ОБЕСПЕЧЕНИЕ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. ТЕСТИРОВАНИЕ

КАЧЕСТВО ПО

- **Качество** - это цель инженерной деятельности.
  - Построение качественного ПО – это цель программной инженерии.
  - Все характеристики качества ПО делятся на внешние и внутренние.

    Некоторые из этих характеристик перекрываются, однако каждая имеет свои отличительные черты.
  - **Внешние характеристики** – это свойства, важные (осознаваемые) для пользователей программы.

    Единственная категория свойств ПО, которая волнует пользователей.

    Например:
    - пользователей беспокоит легкость работы с ПО, а не легкость его изменения.
    - пользователей заботит корректность ПО, а не удобочитаемость или структурированность кода.
  - **Внутренние характеристики** – это свойства программы, которые важны для разработчиков.

    Например: гибкость, сопровождаемость.

ОСНОВНЫЕ ХАРАКТЕРИСТИКИ КАЧЕСТВА ПО (СТАНДАРТ ISO 9126)

- Функциональность (Functionality) – способность ПО выполнять требуемые заказчиком функции (решать задачи).
- Надежность (Reliability) – способность предоставлять безотказное обслуживание.
- Удобство и простота использования (Usability) – способность быть понятным, легко осваиваемым и применяемым.

ОСНОВНЫЕ ХАРАКТЕРИСТИКИ КАЧЕСТВА ПО (СТАНДАРТ ISO 9126)

- Производительность (Эффективность, Efficiency) – способность предоставлять полезный результат, соответствующую количеству используемых ресурсов
- Удобство сопровождения (Maintainability) – возможность модифицировать ПО с целью выполнения исправлений, улучшений и адаптаций.
- Переносимость (Portability) – способность приспосабливать ПО к разным требуемым условиям без применения действий или средств, отличающихся от тех, которые были предоставлены для этих целей в данном продукте.

ОБЩАЯ ТЕОРИЯ ТЕСТИРОВАНИЯ

- Тестирование – проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом.

- Цель тестирования — проверка соответствия ПО предъявляемым требованиям, обеспечение уверенности в качестве ПО, поиск очевидных ошибок в программном обеспечении, которые должны быть выявлены до того, как их обнаружат пользователи программы.

ТЕСТИРОВАНИЕ

Тестирование – это наблюдение за функционированием ПО в специфических условиях с целью определения степени соответствия ПО требованиям к нему

- ISO, 1990 г.
- ISO/IEC 29119 Software Testing – новый стандарт, 2011 год.

Тестирование – техническое исследование, проводимое с целью предоставить заинтересованным лицам информацию о качестве тестируемого продукта (Сэм Канер, 1999).

Тестирование – проверка соответствия реального поведения системы ожидаемому, осуществляемая на конечном наборе тестов, выбранным определенным образом (SWEBOK, 2004).

Основная цель тестирования – повысить качество продукта.

ТЕСТИРОВАНИЕ ПО

1. Функциональность
2. Надёжность
3. Удобство использования (практичность)
4. Производительность (эффективность)
5. Удобство сопровождения
6. Переносимость (мобильность)

Тестирование и отладка

Тестирование – поиск ошибок в ПО.

Отладка – поиск местоположения ошибок в программном коде и их исправление.

