Методологии разработки ПО

Лекция 6
Семейство гибких методологий

Старичков Н.Ю., ФКН ВШЭ, 2022/2023 уч.год

СЕМЕЙСТВО ГИБКИХ МЕТОДОЛОГИЙ

AGILE

- Семейство гибких методологий разработки
- Короткие итерации
- Разные метрики качества работы
- Много разных конкретных подходов

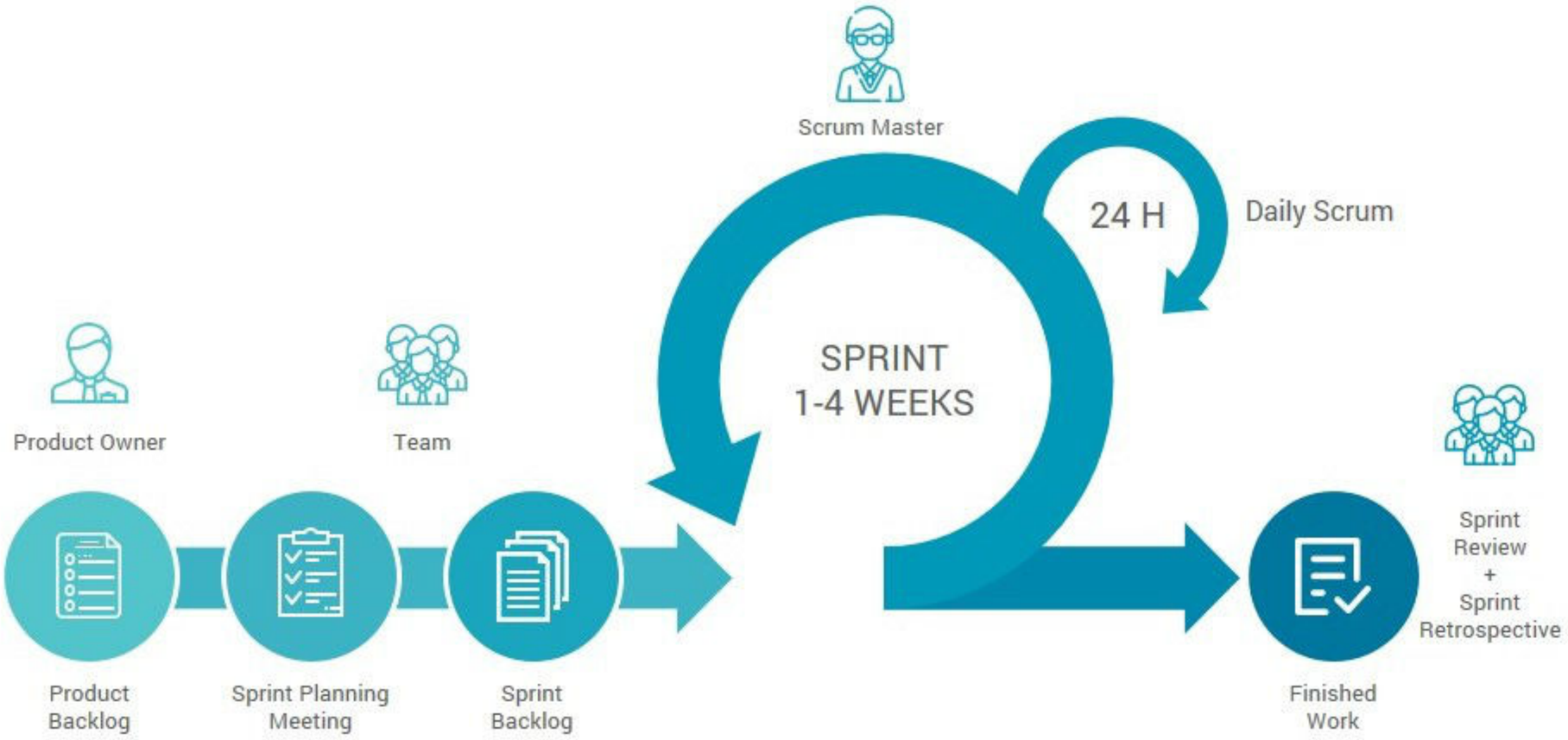
AGILE-МАНИФЕСТ

AGILE-МАНИФЕСТ

- **ЛЮДИ** и **ВЗАИМОДЕЙСТВИЕ** важнее **процессов** и **инструментов**
- **РАБОТАЮЩИЙ ПРОДУКТ** важнее **документации**
- **СОТРУДНИЧЕСТВО С ЗАКАЗЧИКОМ** важнее **условий контракта**
- **ГОТОВНОСТЬ К ИЗМЕНЕНИЯМ** важнее **первоначального плана**

