Введение в тестирование программного обеспечения

Тема 1.

ЦЕЛИ И ЗАДАЧИ ТЕСТИРОВАНИЯ ПО

Думай, как баг.
Делай, как баг.
И ты найдешь баг.

Несколько определений

Дефект (баг, глюк; defect, bug) – любое несоответствие фактического и ожидаемого результата (согласно требованиям или здравому смыслу).

Тест-кейс (test case) – набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства.

Тест-план (test plan) – часть проектной документации, описывающая и регламентирующая процесс тестирования.

Билд (build) – промежуточная версия программного средства (финальный бил часто называют релизом (release)).

Жизненный цикл ПО

Для более глубокого понимания места процесса тестирования в разработке ПО, познакомимся с этапами жизненного цикла ПО. Он включает:

- выработку требований;
- разработку спецификаций;
- общее проектирование;
- проектирование архитектуры;
- детальное проектирование;
- реализацию и кодирование;
- интеграцию;
- сертификацию;
- внедрение;
- сопровождение.

Жизненный цикл ПО: затраты

Аналитики отмечают, что распределение затрат по стадиям жизненного цикла ПО примерно таково:

- Анализ требований 3%
- Спецификация 3%
- Проектирование 5%
- Кодирование 7%
- Тестирование 15%
- Промышленное производство и сопровождение 67%

Продукты, подвергаемые тестированию

Тестировать можно (и нужно!):

- ✔ Программы при их непосредственном запуске и исполнении (software).
- ✔ Код программ без запуска и исполнения (code).
- ✔ Прототип программного продукта (product prototype).
- ✔ Проектную документацию (project documentation):
  - Требования к программному продукту (product requirements).
  - Функциональные спецификации к программному продукту (functional specifications).
  - Архитектуру (architecture) и дизайн (design).
  - План проекта (project plan) и тестовый план (test plan).
  - Тестовые случаи сценарии (test cases,).
- ✔ Сопроводительную документацию (и документацию для пользователей):
  - Интерактивную помощь (on-line help).
  - Руководства по установке (Installation guide) и использованию программного продукта (user manual).

Проверка соответствия программы требованиям, осуществляемая путем наблюдения за ее работой в специальных, искусственно созданных ситуациях, выбранных определенным образом.

ЧТО ТАКОЕ ТЕСТИРОВАНИЕ?

- Сэм Канер - «Тестирование – это поиск ошибок».
- Ли Копланд - «Тестирование – это сведение к минимуму риска пропуска ошибки».
- Крупнейший институт инженеров IEEE утверждает, что «Тестирование – это проверка продукта на соответствие требованиям».
- В некоторых источниках даже можно найти утверждения, что «Тестирование – это процесс, направленный на демонстрацию корректности продукта».

ЧТО ТАКОЕ ТЕСТИРОВАНИЕ?

Цель тестирования – помочь сделать качественный продукт и в намеченные сроки.

Исходя из цели, можно сказать, что Тестирование По – это процесс определения качества программного продукта.

Тестирование, в первую очередь зависит от величины проекта и модели его разработки.

ПРОЦЕСС ТЕСТИРОВАНИЯ

КАЧЕСТВО ПО

Качественный – значит соответствующий ожиданиям того, кому этот продукт предназначается.

Для этого нужны требования к продукту.

В зависимости от того, какие это требования – соответственно, нужно выбирать подход, как мы будем убеждаться, что они соблюдаются.

КАЧЕСТВО ПО

Следует помнить, что качество продукта определяется качеством процесса его разработки.

Некоторые рассуждения о качестве:

- Если заказчик доволен продуктом – продукт качественный.
- Если продукт соответствует требованиям – продукт качественный.
- У качественного продукта всегда есть какие-то преимущества и нет серьёзных недостатков.

КАЧЕСТВО ПО

Для того, что бы понять, что продукт соответствует требованиям пользователя и/или заказчика применяют верификацию и валидацию.