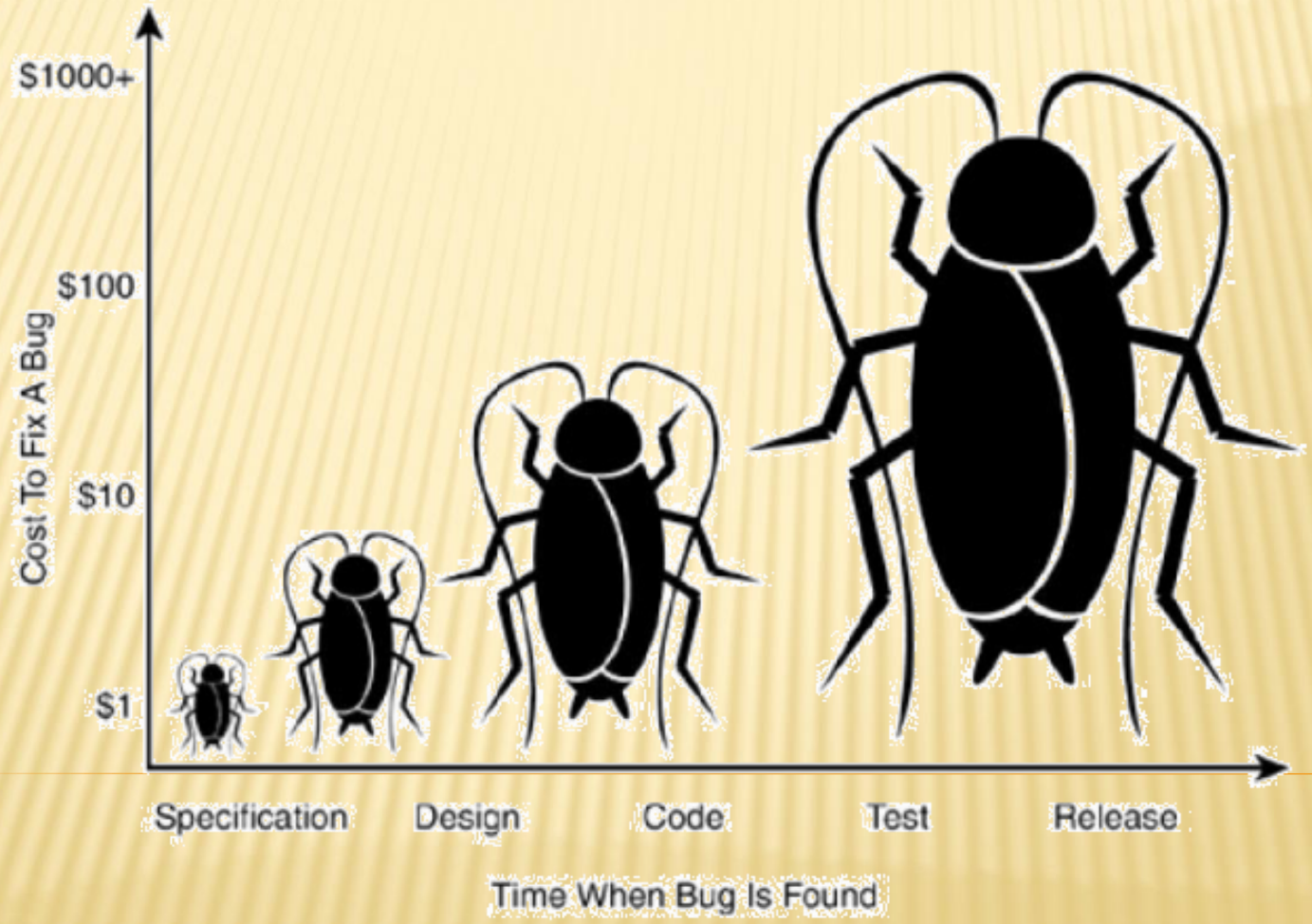
ПРИНЦИПЫ ТЕСТИРОВАНИЯ

1. Тестирование снижает вероятность наличия дефектов, но не гарантирует их отсутствие.
2. Полное тестирование с использованием всех входных комбинаций данных, результатов и предусловий физически невыполнимо (исключение — тривиальные случаи).
3. Раннее тестирование (Early testing) – следует начинать тестирование на ранних стадиях жизненного цикла разработки ПО, чтобы найти дефекты как можно раньше.
4. Скопление дефектов (Defects clustering) – большая часть дефектов находится в ограниченном количестве модулей, обычно в модулях, реализующих бизнес логику приложения.
5. Парадокс пестицида (Pesticide paradox) – если повторять те же тестовые сценарии снова и снова, в какой-то момент этот набор тестов перестанет выявлять новые дефекты.
6. Тестирование зависит от контекста (Testing is context depending) – тестирование проводится по-разному в зависимости от контекста. Например, программное обеспечение, в котором критически важна безопасность, тестируется иначе, чем новостной портал.
7. Заблуждение об отсутствии ошибок (Absence-of-errors fallacy) – отсутствие найденных дефектов при тестировании не всегда означает готовность продукта к релизу. Система должна быть удобна пользователю в использовании и удовлетворять его ожиданиям и потребностям.

