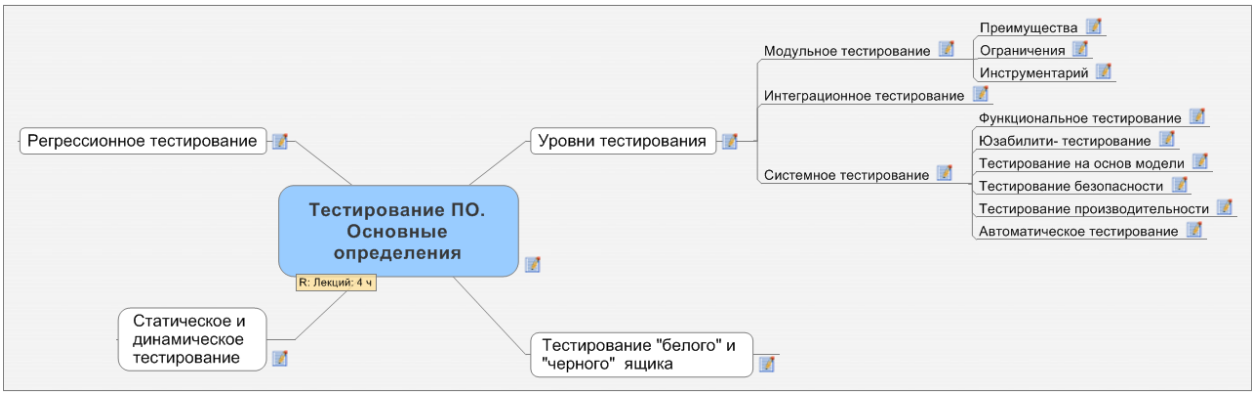
Лекция 11. Тестирование ПО. Основные определения(продолжение)



Resource: Лекций: 4 ч

1.1 Системное тестирование

Системное тестирование программного обеспечения — это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям.

Системное тестирование относится к методам тестирования чёрного ящика, и, тем самым, не требует знаний о внутреннем устройстве системы.

1.1.1 Функциональное тестирование

Функциональное тестирование - это тестирование ПО в целях проверки реализуемости функциональных требований, т.е. способности ПО в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

Функциональные требования включают:

- Функциональная пригодность (англ. suitability).
- Точность (англ. accuracy).
- Способность к взаимодействию (англ. interoperability).
- Соответствие стандартам и правилам (англ. compliance).
- Защищенность (англ. security).

1.1.2 Юзабилити- тестирование

Юзабилити-тестирование — эксперимент, выполняемый с целью определения, насколько хорошо люди могут использовать некоторый искусственный объект (такой как веб-страница, пользовательский интерфейс или устройство) для его предполагаемого применения, то есть юзабилити- тестирование измеряет юзабилити объекта. Юзабилити- тестирование сосредоточено на определенном объекте или небольшом наборе объектов, в то время как исследования взаимодействия человек-компьютер в целом — формулируют универсальные принципы.

Юзабилити-тестирование — метод оценки удобства использования продукта, основанный на привлечении пользователей в качестве тестирующих.

Процесс

Пользователю предлагают решить основные задачи, для решения которых разрабатывался продукт, и просят произносить вслух то, о чем он думает во время выполнения тестовых заданий.

Процесс тестирования фиксируется на аудио- и видеоустройства с целью последующего анализа.

Если юзабилити-тестирование выявляет трудности, такие как сложности в понимании инструкций, выполнении действий или интерпретации ответов системы, то разработчики должны улучшить объект и повторить тестирование.

Наблюдение того, как люди взаимодействуют с продуктом, позволяет найти способы улучшения дизайна. Но бесстрастно наблюдать за этим довольно сложно. Человеку присуще помогать тем, кто нуждается в помощи, а не наблюдать за их неудачами. Эффективный модератор должен быть способен держать респондента сфокусированным на задачах, не помогая ему при этом решать эти задачи.

Основную техническую трудность процедуры юзабилити- тестирования представляет большой объем данных, которые нужно зафиксировать в процессе тестирования для целей последующего анализа:

- Речь модератора и респондента;
- Выражение лица респондента (снимается на видеокамеру);
- Изображение экрана компьютера, с которым работает респондент;
- Различные события, происходящие на компьютере, связанные с действиями пользователя:
- Перемещение курсора и нажатия на клавиши мыши;
- Использование клавиатуры;
- Переходы между экранами (браузера или другой программы).

Все эти потоки данных должны быть синхронизированы, чтобы при анализе их можно было бы соотносить между собой.

Наблюдатели, которые участвуют в тестировании наряду с модератором, ведут заметки по мере обнаружения проблем, которые тоже нужно синхронизировать с другими данными так, чтобы потом можно было легко найти фрагмент записи, прокомментированный в заметке наблюдателя. Отсутствие специальных инструментов делает последующий анализ настолько трудоемким, что время, необходимое для него, превышает две трети общего времени, затраченного на тестирования — от его планирования до сдачи отчета.

1.1.3 Тестирование на основе модели

Тестирование на основе модели (англ. Model-based testing) - это тестирование программного обеспечения, в котором тестовые случаи (test cases) частично или целиком получаются из модели описывающей некоторые аспекты (чаще функциональные) тестируемой системы (system under test).

1.1.4 Тестирование безопасности

Тестирование безопасности — оценка уязвимости программного обеспечения к различным атакам.

