**Лекция 1**

*1. Что такое тестирование?*

Это проверка соответствия желаемых(ожидаемых) результатов поведения программы с фактическими результатами. Проверка проводится путём наблюдения за программой с помощью специальных искуственно-созданных сценариев.

*2.Чем отличается контроль качества и обеспечение качества.*

Контроль качества(Quality control) – это совокупность действий, проводимых над продуктом в процессе разработки, для получения информации о его актуальном состоянии.

Главные задачи:

1.Проверка готовности продукта к релизу

2.Проверка соответствования требований и качества на разных этапах жизненного цикла.

Обеспечение качества – это совокупность мероприятий, которые охватывают весь процесс создания программы, все этапы технологической разработки, для обеспечения нужного уровня качества продукта.

*3.Когда лучше исправить дефект?*

До релиза, это значительно уменьшает затраты на продукт

*4. Перечислите основные обязанности тестировщика.*

* Изучение продукта
* Разработка сценариев и планов тестирования
* Проведение тестов
* Написание отчётности на основании найденных дефектов
* Проверяет доработки разработчиков, на предмет исправления ранее выявленных дефектов
* Автоматизирует тесты и процессы в ходе работы
* Превращает недоработки
* Предоставляет информацию о качестве продукта, об объёме выполненных работ, о сроках…

*5.Перечислите основные процедуры обеспечения качества*

Процедуры делятся на внутренние, формирующие природу качества. И внешние, которые контролируют эту природу.

Внутренние делятся на:

* Тестирование документации
* Unit-тестирование
* Code-review
* Демонстрация продукта

*6.Что такое юнит-тестирование -*

Модульное тестирование, которое позволяет провести проверки на уровне кода, проверяя отдельные функции и методы продукта. Данное тестирование как правило выполняется самим разработчиком ПО на этапе разработки продукта. Сначала пишется тест, а затем ПО, которое удовлетворяет данным теста.

*7.Для чего делается ревью кода.*

Это анализ кода с целью выявить ошибки, недочеты, расхождения самого кода от поставленной задачи.

* Позволяет определить степень соответствия кода нормам и обеспечить его прозрачную поддерживаемость в будущем
* Улучшает качество кода
* Позволяет найти все ошибки в реализации
* Помогает привести код к единому стилю написания
* Происходит обмен знаниями между разработчиками

*8.Для чего проводится демонстрация продукта*

Демонстрация продукта позволяет определить реалии. Степень соотвествие желаемого и фактических результатов.

*9.Чем полезна обратная связь*

Улучшение качества продукта из первых рук

**Лекция 2.**

1. Перечислите различия между тестированием методом черного ящика и белого ящика.

При тестировании белого ящика (также говорят — прозрачного ящика), разработчик теста имеет доступ к исходному коду программ и может писать код, который связан с библиотеками тестируемого программного обеспечения.

При тестировании чёрного ящика, тестировщик имеет доступ к программе 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.

2. Перечислите основные виды тестирования по объекту.

**Функциональное тестирование (functional testing)**

Основной вид тестирования. В целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям.

Источники информации о новой функциональности (feature, фичи): Документация, Коммуникация, Исследование продукта

\* Тестирование безопасности (security testing)

Это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

\* Тестирование взаимодействия (Interoperability Testing)

Проверяет способность приложения взаимодействовать с одним и более компонентами или системами, и включает в себя тестирование совместимости (compatibility testing) и интеграционное тестирование (integration testing).

**Нефункциональное тестирование**

\* Тестирование совместимости (compatibility testing)

\* Тестирование локализации (localization testing)

\* Тестирование удобства использования (usability testing)

\* Тестирование интерфейса пользователя (UI testing)

**Объемное тестирование (volume testing)**

Тестирование стабильности (Stability / Reliability Testing)

3*. Перечислите виды тестирования производительности.*

* Нагрузочное тестирование (Load testing)

Исследует работу системы под регламентируемой нагрузкой. Проводится, чтобы убедиться, что приложение обладает запланированной пропускной способностью и работает стабильно.