ГРАДАЦИЯ СЕРЬЕЗНОСТИ БАГОВ

- Блокирующий (Blocker) – блокирующая ошибка, приводящая приложение в нерабочее состояние, в результате которого дальнейшая работа с тестируемой системой или ее ключевыми функциями становится невозможна для пользователя. Может быть только на prod.
- Критический (Critical) – критическая ошибка, неправильно работающая ключевая бизнес-логика, дыра в системе безопасности, проблема, приведшая к временному падению сервера или приводящая в нерабочее состояние некоторую часть системы, то есть не работает важная часть одной какой-либо функции, либо не работает значительная часть, но имеется workaround (обходной путь/другие входные точки), позволяющий продолжить работу.
- Значительный (Major) – не работает важная часть одной какой-либо функции/бизнес-логики, но при выполнении специфических условий, либо есть workaround, позволяющий продолжить работу, либо не работает не очень значительная часть какой-либо функциональности. Также относится к дефектам с высокими visibility – обычно не сильно влияющие на функциональность дефекты дизайна, которые, однако, сразу бросаются в глаза.
- Незначительный (Minor) – часто ошибки GUI, которые не влияют на функциональность, но портят usability или внешний вид. Также незначительные функциональные дефекты, либо которые воспроизводятся на определенном устройстве.
- Тривиальный (Trivial) – почти всегда дефекты на GUI — опечатки в тексте, несоответствие шрифта и оттенка и т.п., либо плохо воспроизводимая ошибка, не касающаяся бизнес-логики, проблема сторонних библиотек или сервисов, проблема, не оказывающая никакого влияния на общее качество продукта.

КРИВАЯ БОЕМА (BOEHM CURVE)



МЕТОДЫ ТЕСТИРОВАНИЯ

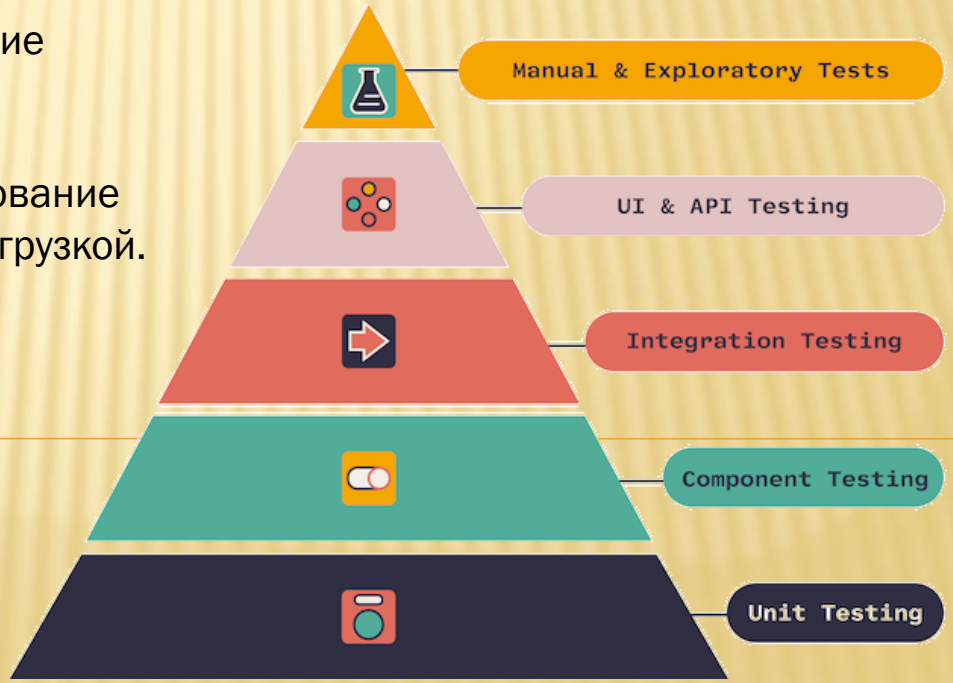
- Метод белого ящика – тестирование, которое учитывает внутренние механизмы программы, обычно включает тестирование ветвей, маршрутов, операторов (т.е. практически всегда это unit-тесты, реже интеграционное тестирование). При тестировании подбирают входные параметры и определяют выходные параметры. Этот метод тестирования не может выявить невыполненные части спецификации.

- Метод черного ящика – также известное как тестирование, основанное на спецификации или тестирование поведения — техника тестирования, основанная на работе исключительно с внешними интерфейсами тестируемой системы (обычно это функциональное тестирование).

- Метод серого ящика – метод тестирования ПО, который предполагает комбинацию White Box и Black Box подходов. То есть, внутреннее устройство программы нам известно лишь частично.

