Agile, Scrum, waterfall

Java Collections
8 уровень, 2 лекция

ОТКРЫТА

— Итак, я хочу рассказать тебе про **Agile** и **Scrum**.

В начале 21 века представление о программировании перевернулось.

Т.к. все убедились, что долгосрочное планирование не работает, было решено вообще от него отказаться.

— Это как же?

— А вот так.

Был придуман максимально гибкий подход к управлению работой.

Основные концепции **гибкой разработки:**

- люди и взаимодействие важнее процессов и инструментов;
- работающий продукт важнее исчерпывающей документации;
- сотрудничество с заказчиком важнее согласования условий контракта;
- готовность к изменениям важнее следования первоначальному плану.

Принципы быстрой разработки:

- удовлетворение клиента за счёт ранней и бесперебойной поставки ценного программного обеспечения;
- приветствие изменений требований даже в конце разработки (это может повысить конкурентоспособность полученного продукта);
- частая поставка рабочего программного обеспечения (каждый месяц или неделю или ещё чаще);
- тесное, ежедневное общение заказчика с разработчиками на протяжении всего проекта;
- проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием;
- рекомендуемый метод передачи информации — личный разговор (с глазу на глаз);
- работающее программное обеспечение — лучший измеритель прогресса;
- спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределённый срок;
- постоянное внимание к улучшению технического мастерства и удобному дизайну;
- простота — искусство не делать лишней работы;
- лучшие технические требования, дизайн и архитектура получаются у самоорганизованной команды;
- постоянная адаптация к изменяющимся обстоятельствам.

Основной проблемой разработки программы было признано то, что никто из участников ни на одном этапе не обладает всей полнотой информации о том, что делать.

Заказчик может рассказать, как он видит программу, но что-то он упускает, что-то считает самим собой разумеющимся.

Менеджер вообще должен переводить требования с языка программистов на язык заказчика и обратно.

Слишком много неопределённости.

Часто требования заказчика выглядят так: сделаете как-нибудь, потом покажите мне, если мне не понравится – переделайте.

— М-да. Как все запущено.

НАЧАТЬ ОБУЧЕНИЕ

— А в чем разница?

— Ну смотри, допустим, раньше на разработку программы уходил год. И только через первых полгода было уже на что посмотреть. Это как строить большой дом: сначала котлован, потом фундамент, стены, крыша, отделка и т.д.

А теперь программисты стараются как можно раньше выпустить нужный заказчику функционал. Это как если бы строители сначала построили шалаш, затем времянку, затем небольшой домик, затем уже начали делать большой дом и тот – по частям.

Если учесть, что заказчик скорее всего сам точно не знает, чего хочет, то это очень разумный подход.

Допустим, хочет заказчик большой охотничий домик.

Построили ему маленький, он в нем пожил зиму и решил, что дом из дерева ему не нравится. Будем делать из кирпича в 4 слоя.

Пожил рядом с озером летом, а его комары заели. Он где-то слышал, что озеро – это круто, вот и хотел. А теперь ему озеро не надо. Так ведь и дом строить легче: нет озера – нет угрозы паводков, вот и можно строить дом не на сваях, а на земле, а это на 25% быстрее.

— Интересная аналогия. Неужто заказчик так часто меняет свои требования?

— Да дело тут не в заказчике.

Во-первых, очень сложно представить наперед, что получится. И менеджеры, и тестировщики, и программисты тоже бывают в такой роли. И тоже меняют свое мнение в зависимости от того, что получилось.

А во-вторых, разве требования заказчика – не главное? **Ведь смысл всей этой работы – это сделать именно то, что заказчику нужно, а не то, что он сказал в самом начале делать.**

Если в нем нет чего-то, что очень нужно заказчику, но про что забыли, никто это что-то делать не будет.

— Ясно. Работать по плану легче, но не все можно сделать по плану!

— Именно.

Поэтому были придуманы методики гибкой разработки.

И о самой популярно из них – **Scrum** — я тебе сегодня расскажу.

Основная фишка скрама – это разбиение разработки проекта на небольшие итерации – обычно 2-4 недели. Такая итерация называется «спринт».

Вначале спринта проводится «планирование спринта» — совещание часа на 3-4.

В конце проводится «демо» — демонстрация всех полностью завершенных задач.

