Требования. Анализ требований. Тестирование документации. Виды и направления тестирования.

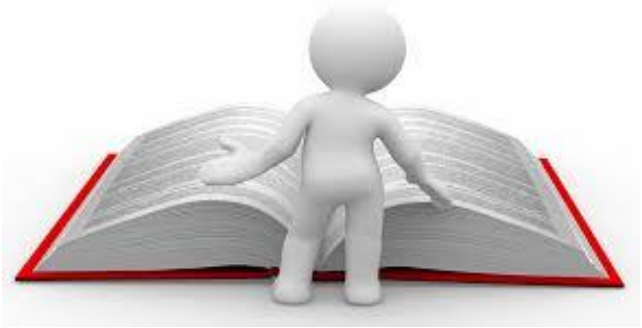
План лекции

- Что такое требование?
- Виды требований
- Источники и пути выявления требований
- Анализ требований
- Тестирование требований
- Виды документации
- Методы тестирования
- Типы тестирования
- Уровни тестирования
- Виды тестирования

Требование

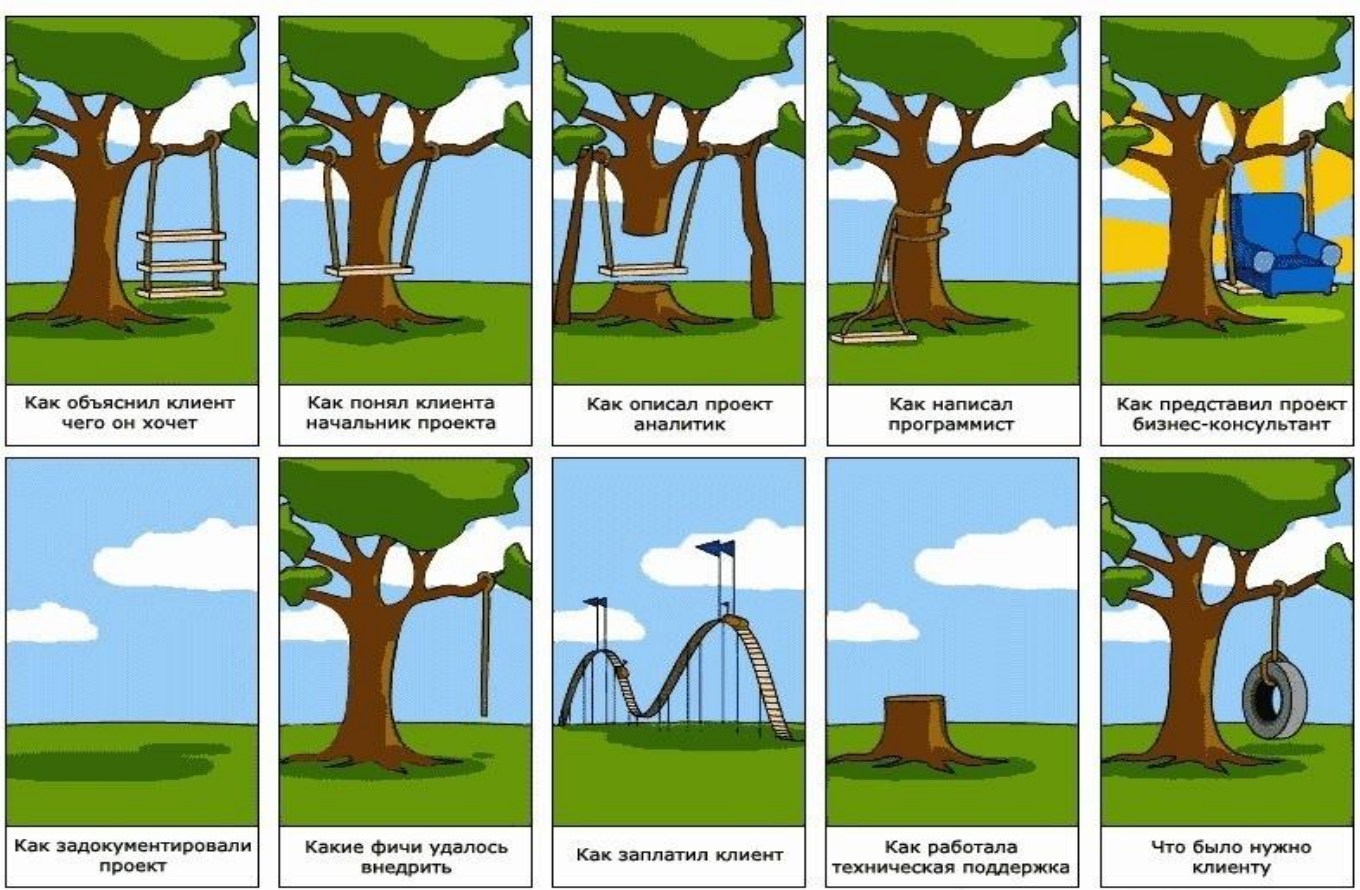
это описание того, какие функции и с соблюдением каких условий должно выполнять приложение в процессе решения полезной для пользователя задачи