Верификация отвечает на вопрос: «Соответствует ли продукт требованиям?», а валидация: «Можно ли использовать продукт для определенных целей?»

Верификация – это процесс оценки системы или её компонентов с целью определения удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в начале этого этапа. Т.е. выполняются ли наши цели, сроки, задачи по разработке проекта, определенные в начале текущей фазы.

Валидация – это определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе.

ХАРАКТЕРИСТИКИ КАЧЕСТВА ПО

- **Функциональность** – это способность программного продукта (ПП) выполнять набор функций, определенных его внешними спецификациями.

- **Надежность ПП** – это способность безотказно выполнять заданные функции при заданных условиях в течение заданного периода времени с высокой степенью вероятности. Таким образом, надежность не означает безошибочности, для надежного ПП важно, чтобы ошибки появлялись при применении ПП достаточно редко и не приводили к катастрофическим последствиям.

- **Практичность** – это способность минимизировать затраты пользователя на подготовку и ввод исходных данных и оценку полученных результатов, а также вызывать положительные эмоции пользователя.

ХАРАКТЕРИСТИКИ КАЧЕСТВА ПО

- **Эффективность** – это отношение уровня услуг, предоставляемых ПП к объему используемых вычислительных ресурсов. Объем используемых вычислительных ресурсов количественно определяется затратами машинного времени и оперативной памяти на выполнение заданных функций.

- **Сопровождаемость** – это такие характеристики ПП, которые позволяют минимизировать усилия по внесению изменений при обнаружении ошибок в ПП и при его модификации. Не последнюю роль в повышении сопровождаемости играют комментарии к тексту программы!

- **Мобильность** – это способность ПП быть перенесенным из одной вычислительной среды (окружения) в другую, в частности, с одной ЭВМ на другую (применяют термин “перенос с одной платформы на другую”).

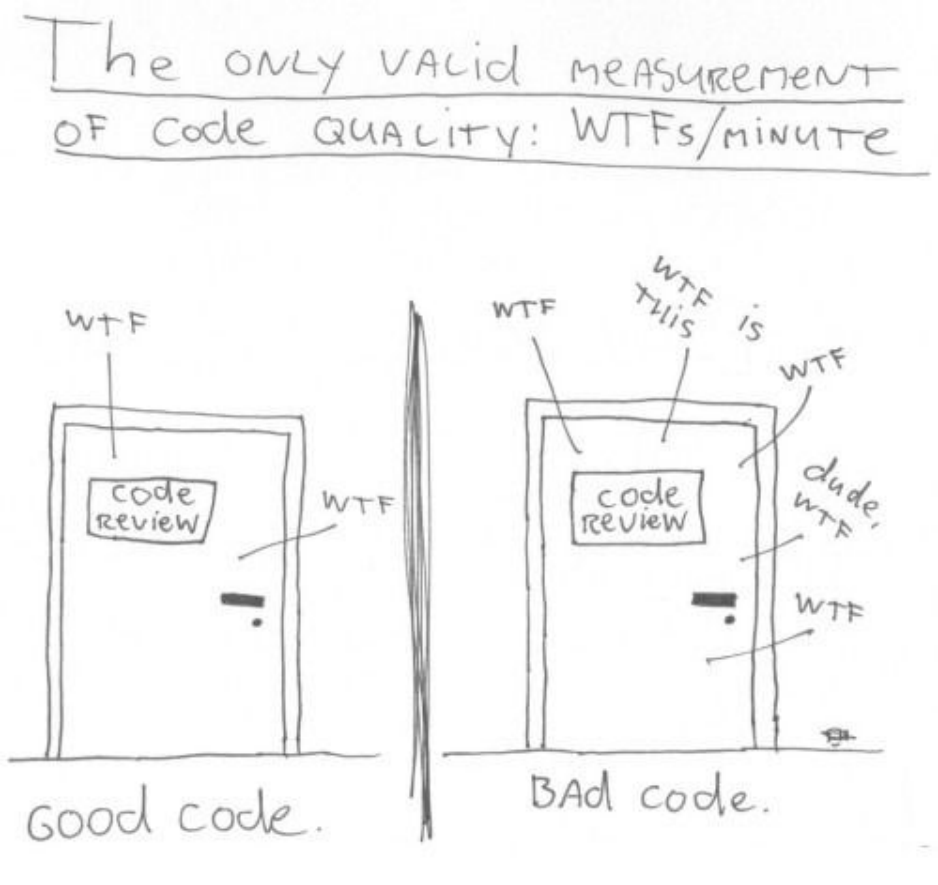
Функциональность и надежность являются обязательными критериями качества ПП, причем обеспечение надежности является неотъемлемой частью всех этапов и процессов разработки ПП.

