Верификация ПО

Обеспечение качества

К.А.Кулаков

Петрозаводск — 2017

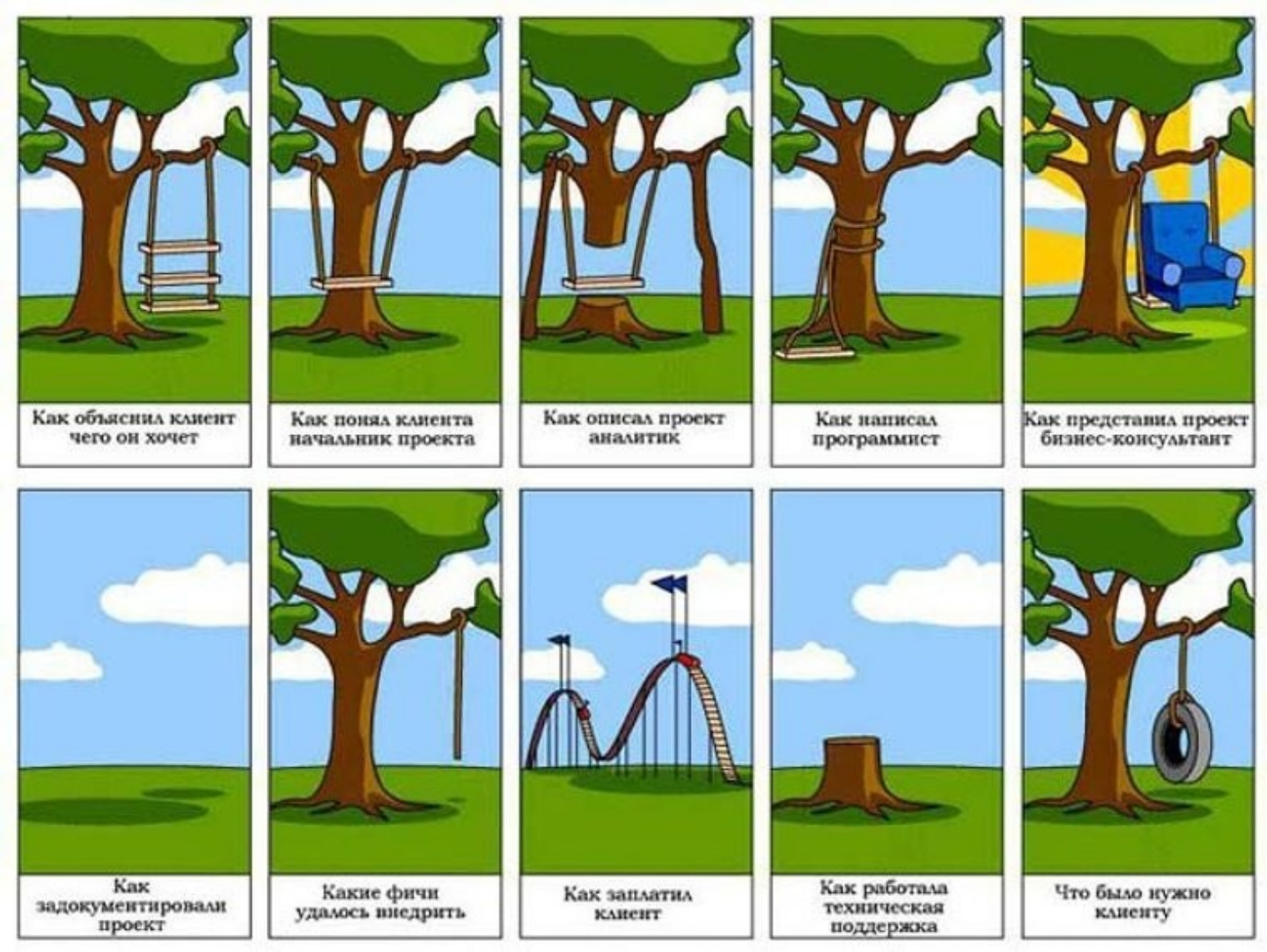
Определение

- ISO 9000:2000: Качество — совокупность свойств и характеристик продукции или услуги, которые придают им способность удовлетворять обусловленные или предполагаемые потребности потребителя
- ISO 9241-11: Usability — степень, в которой продукт может быть использован определенными пользователями при определенном контексте использования для достижения определенных целей с должной эффективностью, продуктивностью и удовлетворенностью.
- Понятийное: Fitness for purpose (пригодность для цели), Adequacy (соответствие ожиданиям), Satisfactoriness (ability to satisfy, удовлетворябельность)