SCRUM

SCRUM


Scrum Master
24 H
Daily Scrum
SPRINT
1-4 WEEKS
Product Owner
Team
Sprint
Review
+
Sprint
Retrospective
Product
Backlog
Sprint Planning
Meeting
Sprint
Backlog
Finished
Work

Плохой пример

Scrum

- *Допустим, мы делаем проект по управлению ракетой*

Плохой пример

Scrum

- *Допустим, мы делаем проект по управлению ракетой*

  Казалось бы, что тут может быть плохого в контексте Scrum?

Плохой пример

Scrum

- *Допустим, мы делаем проект по управлению ракетой*

  Казалось бы, что тут может быть плохого в контексте Scrum?

  Проблема заключается в большом количестве исследовательских задач

Плохой пример

Scrum

- *Допустим, мы делаем проект по управлению ракетой*

  Казалось бы, что тут может быть плохого в контексте Scrum?

  Проблема заключается в большом количестве исследовательских задач
  => сложно прогнозировать сроки выполнения задач

Плохой пример

Scrum

- *Допустим, мы делаем проект по управлению ракетой*

  Казалось бы, что тут может быть плохого в контексте Scrum?

  Проблема заключается в большом количестве исследовательских задач
  => сложно прогнозировать сроки выполнения задач
  => сложно прогнозировать и промежуточные этапы

Плохой пример

Scrum

- *Допустим, мы делаем проект по управлению ракетой*

  Казалось бы, что тут может быть плохого в контексте Scrum?

  Проблема заключается в большом количестве исследовательских задач
  => сложно прогнозировать сроки выполнения задач
  => сложно прогнозировать и промежуточные этапы
  ===> сложно (или почти невозможно) планировать спринты

Плохой пример

Scrum

- *Допустим, мы делаем проект по управлению ракетой*

  Казалось бы, что тут может быть плохого в контексте Scrum?

  Проблема заключается в большом количестве исследовательских задач
  => сложно прогнозировать сроки выполнения задач
  => сложно прогнозировать и промежуточные этапы
  ===> сложно (или почти невозможно) планировать спринты

  => не всегда в конце задачи нам есть, что показать заказчику

Плохой пример

Scrum

- *Допустим, мы делаем проект по управлению ракетой*

  Казалось бы, что тут может быть плохого в контексте Scrum?

  Проблема заключается в большом количестве исследовательских задач
  => сложно прогнозировать сроки выполнения задач
  => сложно прогнозировать и промежуточные этапы
  ===> сложно (или почти невозможно) планировать спринты

  => не всегда в конце задачи нам есть, что показать заказчику
  А иногда и не стоит показывать то, что у нас на текущий момент есть

Плохой пример

Scrum

- *В целом плохо подходит для проектов, в которых высока доля неопределенности при выполнении задач*

Плохой пример

Scrum

- *В целом плохо подходит для проектов, в которых высока доля неопределенности при выполнении задач*
- *Плохо подходит для проектов, у которых сложно выделить большое количество промежуточных этапов*

Плохой пример

Scrum

- *В целом плохо подходит для проектов, в которых высока доля неопределенности при выполнении задач*
- *Плохо подходит для проектов, у которых сложно выделить большое количество промежуточных этапов*
- *Могут быть сложности из-за приоритизации задач*

Хороший пример

Scrum

Допустим, мы делаем веб-сервис для студентов

Хороший пример

Scrum

Допустим, мы делаем веб-сервис для студентов

- Нам хорошо понятно, что делать
- Задачи небольшие и простые - легко планировать спринты
- В конце каждого спринта мы можем выкатывать новую версию