* Стрессовое тестирование (Stress testing)

Используется для определения пределов пропускной способности приложения. Система подвергается экстремальным нагрузкам, и исследуется её поведение в ситуациях, когда регламентируемые пределы нагрузок в несколько раз выше

* Объемное тестирование (Volume testing)

Исследует работу системы при увеличении объема обрабатываемой информации. Можно выяснить, что будет, если база данных приложения вырастет в несколько раз

* Тестирование стабильности (Stability testing)

Проверяет длительную работу системы при среднем уровне нагрузки. Не очень важна скорость выполнения отдельных операций. Главные объекты исследования: утечки памяти, перезагрузки серверов и другие аспекты, которые могут повлиять на стабильность работы системы.

4. Чем компонентное тестирование отличается от интеграционного?

**Компонентное (модульное) тестирование** проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по отдельности (модули программ, объекты, классы, функции и т.д.).

**Интеграционное тестирование** – это проверка целого программного продукта на предмет внутреннего взаимодействия его частей (модулей), а также на сочетаемость этой системы с другими сервисами.

5. Что такое регрессионное тестирование?

Собирательное название для всех видов тестирования программного обеспечения, направленных на обнаружение ошибок в уже протестированных участках исходного кода. Такие ошибки — когда после внесения изменений в программу перестаёт работать то, что должно было продолжать работать, — называют регрессионными ошибками (англ. regression bugs).

6. Что такое BVT?

BVT - это набор базовых тестовых сценариев, нацеленных на выявление явных ошибок. Минимальный набор тестов на явные ошибки. Дымовой тест обычно выполняется программистом.

7. Для чего используется дымовое тестирование?

Чтобы посмотреть, что ничего не сломалось из того, что раньше работало

8. В чем отличие альфа-тестирования от бета-тестирования?

Альфа-тестирование — внутреннее тестирование. Имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа-тестирование проводится на ранней стадии разработки продукта, 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 — в некоторых случаях выполняется 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.

9. Что такое негативные тесты?

Негативным называют тестирование, в рамках которого применяются сценарии, которые соответствуют внештатному поведению тестируемой системы. Это могут быть, например, исключительные ситуации или неверные данные.

Прежде всего негативное тестирование направлено на проверку устойчивости системы к различным воздействиям, валидации неверных данных, обработки исключительных ситуаций. Сценарии позитивного тестирования, в свою очередь, направлены на проверку работы системы с теми типами данных, для которых она разрабатывалась.

Создание позитивных сценариев (тест-кейсов), как правило, предшествует созданию негативных тест-кейсов. Сначала мы проверяем работу системы, когда наш условный пользователь работает с системой “правильно”, а потом приступаем к проверке отклика системы на пользователя, который допускает различные ошибки (ввод неверных данных, например). И наша система должна быть готова ответить на неверный запрос. Это и есть цель негативного тестирования.

10. В чём различие между Ad Hoc и Exploratory тестированиями?

Специализированное тестирование (Ad hoc testing)

Возможно, наиболее широко практикуемая техника. Тесты основываются на опыте, интуиции и знаниях инженера, рассматривающего проблему с точки зрения имевшихся ранее аналогий.

Исследовательское тестирование (Exploratory testing)

Такое тестирование определяется как одновременное обучение, проектирование теста и его исполнение. Данный вид тестирования заранее не определяется в плане тестирования и такие тесты создаются, выполняются и модифицируются динамически, по мере необходимости. Эффективность исследовательских тестов напрямую зависит от знаний инженера, формируемых на основе поведения тестируемого продукта в процессе проведения тестирования, степени знакомства с приложением, платформой, типами возможных сбоев и дефектов, рисками, ассоциированными с конкретным продуктом и т.п.

Разница между ad-hoc и exploratory testing в том, что теоретически, ad-hoc может провести кто угодно, а для проведения exploratory необходимо мастерство и владение определёнными техниками. При этом, определённые техники – это не только техники тестирования. Ad-hoc testing — это более интуитивное и беспорядочное тестирование, когда тестировщик просто идет и проверяет, что ему хочется.