Модель качества программного продукта

- ISO/IEC 25010:2011 (ГОСТ Р ИСО/МЭК 25010-2015) — Systems and software Quality Requirements and Evaluation (SQuaRE):
  - Функционально пригоден
  - Производителен
  - Совместим со средой эксплуатации
  - Удобен
  - Надёжен
  - Защищён
  - Сопровождаем
  - Переносим

Субъективность качества



Качество и ТЗ

- Нет непротиворечивого ТЗ со всеми нюансами
- Стоимость разработки и изменения ТЗ очень высока
- Если ТЗ правильное, то получится ли качественный продукт с точки зрения заказчика? А пользователя?

- Степень в которой продукт может быть использован определенными пользователями при определенном контексте использования для достижения определенных целей с должной эффективностью, продуктивностью и удовлетворенностью

Обеспечение качества

- **Обеспечение качества (Quality Assurance - QA)** - это совокупность мероприятий, охватывающих все технологические этапы разработки, выпуска и эксплуатации программного обеспечения (ПО) информационных систем, предпринимаемых на разных стадиях жизненного цикла ПО, для обеспечения требуемого уровня качества выпускаемого продукта.

Характеристики качества

- **Функциональность** (Functionality) - определяется способностью ПО решать задачи, которые соответствуют зафиксированным и предполагаемым потребностям пользователя, при заданных условиях использования ПО. Т.е. эта характеристика отвечает за то, что ПО работает исправно и точно, функционально совместимо, соответствует стандартам отрасли и защищено от несанкционированного доступа.
- **Надежность** (Reliability) – способность ПО выполнять требуемые задачи в обозначенных условиях на протяжении заданного промежутка времени или указанное количество операций. Атрибуты данной характеристики – это завершенность и целостность всей системы, способность самостоятельно и корректно восстанавливаться после сбоев в работе, отказоустойчивость.

Характеристики качества

- **Удобство использования** (Usability) – возможность легкого понимания, изучения, использования и привлекательности ПО для пользователя.
- **Эффективность** (Efficiency) – способность ПО обеспечивать требуемый уровень производительности в соответствие с выделенными ресурсами, временем и другими обозначенными условиями.
- **Удобство сопровождения** (Maintainability) – легкость, с которой ПО может анализироваться, тестироваться, изменяться для исправления дефектов, для реализации новых требований, для облегчения дальнейшего обслуживания и адаптироваться к имеющемуся окружению.
- **Портативность** (Portability) – характеризует ПО с точки зрения легкости его переноса из одного окружения (software/hardware) в другое.

Модель качества ПО

Контроль качества

- Контроль качества (Quality Control - QC) - это совокупность действий, проводимых над продуктом в процессе разработки, для получения информации о его актуальном состоянии в разрезах:
- готовность продукта к выпуску,
- соответствие зафиксированным требованиям,
- соответствие заявленному уровню качества продукта.

- Тестирование программного обеспечения (Software Testing) - это одна из техник контроля качества

Инженер контроля качества

- Тестировщик
  - Находит существующие дефекты
  - Работает только с самим продуктом
  - Работает только на стадии разработки и сдачи проекта
- Инженер контроля качества (QA Engineer)
  - Предотвращает появление дефектов
  - Настраивает процессы в компании
  - Работает на всех стадиях: от разработки ТЗ до анализа удовлетворенности клиента

Функции тестировщика

- Тестирование на соответствие ТЗ
- Поиск очепяток
- Ручное дымовое и регрессионное тестирование перед релизами
- Проверка граничных значений

- ... допекать программистов глупыми вопросами и придирками

Функции инженера контроля качества

- Участие в коммуникациях с заказчиком
- Сбор и актуализация требований (и их изменений!)
- Гарантия качества релизов
- Continuous Integration/Delivery
- Поддержка базы знаний в актуальном состоянии
- Контроль логики проекта/стандартов UI
- Автоматизация тестирования Unit Tests/UI
- Выработка метрик качества для регулярных процессов

Зачем программистам QA?