Хороший пример

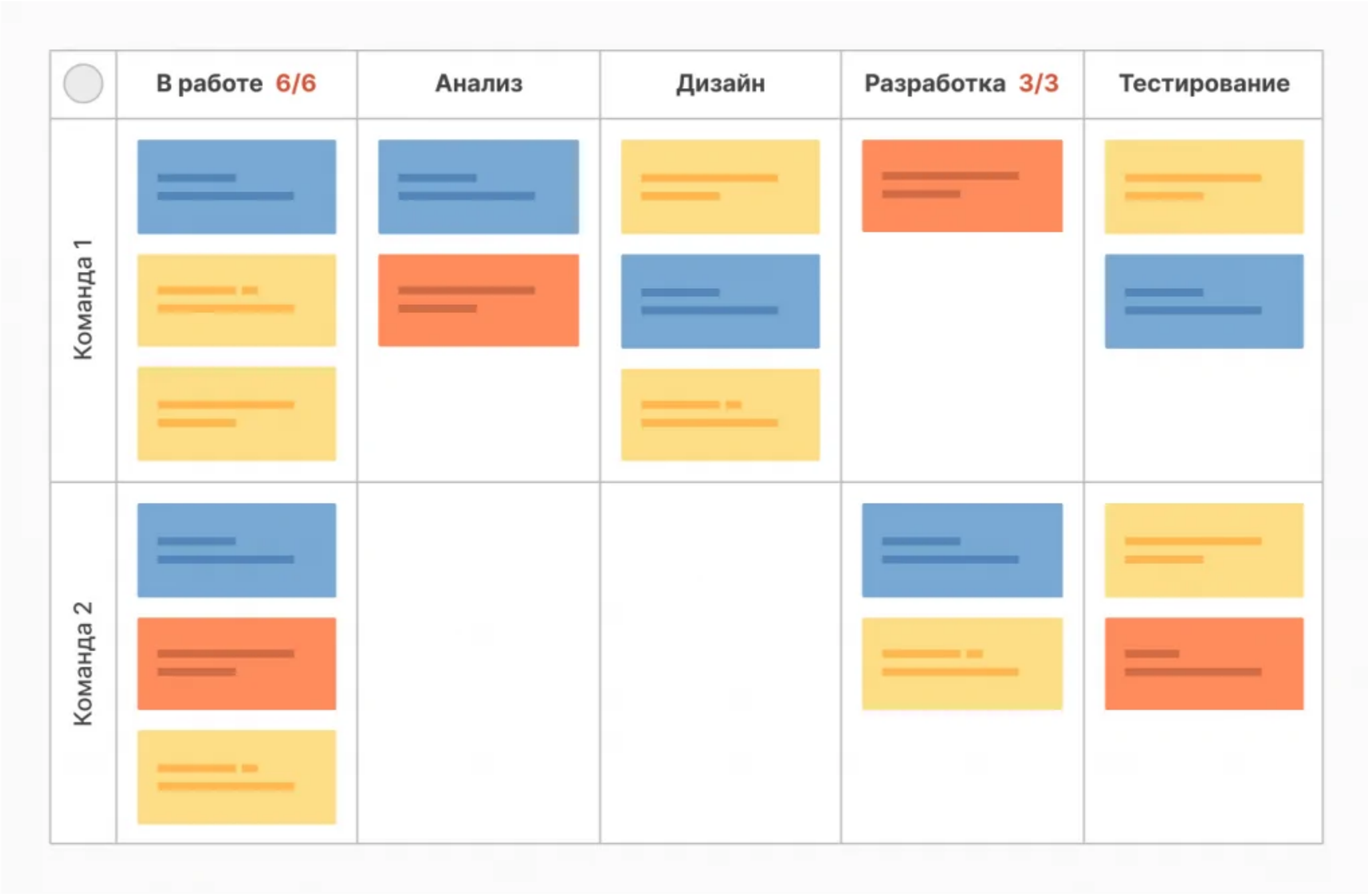
Scrum

- *Хорошо работает, если проект состоит из множества несвязанных относительно небольших задач и/или богатой, но простой в реализации функциональности*
- *Важно (и реалистично) делать частые релизы*
- *Есть стабильный канал обратной связи*
- *Команда состоит из «универсальных бойцов» (и задачи «эксклюзивными» не бывают)*