Вот как обычно все происходит:

Перед самым первым спринтом, заказчик (или его представитель), формирует список требований – набор того, что должна уметь программа. Такие требования обычно называют «**user story**», а заказчика – «**product owner**».

Product owner переводится как владелец продукта, т.к. продукт пишется для него и он и только он определяет список требований – что надо, когда и в каком порядке.

Кроме того, product owner обычно еще назначает задачам приоритет. Сначала будут реализовываться задачи с самым высоким приоритетом. Весь список требований еще называют «**product backlog**» или «**резерв продукта**».

Когда начинается спринт, то все собираются на совещание. Ведет его обычно **scrum-master**: как правило это кто-то из команды. Цель совещания – отобрать задачи (**user story**) на текущий спринт (итерацию разработки).

Сначала каждой задаче дается коллективная приблизительная оценка в «абстрактных человеко-днях», их еще называют «story point». Затем команда решает, сколько задач она успеет сделать за этот спринт.

При этом команда сама решает, сколько задач она успеет сделать за спринт.

Допустим, **product owner** ожидал, что команда отберет 7 первых задач, а она отобрала всего 5, то задачи 6 и 7 переносятся на следующий спринт. Если **product owner** это не устраивает, он может повысить приоритет 6 и 7 задачи, чтобы их точно отобрали, но тогда из спринта выпадут какие-то другие задачи.

Также **scrum master** может предложить разбить часть задач на более мелкие и расставить им различные приоритеты, чтобы **product owner** оказался максимально довольным.

В этом и есть смысл совещания – задачи можно менять и разбивать, можно менять им приоритеты и т.д. Это именно та работа, которая не была видна в самом начале, но которая приносит очень много пользы.

— Понял, это как рулить машиной. Даже если в начале кажется, что надо ехать только прямо, то на деле надо постоянно объезжать ямы и рулить то вправо, то влево, обгонять и уступать дорогу.

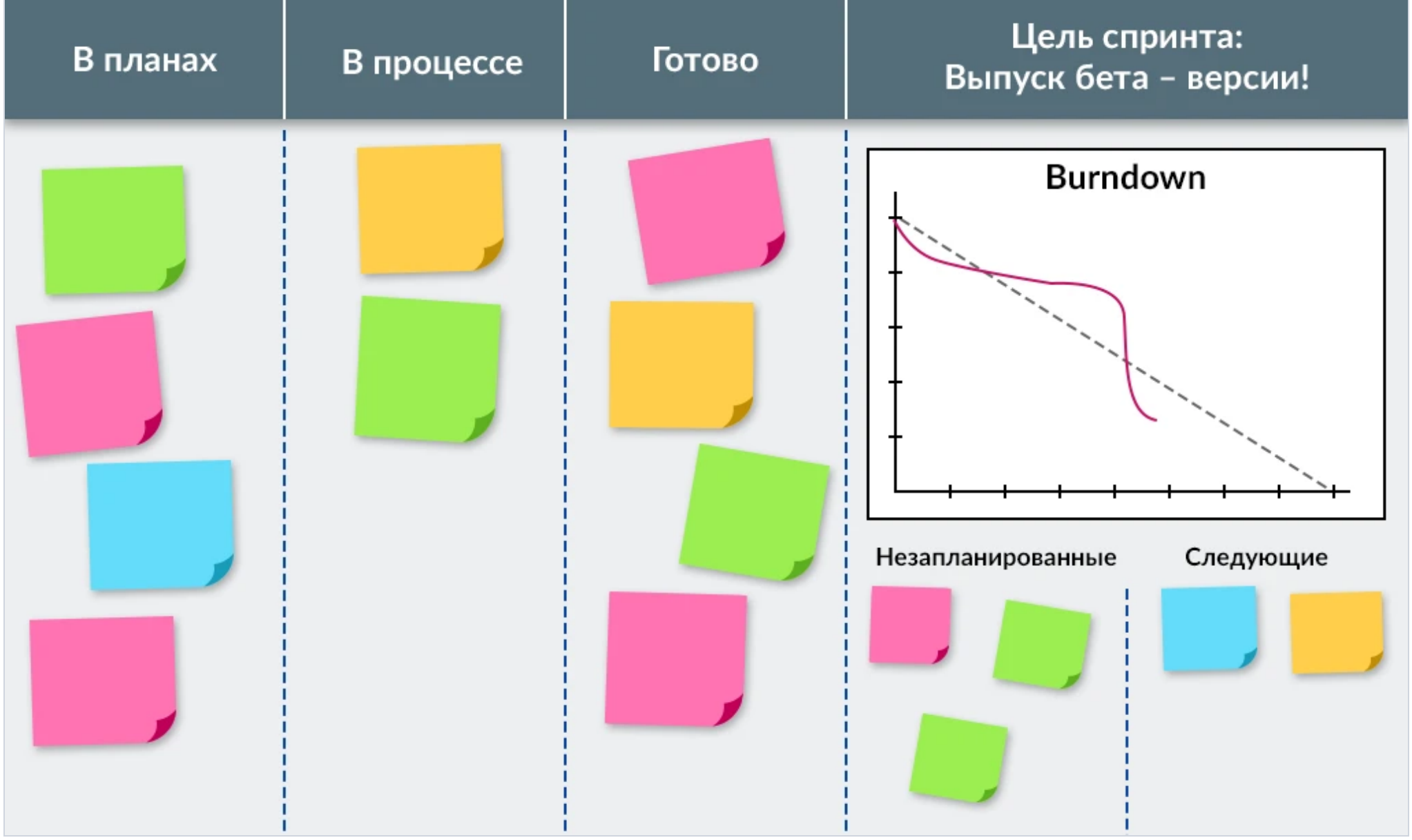
— Да, как-то так.

