и мы боимся, что не все учли
Могут быть какие-то ошибки и проблемы у пользователей

EAT YOUR OWN DOG FOOD

Хороший пример №3

Допустим, делаем тиражное решение для автоматизации документооборота

- *Понимаем требования*
- *Понимаем, что нужно сделать*
- *Хотим выпустить “полный”, готовый продукт*

Проект сложный, и мы боимся, что не все учли
Могут быть какие-то ошибки и проблемы у пользователей

EAT YOUR OWN DOG FOOD
Т.е. мы можем организовать итерации для “внутреннего клиента” - и вести разработку таким способом

Хороший пример №3

Допустим, делаем тиражное решение для автоматизации документооборота

- *Понимаем требования*
- *Понимаем, что нужно сделать*
- *Хотим выпустить “полный”, готовый продукт*

Проект сложный, и мы боимся, что не все учли
Могут быть какие-то ошибки и проблемы у пользователей

EAT YOUR OWN DOG FOOD
Т.е. мы можем организовать итерации для “внутреннего клиента” - и вести разработку таким способом
А когда все будет “ОК” - выйдем на рынок

>>: tbc…