KANBAN

KANBAN

- Контролируем каждую задачу
- Общая идея - минимизация времени реализации каждой задачи
- Отлично работает, когда команда гетерогенная



Плохой пример

Kanban

Допустим, мы осуществляем поддержку сложного внедрения ПО (и исправляем выявленные ошибки)

Плохой пример

Kanban

Допустим, мы осуществляем поддержку сложного внедрения ПО (и исправляем выявленные ошибки)

Казалось бы, отличный пример для Kanban: заведем доску, будем «регистрировать ошибки» - «исправлять» - «проверять» - «выпускать»

Плохой пример

Kanban

Допустим, мы осуществляем поддержку сложного внедрения ПО (и исправляем выявленные ошибки)

Казалось бы, отличный пример для Kanban: заведем доску, будем «регистрировать ошибки» - «исправлять» - «проверять» - «выпускать»

И при этом у нас разные люди будут работать на разных этапах, выполняя разные задачи.

Плохой пример

Kanban

Допустим, мы осуществляем поддержку сложного внедрения ПО (и исправляем выявленные ошибки)

В чем тогда проблема?

Плохой пример

Kanban

Допустим, мы осуществляем поддержку сложного внедрения ПО (и исправляем выявленные ошибки)

И при этом у нас разные люди будут работать на разных этапах, выполняя разные задачи.

То есть ответственного (крайнего, козла отпущения - тут уж как кому нравится) у нас нет.

Плохой пример

Kanban

Допустим, мы осуществляем поддержку сложного внедрения ПО (и исправляем выявленные ошибки)

И при этом у нас разные люди будут работать на разных этапах, выполняя разные задачи.

То есть ответственного (крайнего, козла отпущения - тут уж как кому нравится) у нас нет.

Нет ответственного - не с кого спросить, если будут какие-то проблемы

Плохой пример

Kanban

Допустим, мы осуществляем поддержку сложного внедрения ПО (и исправляем выявленные ошибки)

И при этом у нас разные люди будут работать на разных этапах, выполняя разные задачи.

То есть ответственного (крайнего, козла отпущения - тут уж как кому нравится) у нас нет.

Нет ответственного - не с кого спросить, если будут какие-то проблемы

Отсюда получаем низкую удовлетворенность пользователями.

Плохой пример

Kanban

- *Плохо работает, если у нас нет сильного менеджмента*
- *Не подходит для проектов со сложными и/или объемными задачами*
- *Требует умения крайне быстро принимать решения - как управленческие, так и проектные*

Хороший пример

Kanban

Допустим, мы делаем дополнения к игре (например, дополнительный контент)

Хороший пример

Kanban

Допустим, мы делаем дополнения к игре (например, дополнительный контент)

- У нас есть четкое разделение этапов и ролей («геймдизайнер» - «дизайнер» - «программист» - «тестировщик»)
- Мы хотим добиться того, чтобы запланированные (и обещанные!) дополнения были выпущены точно в срок (например, они могут быть привязаны к каким-то датам)

Хороший пример

Kanban

Допустим, мы делаем дополнения к игре (например, дополнительный контент)

- У нас есть четкое разделение этапов и ролей («геймдизайнер» - «дизайнер» - «программист» - «тестировщик»)

- Мы хотим добиться того, чтобы запланированные (и обещанные!) дополнения были выпущены точно в срок (например, они могут быть привязаны к каким-то датам)

Нам сильно проще контролировать процесс и управлять им - очень легко и просто отслеживать множество задач и их статус

Хороший пример

Kanban

- *Хорошо работает, когда над проектом трудятся люди разных профессий и есть четкое разделение этапов*
- *Когда у нас есть возможность выпускать много версий, и при этом хочется соблюдать строгие сроки по каждой задаче*
- *Хорошо показывает себя на проектах, в которых очень много небольших задач, и нет очень больших*

>>: tbc…