Гибкая методология разработки

Java Collections
8 уровень, 1 лекция

ОТКРЫТА

— Привет, Амиго!

— Привет, Билаабо!

— Сегодня я расскажу тебе о том, как обычно разрабатываются программы.

В 20 веке, когда современное ИТ только зарождалось, всем казалось, что программирование чем-то похоже на строительство или производство.

Обычно дело обстояло примерно так:

Заказчик рассказывал, какая программа ему нужна, что и как она должна делать.

Бизнес-аналитики выслушивали его и, на основе его историй, составляли полный список требований к программе.

Затем **менеджеры проекта** разбивали эти требования на задачи (таски), а также определяли, какой программист будет делать какую таску и в каком порядке.

Затем **программисты** приступали к работе. Иногда это могло занять несколько лет(!).

Когда работа была сделана, программу отдавали тестировщикам.

Тестировщики проходились по каждому пункту требований к программе и проверяли, что он реализован и программа работает как надо.

Если что-то было не так, тестировщики писали баги и отправляли их программистам.

Затем программисты фиксили (исправляли) эти баги и опять отправляли исправленную программу тестировщикам. И так по кругу.

Когда основные баги были исправлены, программа отдавалась заказчику.

— Что, так прям все и происходило?

— Ну, тут конечно многое упрощено, но дело почти так и было.

— И что, проект реально мог писаться полтора-два года?

— Иногда, когда разработка проекта была очень длительной, ее разбивали на отдельные релизы. Каждые 3-6 месяцев разработчики обязаны были сделать определенную часть функционала программы, протестировать ее, исправить все баги и показать заказчику.

Чтобы он мог внести свои замечания и, что более важно, продолжал платить деньги за разработку программы.

— Продолжал платить?

— Тогда разработка очень часто растягивалась в 2-3 раза. А т.к. оплата труда программистов часто была почасовая, то и стоимость программы вырастала в 2-3 раза. А вот польза от нее при этом падала. Ведь то, что нужно сегодня и за $100,000, совсем не обязательно будет нужно через 3 года, да еще за втрое большую цену.

— Т.е. заказчики часто отказывались платить?

НАЧАТЬ ОБУЧЕНИЕ

— А почему так?

— Ну, во-первых, на этапе планирования слишком многое не известно. Чаще всего, проблемы, с которыми сталкивались программисты, в самом начале не мог предсказать никто.

— Но ведь опытные программисты должны были бы сами все предвидеть, разве нет?

— Видишь ли, программирование – это уникальная профессия.

Обычный специалист очень часто выполняет одну и ту же работу. Часовщик делает часы, повар – готовит, учитель – учит, врач – лечит, и т.д.

Фактически каждый из них ежедневно занимается тем же, что и вчера. И поэтому начинает делать свою работу все лучше и лучше.

В программировании другой подход. Как только у программиста каждый день возникает одна и та же задача, он пишет для нее функцию/модуль/программу и больше к ней не возвращается.

Каждую свою задачу программист обычно решает один раз в жизни.

Чем-то похоже на ученого или на инженера-конструктора, который что-то изобретает.

— Ага. А какая роль в проекте самая важная?

— Ну, что тебе сказать. Легко сказать, какая самая важная, сложно сказать – какая самая не важная.

Основная задача тестировщиков (Quality Assurance, QA) – следить за состоянием работы программы и вовремя рапортовать о найденных ошибках. Чем больше и раньше тестировщик найдет ошибок, тем больше их будет исправлено. **Хороший тестировщик влияет на качество продукта больше, чем хороший программист.**

— А почему программист не может сам тестировать свою программу. Ведь он лучше знает, что в ней работает, а что – нет?

— Хороший программист просто не может быть хорошим тестировщиком. Программист хорошо знает, как устроена программа, поэтому пользуется ей всегда специфическим образом. В отличие от пользователя, который пользуется программой, как бог на душу положит.

Кроме того, тестировщик не тестирует то, что еще не работает. Тестировщик тестирует ту функциональность, те части программы, которые по мнению программиста уже работает и работают чуть ли не идеально.