- Особый тип мышления – критичность и педантизм, стремление к порядку.
- Программист, читая ТЗ думает – как будем ДЕЛАТЬ, а QA – как будем СДАВАТЬ. Трудносовмещаемые мышления.
- Умение адаптироваться под сроки, заказчика, продукт. А не просто следовать правилам.
- Умение брать на себя огромную ответственность за продукт (и за всю свою команду). Команда – как за стеной.

- Помните. QA часто участвуют в решениях по сотрудникам (в т.ч. Начислении премий).

Обязанности программиста

- Думать головой прежде чем кодить, внимательно читать ТЗ
- Знать стандарты кода
- Знать стандарты UI
- Самоконтроль граничных значений и грамотная обработка исключительных ситуаций
- Выстраивать хорошую (unit-testable) архитектуру
- Пытаться выполнять основное тестирование самостоятельно. В идеале QA должны убеждаться в вашем качестве (долой перепинывание ошибок туда-сюда)

Программные инструменты

- Использование автоматических средств проверки правописания (IDE spellcheckers)
- Использовать средства ускорения рефакторинга (Resharper/Visual Assist)
- Процедура CodeReview (Upsource, CodeCollaborator) с двухуровневой системой:
  - Программиста проверяет тимлид
  - Тимлида периодически проверяют архитекторы
- Статические анализаторы кода (Resharper, PVS Studio)
- Модульное тестирование (NUnit, MSTest)

Организация по обеспечению качества

1) Договоритесь об общих шаблонах
2) Определите последовательность действий
3) Убедитесь, что стандарты и процессы используются
4) Проводите анализ выполненных проектов
5) Анализируйте и учитесь, используя данные дефектов
6) Используйте то, что вы изучили

Общие шаблоны

- **Общие шаблоны** предоставляют всем членам команды важную основу для сотрудничества. Когда каждый человек выполняет задачу своим способом, о сотрудничестве можно забыть.
- **Виды деятельности по Контролю Качества** (анализ, рецензии и тестирование) принесут больше пользы и будут более продуктивны, если продукт был сделан, используя общую модель.
- **Общие шаблоны способствуют улучшению технической работы.** Разработчик, выполняющий задачи своим собственным способом, может с легкостью пропустить важные детали или информацию. Когда работа стандартизирована, не возникает вопросов, что проделанная работа должна в себя включать.

Общие шаблоны

- **Стандарты должны применяться** при написании тест-планов, спецификаций, пользовательских интерфейсов, документации, тренинговых материалов и других продуктов, т.к. общее видение того, как проект должен быть сделан, **может помочь обеспечить его качество**. Но наряду со стандартами, необходимо определить ситуации их использования и разработать руководство по адаптации стандартов под нужды организации, если это необходимо. Любой стандарт, который вы принимаете, **должен помочь вам выполнять свою работу** как можно лучше и **не должен связывать вам руки**.

Определение последовательности действий (процессов)

- Видимые **высококачественные процессы** способствуют более плавному течению вещей. Они стимулируют повышение мастерства, позволяя гибко адаптироваться к уникальным нуждам каждого проекта. Другими словами, они **отвечают нуждам проектов**.
- Если вы выполняете свои задачи **ненадлежащим образом** и непоследовательно, **игнорируя договоренности**, то **никакой пользы** даже от хороших процессов вы **не получите**. Что означает последовательное использование процессов Обеспечения Качества? Это когда каждый человек четко умеет им следовать, знает, когда и как это делать и **строго их соблюдает**. Естественно, что такое **поведение ожидается от всей команды**...

Использование Стандартов и Процессов

- Чтобы получить пользу от использования внедренных стандартов и процессов, вы должны **постоянно контролировать**, что все делается согласно установленной договоренности, и вы **получаете именно тот результат**, который планировали.
- **Интегрированная модель зрелости процессов** программного обеспечения (CMMI - Capability Maturity Model Integration) реализует это при помощи аудитов (CMMI определяет аудит, как вид деятельности по Обеспечению Качества, потому что данная модель тестирует процессы, а не продукт).