ВИДЫ (УРОВНИ) ТЕСТИРОВАНИЯ

- Unit-тесты – проверка отдельного класса программы.
- Сервисные тесты (Component Tests) – проверка, что изолированная система работает корректно (т.е. все внешние вызовы закрыты заглушками).
- Интеграционные тесты (Integration Tests) – тестирование контура с реальными запросами и ответами между системами.
- UI-тесты – проверка работы пользовательского интерфейса.
- Security tests – тестирование, направленное на выявление дырок в безопасности системы (как на уровне самого приложения, для уверенности в том, что эти изменения не внесли ошибки в областях, которые не подверглись изменениям. приложения, так и на уровне инфраструктуры).
- Регрессионное тестирование – тестирование уже проверенной ранее функциональности после внесения изменений в код
- Нагрузочные тесты (стресс-тесты) – тестирование поведения системы под возрастающей нагрузкой.



МОДУЛЬНОЕ ТЕСТИРОВАНИЕ

- Модульному тестированию подвергаются небольшие модули (процедуры, классы и т.п.).
- При тестировании небольшого модуля (100-1000 строк) можно проверить,
  - многие логические ветви в реализации,
  - разные пути в графе зависимости данных,
  - граничные значения параметров.
- Для этого составляются критерии тестового покрытия
  - покрыты все операторы, все логические ветви, все граничные точки и т.п.
- Модульное тестирование обычно выполняется для каждого независимого программного модуля и является наиболее распространенным видом тестирования
  - особенно для систем малых и средних размеров.

ИНТЕГРАЦИОННОЕ ТЕСТИРОВАНИЕ

- Проверка корректности всех модулей по отдельности не гарантирует корректности функционирования системы модулей.
  - Необходимо проверять (тестировать) их совместную работу.
  - Для больших программных систем нереально использовать метод «большого скачка»
    - разработать все приложение
    - протестировать каждый модуль в отдельности
    - потом объединить их в систему и тестируется система целиком.
- Требуется очень много времени на поиск ошибок, а качество тестирования останется невысоким.
- Интеграционное тестирование альтернатива «большому скачку» – система строится поэтапно, группы модулей добавляются постепенно.

СИСТЕМНОЕ ТЕСТИРОВАНИЕ

- Полностью реализованный ПО подвергается системному тестированию.
- Тестировщика интересует не корректность реализации отдельных процедур и методов, а вся программа в целом,
  - как ее видит конечный пользователь.
- Основой для тестов служат общие требования к программе, включая не только корректность реализации функций, но и:
  - производительность,
  - время отклика,
  - устойчивость к сбоям, атакам, ошибкам пользователя и т.д.
- Для системного и компонентного тестирования используются специфические виды критериев тестового покрытия
- Например:
  - покрыты ли все типовые сценарии работы,
  - покрыты ли все сценарии с нештатными ситуациями, попарные композиции сценариев и прочие

ПРИЁМОЧНЫЕ ТЕСТЫ

- Модульные-тесты — это белые ящики, которые проверяют отдельные механизмы работы системы
  - тест, который знает о внутренней структуре тестируемого модуля и зависит от нее.
- Приемочные тесты — это черные ящики, которые проверяет, удовлетворены ли требования заказчика.
  - Тест, который не знает о внутренней структуре тестируемого модуля и не зависит от нее.
- Приемочные тесты пишутся специалистами, которые не о внутреннем устройстве системы.

  Например:
  - заказчиками;
  - бизнес-аналитиками;
  - специалистами по контролю качества.
- Выполнение приемочных тестов часто автоматизируется.
- Обычно они составляются на специальном языке спецификаций, понятном людям, не обладающим техническими навыками.
- Приемочные тесты — это вершина документирования функции.
- Заказчик пишут приемочные тесты, проверяющие, что некоторая функция работает правильно.
- Программисты могут, прочитав их текст, полностью разобраться в назначении функции.