Список задач, отобранных для спринта, называют **sprint backlog** или **резерв спринта**.

Программисты распределяют, кто и что будет делать, а только затем приступают к работе. Для повышения эффективности, скрам предлагает, чтобы каждый день проводились совещания по 5-15 минут, где все бы рассказывали друг-другу, кто что сделал за прошедший день и чем будет заниматься сегодня.

— Командная работа. Уважаю!

— Для лучшей наглядности, обычно предлагают отображать текущее состояние спринта на специальной доске:



Обрати внимание на три колонки слева.

На стикерах пишутся краткие названия задач. А стикеры переклеиваются в другую колонку в зависимости от статуса (В планах, В процессе, Готово)

Справа ты видишь «диаграмму сгорания». Там по дням отображается список несделанных задач. В идеале за время спринта количество неготовых задач падает до нуля.

Когда спринт завершается, **scrum-master** проводит **demo**, на котором демонстрируется список всего, что полностью сделано.

Затем проводится «**разбор полетов**» — совещание тоже на пару часов. На нем обычно пытаются выяснить, что было сделано хорошо, а что (и как) можно было сделать лучше.

Обычно за 2-3 спринта можно выявить основные проблемы, которые мешают команде работать эффективнее, и устранить их. Это приводит к большей продуктивности, не увеличивая нагрузку на команду. Такое было невозможным до эры гибких методологий.

В середине спринта, иногда еще проводят «вычесывание» — совещание посвященное планированию следующего спринта. На нем обычно уточняют приоритеты задач, а так же можно разбить некоторые задачи на части и/или добавить новые задачи в **product backlog** – резерв продукта.

В принципе у меня все – это обзорная лекция, и на пальцах ее не объяснить, но вот тут ты можешь почитать пару хороших статей на эту тему:

http://scrum.org.ua/wp-content/uploads/2008/12/scrum_xp-from-the-trenches-rus-final.pdf

http://ru.wikipedia.org/wiki/SCRUM

+55

Комментарии (20) популярные новые старые

JavaCoder

Введите текст комментария

LuneFox инженер по сопровождению в **BIFIT** EXPERT — 8 марта, 06:51

Стыд и скрам.

Ответить +4

Vladrip Уровень 41, Ивано-Франковск, Украина — 17 октября 2021, 16:24

♂Scrum master♂

Ответить +5

Артур Харатян Java Developer — 20 января 2021, 14:59

— Итак, я хочу рассказать тебе про Agile и Scrum.
Но расскажу только про скрам...

Ответить +12

Anna Avilova architect — 27 апреля, 11:34

я так понимаю, что принцип гибкости (Agile), подразумевается априори при применении принципов Scrum... а вот про Waterfall, заявленный в названии лекции, хотелось бы хоть пару слов услышать))) ... ну, к этому уровню, думаю, уже все привыкли

Ответить +3

Pavlo Buidenkov Уровень 41 — 29 мая 2020, 22:44

как жаль что на практике в разработке ПО многие забывают, что отцами-создателями Agile (Kent Beck, Martin Fowler, Robert T. Martin ...) была четко освещена важность юнит тестов кода и важность TDD (Test Driven Development).
Если природа Agile проекта это постоянная изменяемость, то юнит тесты и хороший QA это как подушка безопасности, если этого нет, то проект который чуть сложнее простого CRUD просто обречен на провал. Разработчики просто прогорают в таких конторах и увольняются.