И вот когда тестировщик находит там горы ошибок, а программист их исправляет, продукт действительно становится ближе к идеалу.

Основная задача программиста (Software Developer Engineer, Developer, SDE) – реализовывать новую функциональность. Или, если быть проще – выполнять назначенные на него задачи. Назначили задачи с новыми фичами – делает их. Назначили баги – фиксит баги.

— Что-то у тебя одни англицизмы в разговоре. Каждое второе слово.

— Это не столько англицизмы, сколько профессиональный сленг. Привыкай.

Но иногда бывают задачи и посложнее, например, придумать архитектуру программы или ее части. Чем лучше будет предложенная архитектура, тем легче будет делать дальнейшую работу.

Проблема в том, что архитектуру надо выбрать в самом начале, а удачную ты выбрал архитектуру или нет, становится понятно не раньше середины разработки.

Более того, если архитектура была удачной и программа получилась отличной, то, скорее всего, на ее базе заказчик захочет выпустить новую версию программы.

Это я к чему.

Какую бы версию архитектуры программы ты ни выбрал, всегда найдется куча изменений, дополнений, новых фич, которые этой архитектурой не учитываются.

Заказал клиент построить дом на 5 этажей, ты придумал архитектуру, и дом построили.

Затем заказчик говорит, а давайте достроим еще один этаж, а затем еще один и т.д.

А ведь стены первых этажей на такую нагрузку не рассчитаны, фундамент тоже, вот и приходится все переделывать.

А если заказчик после дома в 5 этажей захочет дом сразу в 50?

— Тогда легче снести то, что было и построить все заново…

— Но насчет архитектуры, у меня для тебя есть один совет.

Архитектура приложения в первую очередь должна быть гибкой и подразумевать, что при решении переделать половину программы не придется полностью переделывать вторую половину. Такую архитектуру обычно называют **гибкой и модульной**.

Основная задача менеджера – это принимать решения. Менеджер – это человек, который видит всю картину и принимает решения исходя из этого.

Допустим, в процессе разработки выяснилось, что некоторая задача не получится в таком виде, в каком ее хотят видеть. И тут менеджер может:

а) попробовать договориться с заказчиком, чтобы эту задачу поменяли;

б) выделить больше времени на эту задачу;

в) привлечь более опытных программистов с других проектов.

Тут еще много вариантов.

Дело в том, что в каждом из них придется чем-то жертвовать, а задача менеджера – свести суммарный ущерб от таких жертв до минимума.

Или, например, ведущего программиста перекупают конкуренты, предложив в два раза больше денег.

Менеджер может:

а) ничего не делать. Программист уходит, проект, вероятно, затянется и попадет на штрафные санкции.

б) поднять ему зарплату в два раза. Тогда остальные люди в команде тоже захотят себе повышения ЗП. Если всем им дать денег, расходы на проект вырастут, и он может стать убыточным.

в) свой вариант.