Лекция 3

*1. Перечислите минимальный рекомендуемый набор процедур для тестирования.*

Цель тестовых процедур – определить набор последовательных действий для полного тестирования определенного набора требований для определенного тестируемого элемента.

Необходимо составить. Что тестируем, какой результат хотим получить, что имеем.

*2. Из каких шагов состоит цикл тестирования программного обеспечения?*

* Изучение и анализ предмета тестирования
* Планирование тестирования
* Исполнение тестирования
* Анализ полученных результатов (not included)

*3. В чём отличие тест-плана от тест-комплекта?*

**Тест-кейс** - документ, который описывает последовательность шагов, условий и параметров, необходимых для проверки объекта тестирования. Это проверка данных. Ваш сценарий проверки.

**Тест-комплект** – набор тестовых случаев, связанных с определенной функциональностью. Это набор тест кейсов объединенных некоей логикой (по модулям системы, по пользовательским сценариям, по уровням тестирования).

*4. В чём отличие тест -кейса от чек-листа и тест-рана?*

**Тест-Кейс** – это сценарий проверки, который описывает шаги, условия и параметры

**Чек-лист** – список тест-кейсов, содержащий позитивные проверки в рамках тестирования.

**Тест-ран** – список выполненных тестовых случаев, позволяющих определить качество разработанного программного обеспечения в конкретный момент времени.

*5. Перечислите известные техники тест-дизайна.*

**Доменное тестирование (domain testing/analysis) -** техника создания эффективных и результативных тест-кейсов в случае, когда несколько переменных могут или должны быть протестированы одновременно.

**Комбинаторное тестирование -** для нахождения багов, возникающих вследствие явных и неявных зависимостей между параметрами. При большом числе параметров и сложных зависимостей между ними количество тестов будет слишком велико для ручного тестирования.

Потому основное применение комбинаторики (и соответственно, инструментов, осуществляющих генерацию комбинаций параметров) — автоматизированное составление наборов тестовых данных по определенным законам.

**Блок-схемы** — это визуальные источники идей для тестирования.

*6. Для чего используется матрица соответствия требований?*

Матрица принятия решений — разделение на возможные варианты действий пользователя и выбор необходимых тестов. Для выборки наиболее важных и необходимых областей тестирования. Для оптимизации

Цель «Traceability Matrix» состоит в том, чтобы выяснить:

-какие требования «покрыты» тестами, а какие нет.

-избыточность тестов (одно функциональное требование покрыто большим количеством тестов).

Данный тестовый артефакт является неотъемлемой частью тестирования.

*7. Опишите метод черновика/чистовика.*

Черновик - ВСЕ идеи, связанные с тестированием, которые только могут прийти в голову, — идеи в самом широком смысле этого слова, включая идеи для тест-кейсов, сценарии, отдельные элементы сценариев (шаги и/или данные), ожидаемые результаты, вопросы для выяснения у продюсера и пр. ВСЕ идеи — даже самые на первый взгляд далекие от здравого смысла. Локальный мозговой штурм.

Чистовик- Затем мы начинаем анализировать написанное (и, если нужно, получать ответы на вопросы) и переносим на чистовик вещи, имеющие право на жизнь. Право на жизнь определяется на основании информации из спека, общения, интуиции, критериев отбора тестов, разговора с программистом и пр. При переносе на чистовик мы также уточняем наши идеи и группируем их (например, по позитивности и негативности; по функциональным направлениям и т.п.). Таким образом, как правило, первый чистовик превращается во второй черновик, и мы берем следующий лист бумаги и, надеясь, что он будет чистовиком, начинаем переносить на него наши идеи и т.д. В итоге, в один из светлых майских дней мы все-таки получаем чистовик. На основании материала из чистовика мы пишем тест-кейсы.

*8. Что такое эквивалентное разделение классов?*

Разделение на классы эквивалентности — резкое уменьшение количества тестов, за счет разделения на классы и выбора одного значения из класса эквивалентности.

*9. Как проводится анализ граничных значений?*

Анализ граничных значений - проверка значениями, находящимися на границах классов эквивалентности.

