

Методологии разработки ПО

Лекция 4

Базовые методологии разработки ПО (ч.1 / продолжение)

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

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

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...