КРИТЕРИИ КАЧЕСТВА

Ниже приведены лишь некоторые примеры того, как могут формулироваться критерии качества.

- Покрытие требований тестами – не менее 80%.
- Закрыто 100% известных критических дефектов, 90% дефектов средней критичности, 50% остальных дефектов.
- Общий показатель прохождения тестов – не менее некоторого значения: X = (Passed/Executed)*100%

КРИТЕРИИ КАЧЕСТВА



Боб Мартин на конференции “Agile 2008” предложил прекрасную метрику качества кода: “What the f… / per minute”. Для проведения исследования нужен лишь секундомер, эксперт и калькулятор.

КЛАССИФИКАЦИЯ ТЕСТИРОВАНИЯ ПО

Статическое тестирование (static testing) - это процесс анализа самой разработки программного обеспечения, иными словами – это тестирование без запуска программы (проверка кода, требований, функциональной спецификации, архитектуры, дизайна и т.д.)

Динамическое тестирование (dynamic testing) - это тестовая деятельность, предусматривающая эксплуатацию (запуск) программного продукта.

- Оно делится на несколько подтипов: тестирование белого ящика, тестирование черного ящика, а иногда выделяют и тестирование серого ящика.
- Эта классификация уже относится к ***методам тестирования***, т.е. ***как именно*** тестируют программу.

МЕТОДЫ ТЕСТИРОВАНИЯ

Метод белого ящика (white-box testing, glass-box testing) – тестирование, при котором тестировщик ***имеет доступ к коду***. Его еще называют ***тестированием стеклянного ящика*** или ***тестированием прозрачного ящика***..

Тесты основаны на знании кода приложения и его внутренних механизмов.

Метод белого ящика часто используется на стадии, когда приложение ещё не собрано воедино, но необходимо проверить каждый из его компонентов, модулей, процедур и подпрограмм.

МЕТОДЫ ТЕСТИРОВАНИЯ

Метод чёрного ящика (black-box testing) заключается в том, что тестировщик имеет доступ к ПО только через те же интерфейсы, что и заказчик или пользователь.

Тестирование чёрного ящика ведётся с использованием спецификаций или иных документов, описывающих требования к системе на основе применения пользовательского интерфейса для ввода входных и получения выходных данных.

Цель данного метода – проверить работу всех функций приложения на соответствие функциональным требованиям.

Как вы думаете, каковы основные преимущества метода белого ящика? А чёрного ящика?

МЕТОДЫ ТЕСТИРОВАНИЯ

Метод серого ящика (gray box testing) – совокупность подходов из методов белого и чёрного ящика.

Этот метод, как правило, используется при тестировании веб-приложений, когда тестировщик знает принципы функционирования технологий, на которых построено приложение, но может не видеть кода самого приложения.