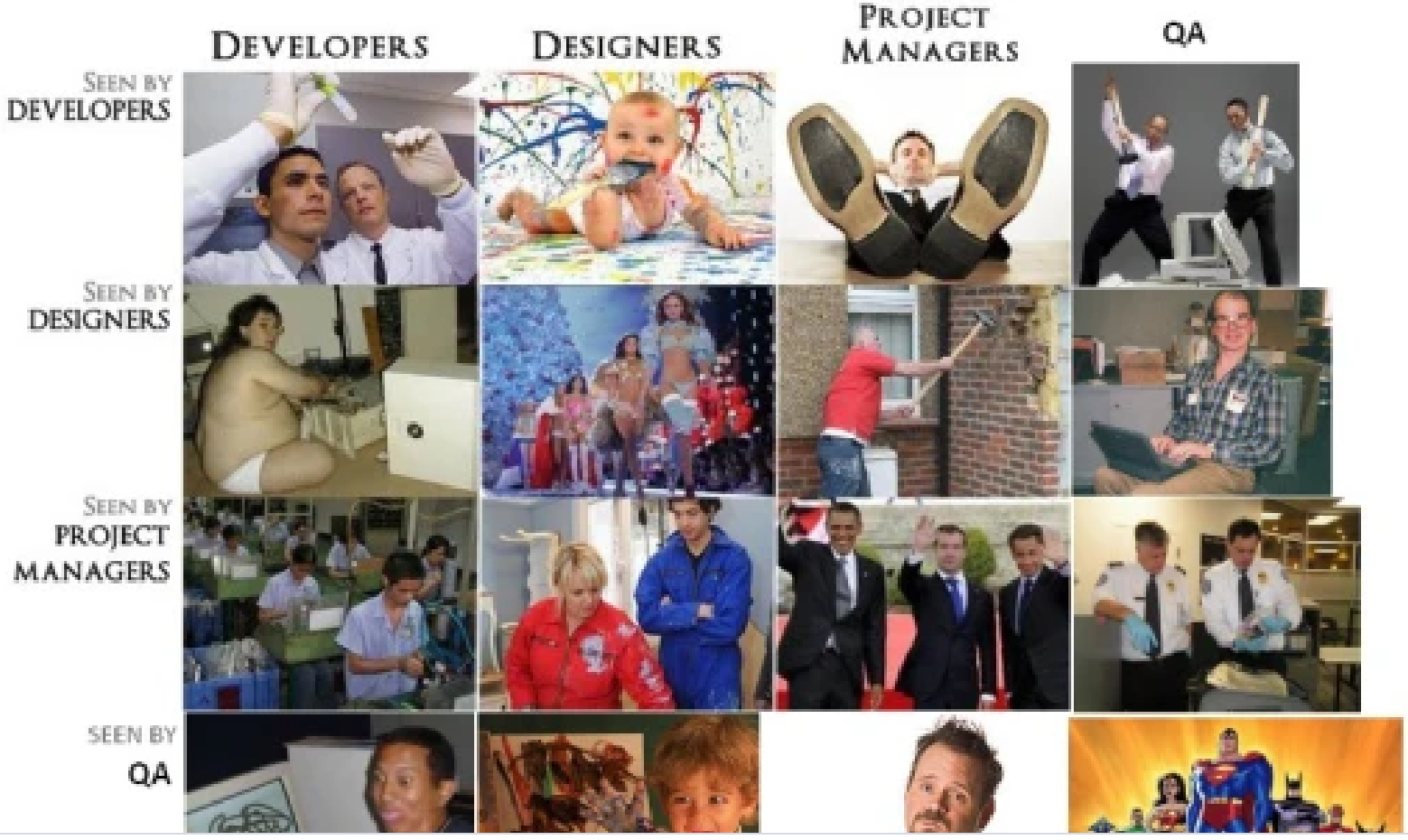
— Ясно.

— Обычно, когда плохой менеджер, проект может вытянуть хорошая команда, но не всегда.

Хороший менеджер и команда средних программистов почти всегда сделают проект быстрее, чем плохой менеджер и команда отличных программистов.

— Ясно.

— Вот как выглядят эти профессии глазами друг друга:



‹ Предыду

×28

+70

Комментарии (23) популярные новые старые

JavaCoder

Введите текст комментария

Rustam A. Nagaev Уровень 41, Москва, Россия — 26 января, 18:46

"...тестировщики писали баги и отправляли их программистам..." 😂😂😂

Ответить +3

LuneFox инженер по сопровождению в **BIFIT** EXPERT — 8 марта, 05:47

Письмом. В конверте. В 3 часа ночи. Доставляли их, стучась в окно во время дождя.

Ответить +4

Ars Уровень 41 — 29 ноября 2021, 14:54

```
1  Архитектура приложения в первую очередь должна быть гибкой и подразумевать, что пр
```

Как и всё остальное в программирование, свойства архитектуры зависит от проекта. С одной стороны если всё пойдёт хорошо у компании, заказчик может захотеть ещё 100500 функций. И тогда масштабируемость и гибкость нужна.
С другой стороны заказчик работает на рынке стабильного размера, который очень медленно меняется и его доля стабильна. В ближайшие лет 5, он не планирует вообще никаких изменений. Здесь гибкость и масштабируемость нужна по минимуму.