*10. Из чего состоит тест-кейс?*

Тест кейс состоит из описания последовательных шагов, условий и параметров, которые необходимы для проверки объекта тестирования. Шаги и ожидаемый результат.

**Лекция 4**

1. *Клиент-серверная архитектура*

Многие приложения строятся по этому принципу

Есть две взаимодействующие стороны – клиент и сервер, которые взаимодействуют друг с другом по сети при помощи протокола, например, HTTP и его различные версии

Сервер предоставляет некоторый ресурс или функции, а клиент хочет получить этот ресурс или использовать функции



У сервера может быть много клиентов

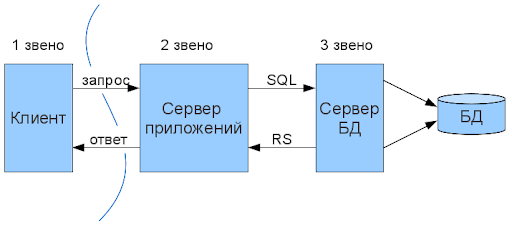
Сервер пассивен – он никогда не инициирует взаимодействие с клиентом. Инициатором всегда выступает клиент – клиент что-то просит, а сервер ему отвечает.

Клиентской программой является браузер. Браузер – приложение, предназначенное для просмотра и работы с веб-страницами

Браузер умеет:

* Отображать HTML страницы
* Применять к ним стили CSS
* Исполнять для страницы код на языке JavaScript

Возможности браузеров сильно ограничены. Поэтому все данные хранятся на сервере, на стороне клиента можно хранить только очень малые объемы данных.



1. *Кроссбраузерность*

Чтобы посетитель увидел сайт в разных браузерах таким, каким задумал его веб-мастер, проводят кроссбраузерное тестирование. По его окончании исправляют обнаруженные ошибки в коде.

Поэтому клиентскую часть приложения обязательно нужно проверять на всех целевых браузерах

1. *Редиректы*

Редирект – это перенаправление клиента сервером на другой адрес

Виды редиректов:

* 301 Moved Permanently – перемещено навсегда

Это постоянный редирект

* 302 Moved Temporarily – перемещено временно

1. *Cookies*

Cookie (куки) – небольшой фрагмент данных, который может храниться на стороне клиента, и который прикрепляется к каждому запросу

Куки чаще всего применяются для:

* Аутентификации - «залогиненность»
* Хранения сессии - данных, связанных с текущим пользователем в этом сеансе. Например, корзина в интернет-магазине
* Хранения персональных настроек – выбор конкретного языка на сайте, или расположения элементов, или темы оформления

Cookie – это пары ключ-значение (Name и Value). Cookie привязываются к определенному адресу (Path), либо к домену в целом (Domain). Если запрос относится к этому адресу, то клиент будет отправлять cookie, относящиеся к этому адресу. У cookie есть срок истечения (Expires) – когда наступит указанное время, то cookie считается истекшей, и автоматически удаляется браузером.

1. *API*

API (Application Programming Interface) – набор функций, предоставляемых приложением. Можно представить себе API как сайт без графического интерфейса.

Например, есть некоторый сервер у него по некоторым адресам доступны некоторые команды, это и есть API.

1. *RPC*

RPC (Remote procedure call) – возможность вызывать команды другой программы (которая обычно находится на другом компьютере).

Отличие от REST только в том, что RPC ориентирован не на ресурсы, а на функции, которые может выполнять API.

1. *SOAP*

SOAP – протокол для обмена XML сообщениями, в основном используется для RPC (удаленный вызов процедур) при работе с веб-сервисами.

**Структура SOAP сообщения**

* SOAP сообщение называют конвертом (envelope)
* В конверте есть заголовок (header) и тело (body)

1. *Валидация*

Валидация – это проверка корректности данных. Что данные имеют нужный тип (число/строка/email и т.д.), что все необходимые данные заполнены, что не превышена максимальная длина данных и т.д.

Валидация бывает:

* Серверная – на стороне сервера
* Клиентская – на стороне клиента