ПО ОБЪЕКТУ ТЕСТИРОВАНИЯ

- Функциональное тестирование
- Тестирование производительности
  - Нагрузочное тестирование
  - Стресс-тестирование
  - Тестирование стабильности
- Конфигурационное тестирование
- Юзабилити-тестирование
- Тестирование безопасности
- Тестирование локализации
- Тестирование совместимости
- Конфигурационное тестирование
- Инсталляционное тестирование
- Тестирование документации

ПО ОБЪЕКТУ ТЕСТИРОВАНИЯ

Функциональное тестирование (functional testing) – процесс проверки программного обеспечения, сконцентрированный на анализе соответствия ПО требованиям и спецификациям. Функциональное тестирование ещё называют поведенческим или тестированием на поведенческом уровне.

Цели функционального тестирования:

- **Обнаружить дефекты** в программном продукте.
- **Определить степень соответствия** программного продукта требованиям и ожиданиям заказчика.
- **Принять решение** о возможности передачи продукта заказчику.

ПО ОБЪЕКТУ ТЕСТИРОВАНИЯ

Тестирование производительности (performance testing) – проверяет способность программы выполнять заданное количество операций в заданный промежуток времени:

- **Нагрузочное тестирование** (load testing) – проверяет способность приложения работать при запланированной нагрузке.
- **Стресс-тестирование** (stress testing) – обычно используется для понимания пределов пропускной способности приложения. Этот тип тестирования проводится для определения надёжности системы во время экстремальных или диспропорциональных нагрузок и отвечает на вопросы о достаточной производительности системы в случае, если текущая нагрузка сильно превысит ожидаемый максимум.
- **Тестирование стабильности** (stability/endurance/soak testing) –проводится с целью убедиться в том, что приложение выдерживает ожидаемую нагрузку в течение длительного времени.
- **Конфигурационное тестирование** – ещё один из видов традиционного тестирования производительности. В этом случае вместо того, чтобы тестировать производительность системы с точки зрения подаваемой нагрузки, тестируется эффект влияния на производительность изменений в конфигурации. Конфигурационное тестирование может быть совмещено с нагрузочным, стресс или тестированием стабильности.

ПО ОБЪЕКТУ ТЕСТИРОВАНИЯ

Тестирование удобства использования (usability testing) – проверка того, насколько пользователю удобно и приятно работать с приложением.

Суровая правда жизни (C) bash.org.ru
«Поражаюсь старательности подрастающего поколения... Сегодня по дороге на работу заметил появление большого количества скворечников на деревьях. Дабы не долбить деревья гвоздями, скворечники примотаны скотчем. Многие прямо поперек кругленького отверстия...»

ПО ОБЪЕКТУ ТЕСТИРОВАНИЯ

Тестирование безопасности (security testing)

Тестирование безопасности представляет собой ряд работ: от разработки политики безопасности до тестирования безопасности на уровне приложения, операционной системы и сетевой безопасности.

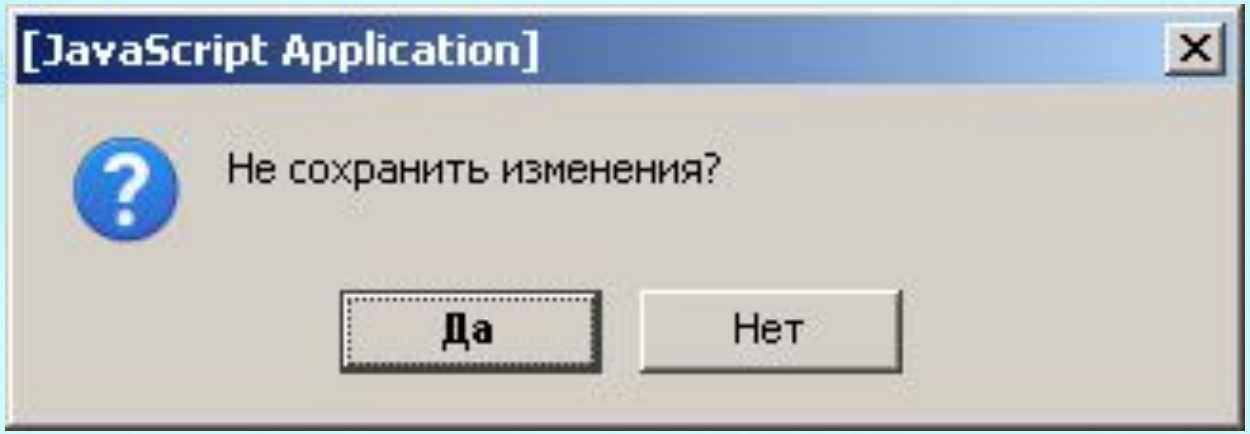
Тестирование безопасности может иметь различную степень покрытия:

- Первичное тестирование безопасности.
- Полное тестирование приложения.
- Полное тестирование приложения и сервера.

ПО ОБЪЕКТУ ТЕСТИРОВАНИЯ

Тестирование интернационализации (internationalisation testing) – проверка готовности продукта к переводу на различные языки.

Тестирование локализации (localisation testing) – проверка качества перевода продукта на конкретный язык.



ПО ОБЪЕКТУ ТЕСТИРОВАНИЯ

И ещё немного о тестировании локализации …
суровая правда жизни с bash.org.ru

«Попалась одна прога. Всё бы ничего, но русификация интерфейса не смогла оставить равнодушной. Итак – лучшее (орфография сохранена):

- монитор силы (power monitor);
- пульт команды (command console);
- штепсельная вилка и игра (PnP);
- предметы Хелпера Браузера (???);
- смеситель (mixer);
- волна внутри (wave in);
- волна вне (wave out);
- сердечник ODBC (ODBC core);
- водители ODBC (ODBC drivers);
- резьбы (threads);
- побегите история (???);
- освежите (refresh);
- о (about).»

ПО ОБЪЕКТУ ТЕСТИРОВАНИЯ

Тестирование совместимости (compatibility testing) – проверка того, как приложение взаимодействует с другими приложениями и операционной системой. В случае веб-ориентированных приложений особое внимание уделяется совместимости с различными браузерами.

Инсталляционное тестирование (installation testing) – проверка всего того, что связано с инсталляцией продукта в систему и удалением продукта из системы.

Тестирование документации (documentation testing) – вид тестирования, с которого начинается почти любой проект. Призвано обнаружить ошибки в документации. Эти ошибки опасны тем, что они, как маленький комок снега могут вызвать лавину проблем, вырастая на более поздних стадиях работы с проектом в очень сложно-устранимые и дорогостоящие последствия.

ПО СТЕПЕНИ АВТОМАТИЗАЦИИ:

- **Ручное тестирование** (manual testing) – тестирование без применения различных средств автоматизации.
- **Автоматизированное тестирование** (automated testing) – тестирование с применением различных средств автоматизации.

ПО СТЕПЕНИ ИЗОЛИРОВАННОСТИ КОМПОНЕНТОВ:

Компонентное (модульное) тестирование (component/unit testing) – тестирование отдельного модуля программного средства (под модулем может пониматься в т.ч. отдельный класс, метод и т.д.)

Интеграционное тестирование (integration testing) – проверка того, как отдельные компоненты, проверенные на предыдущем уровне, взаимодействуют друг с другом.

Системное тестирование (system/end-to-end testing) – полная проверка приложения: проверяются как функциональные, так и нефункциональные требования.

ПО ВРЕМЕНИ ПРОВЕДЕНИЯ ТЕСТИРОВАНИЯ:

Альфа-тестирование (alpha testing) – имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком.

Тестирование при приёмке (smoke testing)

Тестирование новой функциональности (newfeature testing) – проверка того, что заявленный в данном билде новый функционал работает должным образом.

Регрессионное тестирование (regression testing) - проверка того, что внесённые в приложение изменения **не привели** к потере работоспособности того, что ранее работало, и/или **привели** к работоспособности того, что ранее не работало.