Ответить +2

Justinian Judge в **Mega City One** MASTER — 11 февраля, 12:40

```
1  Здесь гибкость и масштабируемость нужна по минимуму.
```

Логично, поскольку про эджайл и скрам это методологии разработки ПО.
Нет разработки (проект на стадии саппорта) = нет необходимости в гибкой методологии разработки ПО, хотя канбан могут оставить

Ответить +2

Лиза Воренувкина Уровень 43, Кривой Рог, Ukraine — 26 сентября 2021, 14:17

:)

19 декабря 2020, 00:35

А ведь очень классная и интересная статья)

Ответить +4

wan-derer.ru Уровень 40, Москва, Россия — 16 декабря 2020, 23:23

Каждую свою задачу программист обычно решает один раз в жизни
(глядя на список задач JavaRush) Хмммм....

Ответить +2

wan-derer.ru Уровень 40, Москва, Россия — 16 декабря 2020, 14:08

Те же и клиенты:

Те же и одмины:

:)

Ответить +14

Interstellar Java Developer в **EPAM** EXPERT — 14 августа 2020, 14:36

Почему тестировщики на работе такие злые?

Ответить 0

wan-derer.ru Уровень 40, Москва, Россия — 16 декабря 2020, 23:21

Дерутся?

Ответить +1

MartyMcAir Уровень 41, Россия — 19 марта 2020, 18:29

Как то на хабре читал статью, про то как тестировщики изощраются, и + к этому приводят примеры оптимизаций с кучей графиков, статистики и прочего, тот объем материалов и работы который они проделывают иногда просто поражает..
Что хотелось бы добавить, ммм кажется, что если бы в мире в любой компании (это про не IT компании), был отдел тестирования и оптимизации рабочих процессов и всего остального, фишка в том что если бы, в любой компании на столько точно и досконально всё высчитывалось, оптимизировалось (настолько круто как и в IT), и тщательно собирало статистику и т.д.. вот это был бы космос..

Ответить +3

Roman Уровень 32, Германия — 12 марта 2020, 13:00

Наличие "тестировщиков" и "менеджеров" - это путь вникуда. В компаниях, в которых будет хотеться и будет приятно работать будущим выпускникам JavaRush, ни менеджеров, ни тестировщиков в том смысле, в котором они описаны в статье, не будет :)
Это я как бывший менеджер с почти 10 летним опытом говорю)) Ну и последние 2 компании, где я работал, имели ноль тестировщиков, потому что ответственность за качество кода несли те, кто его писали.

Ответить +3

Regina Software Developer EXPERT — 22 мая 2020, 00:23

причем тут тестировщики и ответственность за качество кода? 🤦‍♀️

Ответить +7

Vladimir QA Engineer в **Qulix** — 15 августа 2020, 13:11

На сегодняшний день в любой сколь-нибудь серьезной компании имеется свой отдел тестирования, либо, если компания небольшая, как минимум несколько QA инженеров. Насчет отсутствия менеджеров Вы, извиняюсь, тоже глупость сказали.

Ответить +3

Justinian Judge в **Mega City One** MASTER — 1 ноября 2020, 01:51

та да, логика интересная, зачем менеджеры когда люди сами честно могут эффективно работать, зачем врачи, когда пациенты могут не болеть, зачем учителя, если дети и студенты могут хорошо учиться 😀

Ответить +5

Alex T Уровень 35, Москва, Россия — 7 апреля 2021, 00:20

если тестировщиков не было, то их работу делали программисты... кгхм, а значит им недоплачивали)))

Ответить +3

Kirill Lvsov Уровень 40, Санкт-Петербург, Россия — 8 ноября 2019, 20:13

+5

finnadzorservice Уровень 36, Москва 24 сентября 2021, 03:49

начинать отсчет от нуля или единицы?

Ответить

+5

Показать еще комментарии