Эта картинка с побитым менеджером и есть пример что происходит на практике если тебе 3-infinity раза за твой 2-х недельный спринт менеджер прийдет и скажет что что-то нужно поменять. Без юнит тестов ты можешь даже потерять уже сделанную работу и привнести новые баги.
Юнит тестами конечно все баги не отловишь и тут нужен хороший QA желательно автоматизированный.

на заметочку:
валидатор javarush написан чуть более чем полностью с использованием JUnit. Создатель JUnit - Kent Beck он же один из евангелистов Agile и TDD. Это ни о чем не говорит?

Создание таких крутых проектов как javarush это "не поле перейти", и тут нужны знание best practices как в архитектуре так и в простых вещах как Unit Testing

...

совет на будущее как отличить хорошую компанию/команду от плохой:

(сначала самому выучить TDD и BDD методологии разработки, отдельно архитектуру проектов (гугли System Design Primer))

на интервью спросить:
1. пишут ли они в команде юнит тесты (TDD в идеале)
2. Как осуществляется QA (BDD в идеале)
3. как много внимания уделяют рефакторингу кода.
4. как они реагируют на feedback от заказчика (тут scrum, все дела должно быть адекватно сформулировано)
5. как они осуществляют логирование и сбор логов если клиент жалуется на ошибки в продукте
6. вообще спросить какие инструменты разработки они используют (Git, CI/CD tools, Harbour, Docker, Sonarqube ... если не используют Git то вообще зашквар! бегите оттуда!)

MartyMcAir Уровень 41, Россия — 19 марта 2020, 19:28

Не знаю в тему не в тему: четыре часа разработки сократили до 30 минут...
(нооо там дальше заголовок, вообщем не понравился..,
ну да понятно, что экономия, надо всего да побольше да лучше за бесплатно и прямо сейчас.. и т.д...)
впрочем ожидаемо..

Ответить 0

Vitaly Khan Java Developer в **Onollo** MASTER — 4 января 2020, 03:28

самое известное видео про Agile (15 минут)

Ответить +20

Mike Уровень 39, Москва, Россия — 15 июля 2019, 22:38

Про водопадик ничего не сказали)

Ответить +6

Andrii Gorshunov Уровень 41, Польша EXPERT — 17 апреля 2019, 21:35

Это надо знать программисту? В смысле тут подана книга на 100 страниц. Есть смысл ее читать для будущего программиста?

Ответить +3

skybright Project Manager в **construction company** EXPERT — 22 ноября 2019, 10:52

я прочитал, думаю имеет.
Это займет пару часов, но зато вы будете лучше представлять, как сейчас работают команды.

Ответить +1

Жора Нет Уровень 39, енакиево, Украина — 12 мая, 11:08

100 страниц за пару часов?😮
Я так не могу...у меня, наверное дня 2 уйдет😔

Ответить 0

Ilya Sakharov Уровень 41, Москва — 25 октября 2018, 00:01

Хе-хе)
Как все наивно описано)
Детки, чудес не бывает! В реальности далеко не все такие мотивированные, ответственные и отлично знают свою предметную область, каждый митинг это радость, где решаются все проблемы и все благодаря scrum.
У scrum есть определенные преимущества и есть недостатки, не надо пихать его везде.

Ответить +3

Roman Уровень 32, Германия — 12 марта 2020, 13:03

А зачем вообще работать в компании, где люди не мотивированные и не ответственные?)

Ответить +1

Евгений Уровень 41, Санкт-Петербург, Россия — 30 августа 2020, 10:08

Добро пожаловать в реальную жизнь.

Ответить +4

alex_us Уровень 41, Симферополь — 30 января 2021, 15:26

по моему таких людей в любой отрасли большинство

Ответить 0

Андрей Enterprise Java Developer в **John Wiley and Sons** — 20 августа 2018, 16:18

JIRA - https://ru.atlassian.com/software/jira

Ответить +1

Показать еще комментарии

RUSH

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

ДОСТУПНО В Google Play

Загрузите в App Store