Использование Стандартов и Процессов

- Если вы сталкиваетесь с ситуацией, когда принятый стандарт или процесс игнорируется, то необходимо выяснить, почему так происходит. Причины могут быть абсолютно разные:
  - Человек просто забыл использовать какой-либо стандарт или процесс.
  - Человек просто не знал о существовании стандарта или процесса или не знал, как именно его использовать
    - улучшите коммуникации в команде или проведите тренинг
  - Стандарт или процесс не подходит для данной задачи
    - либо адаптируйте сам процесс, либо попробуйте найти альтернативный способ
  - Стандарт или процесс был неэффективным или слишком «громоздким» для данной ситуации
    - упростите его так, чтобы он отвечали нуждам проекта.
- Каждое нарушение стандарта или процесса – это возможность его изучить и улучшить, чтобы он соответствовал нуждам команды!

Анализ выполненных проектов

- **Изученные уроки** (Lessons Learned, Post Project Analysis) - это один из самых мощных инструментов упреждающего улучшения качества вашей работы.
- **Ретроспектива** – это отдельно выделяемый отрезок времени, с целью обратить свой взгляд на проделанную работу, изучить полученный опыт и задать себе следующие вопросы: "Что было хорошо и как сделать так же в будущем?" и "Что было не так и как этого можно избежать?«
- Несмотря на то, что ретроспективы относят к **лучшим практикам** (Best Practises), используются они крайне редко, потому как сложно собрать всю команду: семинары по ретроспективе происходят в конце разработки проектов. Большинство членов команды уже работают на других проектах, а те, кто остались, заняты релизом проекта или его поддержкой.

Анализ выполненных проектов

- Не стоит делать всего лишь одну ретроспективу в конце проекта - необходимо делать ретроспективы на всем его протяжении.
- Преимущества:
  - Члены проекта всегда доступны, потому что они все еще закреплены за проектом
  - Ежемесячный семинар по итогам, например, одной фазы проекта гораздо легче запланировать и он займет всего около часа (вместо целого дня).
  - Весь полученный опыт и наработки все еще свежи в памяти, и вряд ли что-то будет упущено.
  - И самое важное – полученные уроки можно применить к оставшейся части проекта.

Использование данных дефекта

- Информация по дефектам – это просто **золотая жила**. Это записи о просчетах в качестве, а значит – список возможностей для улучшения качества на ваших будущих проектах.
- Информация о дефектах, которая может быть полезна для улучшения качества, включает следующие вопросы:
- Что было не так? Решать нужно саму проблему, а не ее симптомы. Например, зацикливание - это симптом, а изменение индекса цикла - это проблема.
- Когда была создана эта проблема? Какое именно действие при разработке явилось ее источником? Это была проблема в требованиях? Проектировании системы? Коде? Тестировании?

Использование данных дефекта

- Информация о дефектах, которая может быть полезна для улучшения качества, включает следующие вопросы:
  - Когда проблема была выявлена? Может она и не была сразу же устранена, но что нас интересует, сколько она существовала до того, как мы ее обнаружили.
  - Каким образом была найдена эта проблема? Способ обнаружения можно внедрить в постоянно используемую практику. Можно ли было обнаружить ее раньше? Есть ли какой-либо процесс Контроля Качества, который мог бы ее выявить, будь он эффективнее?

Использование данных дефекта

- Информация о дефектах, которая может быть полезна для улучшения качества, включает следующие вопросы:
  - Сколько стоило устранение этой проблемы? Этот момент очень легко недооценить. Убедитесь, что вы учли процесс диагностики проблемы и всю работу по ее устранению, которую вам пришлось сделать, включая ре-дизайн, переписывание кода, ре-компиляцию, переработку тестов, повторное тестирование, повторный релиз, выпуск заплатки, формирование отчета по дефекту, отчет по статусу проекта и т.д.
  - Какого рода была эта проблема? Когда у вас огромное количество дефектов, их категоризация облегчает анализ и обучение.

Использование полученных знаний

- Вы должны особым образом применять на практике все то, чему вы научились.
- Например, если ваш процесс разработки дизайна неэффективен для использования на определенных видах проектов, то тогда вы должны разработать альтернативный процесс и использовать его на всех будущих проектах такого типа.
- На регулярной основе (ежегодно, ежемесячно, ежедневно) рассматривайте новые возможности улучшения ваших стандартов, процессов и методов и вносите необходимые корректировки.
- Перед началом нового проекта просмотрите базу знаний накопленного опыта (Lessons Learned) и историю дефектов, и определите, что должно быть улучшено, основываясь на анализе прошлых проектов.
- Ответьте на вопрос: «Какие действия можно предпринять в этот раз, **чтобы обеспечить лучшее качество продукта**, чем то, которое было достигнуто до этого?»