КЛАССИФИКАЦИЯ МЕТОДОВ ТЕСТИРОВАНИЯ ПО ВРЕМЕНИ ПРОВЕДЕНИЯ ТЕСТИРОВАНИЯ

- Альфа тестирование (alpha testing)
- Тестирование при приёмке (smoke testing)
- Тестирование новых функциональностей (new feature testing)
- Регрессионное тестирование (regression testing)
- Тестирование при сдаче (acceptance testing)
- Бета тестирование (beta testing)

АЛЬФА-ТЕСТИРОВАНИЕ

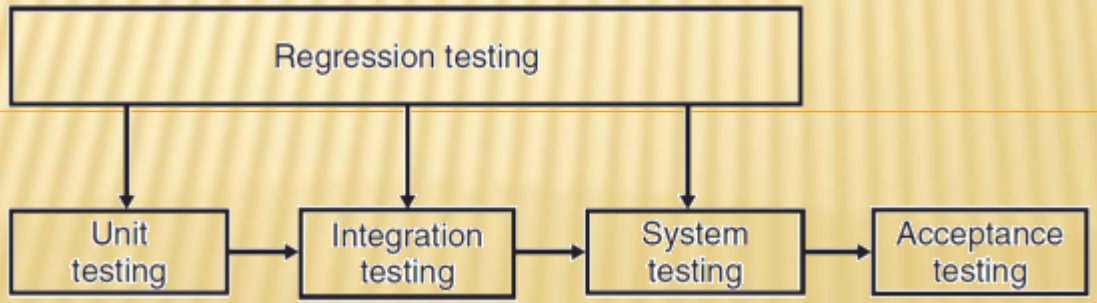
- Альфа-тестирование имитация реальной работы с системой
  - штатными разработчиками,
  - либо реальная работа с системой лотенциальными пользователями / заказчиком.
  - чаще всего альфа-тестирование проводится на ранней стадии разработки лродукта,
  - в некоторых случаях может применяться для законченного лродукта в качестве внутреннего приёмочного тестирования.
- Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки.
- Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться ПО.

БЕТА-ТЕСТИРОВАНИЕ

- Бета-тестирование — распространение версии с ограничениями (по функциональности или времени работы) для некоторой группы лиц, с тем чтобы убедиться, что продукт содержит достаточно мало ошибок.
  - иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.
- Часто для свободного/открытого ПО
  - стадия Альфа-тестирования характеризует функциональное наполнение кода,
  - Бета тестирования — стадию исправления ошибок.
- Как правило на каждом этапе разработки промежуточные результаты работы доступны конечным пользователям

РЕГРЕССИОННОЕ ТЕСТИРОВАНИЕ

- Регрессионное тестйрование — общее название для всех видов тестирования ПО, направленных на обнаружение ошибок в уже протестированных участках исходного кода.
  - Ошибки — когда после внесения изменений в программу перестает работать то, что должно было продолжать работать, — называют регрессионными ошибками (regression bugs).
- Методы регрессионного тестирования включают
  - повторные прогоны предыдущих тестов,
  - проверку, не попали ли регрессионные ошибки в очередную версию в результате слияния кода.



КЛАССИФИКАЦИЯ МЕТОДОВ ТЕСТИРОВАНИЯ ПО ОБЪЕКТУ ТЕСТИРОВАНИЯ

- Функциональное тестирование (functional testing)
- Нагрузочное тестирование (load testing)
- Тестирование производительности (perfomance/stress testing)
- Тестирование стабильности (stability/load testing)
- Тестирование удобства использования (usability testing)
- Тестирование интерфейса пользователя (UI testing)
- Тестирование безопасности (security testing)
- Тестирование локализации (localization testing)
- Тестирование совместимости (compatibility testing)

КЛАССИФИКАЦИЯ ВИДОВ ТЕСТИРОВАНИЯ ПО ЗНАНИЮ СИСТЕМЫ

- Тестирование «белого ящика» - разработчик тестов имеет доступ к исходному коду тестируемого ПО.
  - может писать код, который связан с библиотеками тестируемого ПО.
  - типично для юнит-тестирования, когда тестируются только отдельные части системы.
  - обеспечивает то, что компоненты конструкции — работоспособны и устойчивы, до определённой степени.
- Тестирование «черного ящика» - тестировщик имеет доступ к ПО только через те же интерфейсы,
  - через пользовательский интерфейс
  - через прикладной программный интерфейс, предоставленный тестируемым модулем.

КЛАССИФИКАЦИЯ ВИДОВ ТЕСТИРОВАНИЯ ПО СТЕПЕНИ АВТОМАТИЗИРОВАННОСТИ

- Ручное тестирование (manual testing)
- Полуавтоматизированное тестирование (semiautomated testing)
- Автоматизированное тестирование (automated testing)

Стандартные типы тестовых примеров

- Допустимые данные
- Граничные данные
- Отсутствие данных
- Повторный ввод данных
- Неверные данные
- Реинициализация системы
- Устойчивость системы
- Нештатные состояния среды выполнения