Компьютерные системы очень часто являются мишенью незаконного проникновения. Под проникновением понимается широкий диапазон действий: попытки хакеров проникнуть в систему из спортивного интереса, месть рассерженных служащих, взлом мошенниками для незаконной наживы. Тестирование безопасности проверяет фактическую реакцию защитных механизмов, встроенных в систему, на проникновение. В ходе тестирования безопасности испытатель играет роль взломщика. Ему разрешено все:

- попытки узнать пароль с помощью внешних средств;
- атака системы с помощью специальных утилит, анализирующих защиты;
- подавление, ошеломление системы (в надежде, что она откажется обслуживать других клиентов);
- целенаправленное введение ошибок в надежде проникнуть в систему в ходе восстановления;
- просмотр несекретных данных в надежде найти ключ для входа в систему.

Конечно, при неограниченном времени и ресурсах хорошее тестирование безопасности взломает любую систему. Задача проектировщика системы — сделать цену проникновения более высокой, чем цена получаемой в результате информации.

1.1.5 Тестирование производительности

Тестирование производительности — оценка функции производительности.

- программного обеспечения
- оборудования (например, персонального компьютера)

Иногда тестирование производительности сочетают со стрессовым тестированием. При этом нередко требуется специальный аппаратный и программный инструментарий. Например, часто требуется точное измерение используемого ресурса (процессорного цикла и т. д.). Внешний инструментарий регулярно отслеживает интервалы выполнения, регистрирует события (например, прерывания) и машинные состояния. С помощью инструментария испытатель может обнаружить состояния, которые приводят к деградации и возможным отказам системы

1.1.6 Автоматическое тестирование

Автоматическое тестирование использует специальное программное обеспечение для проверки выполнения проводимых тестов, что помогает сократить время тестирования и упростить его процесс.

В наши дни создаётся большое количество инструментов с графическим интерфейсом (GUI), использование которых помогает облегчить работу программистов и повышает их производительность. Эти процессы увеличили требования к тестировщикам. Сейчас они должны обрабатывать большое количество информации за короткий срок. Потому использование автоматических тестов является необходимым условием для экономии средств и времени тестировщиков.

Современные средства разработки создают довольно сложные приложения, и их ручное тестирование является очень трудоёмким процессом. Недостаток ручного тестирования также в том, что результаты выполнения тестов не сохраняются и их трудно повторить заново. Автоматические тесты позволяют упростить процесс ручного тестирования, сделать его наиболее удобным и точным.

Для автоматизации тестирования существует большое количество приложений. Наиболее популярные из них по итогам 2008 года:

- HP LoadRunner, HP QuickTest Professional , HP Quality Center
- Segue SilkPerformer
- IBM Rational FunctionalTester, IBM Rational PerformaneTester, IBM Rational TestStudio
- AutomatedQA TestComplete

Использование этих инструментов помогает тестировщикам автоматизировать следующие задачи:

- установка продукта
- создание тестовых данных
- GUI взаимодействие
- определение проблемы

Однако автоматические тесты не могут полностью заменить ручное тестирование. Автоматизация всех испытаний — очень дорогой процесс, и потому автоматическое тестирование является лишь дополнением ручного тестирования. Наилучший вариант использования автоматических тестов — регрессионное тестирование.

2 Тестирование "белого" и "черного" ящика

При ***тестировании белого ящика (white-box testing)***, разработчик теста имеет доступ к исходному коду программ и может писать код, который связан с библиотеками тестируемого ПО. Это типично для юнит-тестирования (*unit testing*), при котором тестируются только отдельные части системы.

Оно обеспечивает то, что компоненты конструкции — работоспособны и устойчивы, до определенной степени. При тестировании белого ящика используются метрики ***покрытия кода.***

Покрытие кода - это запуск тестируемоего ПО со специальными настройками или библиотеками и/или в особом окружении.

В результате для каждой используемой (выполняемой) функции программы определяется местонахождение этой функции в исходном коде. Этот процесс позволяет разработчикам и специалистам по обеспечению качества определить части системы, которые, при нормальной работе, используются очень редко или никогда не используются (такие как код обработки ошибок и т.п.). Это позволяет сориентировать тестировщиков на тестирование наиболее важных режимов.

Тестировщики могут использовать результаты теста покрытия кода для разработки тестов или тестовых данных, которые расширят покрытие кода на важные функции.

Как правило, инструменты и библиотеки, используемые для получения покрытия кода, требуют значительных затрат производительности и/или памяти, недопустимых при нормальном функционировании ПО. Поэтому они могут использоваться только в лабораторных условиях.

При ***тестировании чёрного ящика (black-box testing)***, тестировщик имеет доступ к ПО только через те же интерфейсы, что и заказчик или пользователь, либо через внешние интерфейсы, позволяющие другому компьютеру либо другому процессу подключиться к системе для тестирования.

Как правило, тестирование чёрного ящика ведётся с использованием спецификаций или иных документов, описывающих требования к системе.

В данном виде тестирования критерий покрытия складывается из покрытия структуры входных данных, покрытия требований и покрытия модели (в тестировании на основе моделей).

3 Статическое и динамическое тестирование

Тестирование белого ящика и тестирование чёрного ящика — предполагают, что код исполняется, и разница состоит лишь в той информации, которой владеет тестировщик. В обоих случаях это ***динамическое тестирование.***

При ***статическом тестировании*** программный код не выполняется — анализ программы происходит на основе исходного кода, который вычитывается вручную, либо анализируется специальными инструментами. В некоторых случаях, анализируется не исходный, а промежуточный код.

4 Регрессионное тестирование

После внесения изменений в очередную версию программы, регрессионные тесты подтверждают, что сделанные изменения не повлияли на работоспособность остальной функциональности приложения.

Регрессионное тестирование может выполняться как вручную, так и программой, автоматизирующей этот процесс.