План проведения мероприятий по обеспечению качества ПО

- Исследуйте все процессы, с которыми работает организация.
- Составьте список всех стандартов и процессов, инструкций, шаблонов и отметьте те, которые нужно доработать или изменить.
- Так же напишите список того, что нужно добавить в существующий процесс.
- Каждое планируемое изменение должно быть тщательно продумано и обосновано.

План проведения мероприятий по обеспечению качества ПО

- Устройте митинг с руководителями тех групп, где необходимо что-то изменить.
- Очень важно разъяснить то, что надо изменить, что надо убрать и что надо дописать, а главное зачем это надо.
- Не бойтесь признать, что какие-то изменения, запланированные вами, можно пересмотреть или отказаться от них, вообще, в случае если доводы сотрудников компании-заказчика перевешивают ваши (возможно вы не очень хорошо обосновали вашу точку зрения или ваше предложение действительно не даст необходимой выгоды).

План проведения мероприятий по обеспечению качества ПО

- Все изменения проводите как можно аккуратнее, так как любое неловкое движение может повлечь за собой необратимые последствия.
- Люди свойственны привыкать к чему либо, поэтому любое изменение может доставить массу неудобств. Поэтому не начинайте менять сразу все, вводите новое постепенно, при необходимости разъясняя сотрудникам компании-заказчика зачем это надо.

План проведения мероприятий по обеспечению качества ПО

- Проверяйте, что предложенные Вами изменения выполняются и приносят должный результат.
- Требуйте от Ваших сотрудников четкого следования процессам, инструкциям, а также использования шаблонов.
- Не останавливайтесь на достигнутом.
- Продолжайте поиск того, что можно улучшить для создания более качественного продукта, на основании новых уже внедренных процессов, стандартов, инструкций, шаблонов.

Метрики по обеспечению качества

- Метрика - это количественный масштаб и метод, который может использоваться для измерения.
- Введение и использование метрик необходимо для улучшения контроля над процессом разработки, а в частности над процессом тестирования.
- Типы метрик
  - Метрики по тестовым случаям (Test Cases)
  - Метрики по «багам» / дефектам
  - Метрики по задачам

Метрики по тестовым случаям

- Passed/Failed Test Cases
  - Метрика показывает результаты прохождения тест кейсов, а именно отношение количества удачно пройденных к завершившимся с ошибками. В идеале, к концу проекта, количество провальных тестов стремится к нулю
- Not Run Test Cases
  - Метрика показывает количество тест кейсов, которые еще необходимо выполнить в данной фазе тестирования. Имея данную информацию, мы можем проанализировать и выявить причины, по которым тесты не были проведены.

Метрики по багам

- Open/Closed Bugs
  - Метрика показывает отношение количества открытых багов к закрытым (исправленным и перепроверенным)
- Reopened/Closed Bugs
  - Метрика показывает отношение количества переоткрытых багов к закрытым (исправленным и перепроверенным)
- Rejected/Opened Bugs
  - Метрика показывает отношение количества отклоненных багов к открытым
- Bugs by Severity
  - Количество багов по серьезности
- Bugs by Priority
  - Количество багов по приоритету

Метрики по багам

- Метрики "Open/Closed Bugs", "Bugs by Severity" и "Bugs by Priority" хорошо визуализируют степень приближения продукта к достижению критериев качества по багам. Имея требования к количеству открытых багов, после каждой итерации тестирования мы сравниваем их с реальными данными, тем самым видя места, где нам нужно прибавить, для скорейшего достижения цели.
- Метрики "Reopened/Closed Bugs" и "Rejected/Opened Bugs" направлены на отслеживание работы отдельных участников групп разработки и тестирования.

Метрики по задачам

- Deployment tasks
  - Метрика показывает количество и результаты установок приложения. В случае, если количество отклоненных командой тестирования версий будет критически высоким, рекомендуется срочно проанализировать и выявить причины, а также в кротчайшие сроки решить имеющуюся проблему.
- Still Opened Tasks
  - Метрика показывает количество все еще открытых задач. К окончанию проекта все задачи должны быть закрыты. Под задачами понимаем следующие виды работ: написание документации (архитектура, требования, планы), имплементация новых модулей или изменение существующих по запросам на изменения, работы по настройке стендов, различные исследования и многое другое.