Тестирование при сдаче (acceptance testing)

Бета-тестирование (beta testing) – интенсивное использование почти готовой версии продукта (как правило, программного или аппаратного обеспечения) с целью выявления максимального числа ошибок в его работе для их последующего устранения перед окончательным выходом (Релизом) продукта на рынок, к массовому потребителю. Кроме того, открытие бета-тестирования может использоваться как стратегия: продвижение продукта на рынок, т.к. реклама, а также получение предварительных отзывов.

В отличие от альфа-тестирования, проводимого силами штатных разработчиков или тестировщиков, бета-тестирование предполагает привлечение добровольцев из числа обычных будущих пользователей, которым доступна упомянутая предварительная версия продукта (так называемая **бета-версия**).

ПО ПРИЗНАКУ ПОЗИТИВНОСТИ СЦЕНАРИЕВ:

- **Позитивное тестирование** (positive testing) – проверка того, как приложение работает в заведомо “тепличных условиях” (корректные данные, условия работы и т.п.)
- **Негативное тестирование** (negative testing) – проверка того, как приложение реагирует на различные “неприятности” (пропала сеть, повреждён файл, введены некорректные данные и т.п.)

Ожидаемый результат *(expected result) – такое поведение программного средства, которое мы ожидаем в ответ на наши действия.*

ПО СТЕПЕНИ ПОДГОТОВЛЕННОСТИ К ТЕСТИРОВАНИЮ:

- **Тестирование по документации** (formal testing) – тестирование по заданному плану, по разработанным чек-листам и тест-кейсам.
- **Тестирование ad hoc или интуитивное тестирование** (ad hoc testing) – в данном случае тестировщик пытается поставить себя на место пользователя или просто решает «Как можно сломать программу?». При таком тестировании план тестировщик держит в голове или делает для себя пометки.

ВЫВОДЫ

1. Тестирование всегда разное. Задача тестировщика – узнать, какие действия будут наиболее полезными в его условиях работы.

2. Наша основная задача – проверять корректность работы программ и сообщать руководителю проекта "что работает, а что – нет".

И еще…

1. Целью тестирования является обнаружение ошибок в тестируемом объекте, а не доказательство их отсутствия.

2. Тестировщики не отвечают за качество. Они помогают тем, кто за него отвечает.

3. Тестирование даёт тем большую экономическую отдачу, чем на более ранних стадиях работы над проектом оно выявило дефект.

4. Тестирование имеет смысл прекращать тогда, когда устранены все критические и 85% и более некритических дефектов программы, т.к. дальнейшее тестирование, как правило, является неоправданной статьёй расходов.

Психологические аспекты тестирования

Хороший тестировщик должен обладать следующими психологическими качествами:

- Повышенной ответственностью.
- Хорошими коммуникативными навыками.
- Способностью ясно, быстро, чётко выражать свои мысли.
- Исполнительностью.
- Терпением, усидчивостью, внимательностью к деталям, наблюдательностью.
- Гибким мышлением, хорошей способностью к обучению.
- Хорошим абстрактным и аналитическим мышлением.
- Способностью ставить нестандартные эксперименты.
- Высокими техническими навыками.
- Склонностью к исследовательской деятельности.

Технические навыки тестировщика

Тестировщик (в идеале) должен знать следующие технологии:

Программирование: C/C++/C#, Java, Object Pascal, Visual Basic, JavaScript, VBScript, HTML, .NET.

Администрирование СУБД: Oracle, MS SQL, IBM DB2, Sybase, Informix.

Системное администрирование: Windows, Sun Solaris, HP-UX, IBM AIX, Linux, Free BSD.

Сетевое администрирование: NetWare, Cisco IOS, TCP/IP, IPX/SPX, NetBIOS.

Автоматизированное тестирование: Segue SilkTest and SilkPerformer, Mercury Interactive WinRunner, Quick Test Pro and LoadRunner, JUnit, HTTP Unit.