Важность требований:



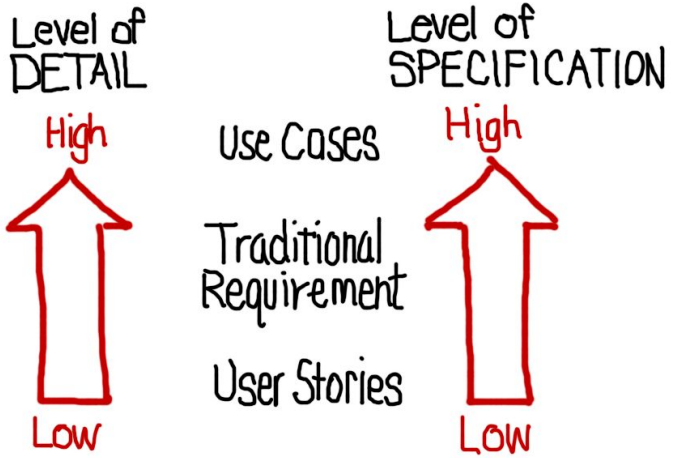
- Позволяют понять, что и с соблюдением каких условий система должна делать.
- Предоставляют возможность оценить масштаб изменений и управлять изменениями.
- Являются основой для формирования плана проекта (в том числе плана тестирования).
- Помогают предотвращать или разрешать конфликтные ситуации.
- Упрощают расстановку приоритетов в наборе задач.
- Позволяют объективно оценить степень прогресса в разработке проекта.

Важность требований



Источники и пути выявления требований

- Use case.
- Совещание.
- Анкетирование.
- Интервью, опросы.
- “Мозговой штурм”, семинары.
- Анализ моделей деятельности.
- Анализ конкурентных продуктов.
- Анализ нормативной документации.
- Наблюдение за производственной деятельностью.
- Анализ статистики использования предыдущих версий системы.



Параметры тестирования требований

- **Четкость и ясность** - требования должны давать предельно ясную информацию о том, как должен работать каждый отдельный модуль и весь продукт в целом.
- **Актуальность** - необходимость поддержания актуальности требований обусловливается внесением изменений на протяжении разработки ПО.
- **Логика** - работа системы должна быть логичной.
- **Возможные сценарии** - требования должны содержать позитивные и негативные варианты использования системы.
- **Интеграция** - описание схемы взаимодействия разрабатываемого продукта со сторонней системой.

Основные принципы тестирования требований

- Тестирование требований проводиться до старта разработки ПО (ранее тестирование снижает итоговую стоимость проекта).
- Тестирование требований проводится и аналитиками, и тестировщиками (для достижения лучшего результата создание документации и ее тестирование должны проводить разные участники команды).
- Выявленные во время тестирования требований баги должны быть занесены в баг-трекинговую систему.
- О выявленных во время тестирования требований багах необходимо предупредить команду разработчиков (в том случае, если тестирование ведется параллельно с разработкой).
- Глубина тестирования требований зависит от уровня их детализации (детализация требований зависит от уровня проекта).

Свойства качественных требований

- **Атомарность, единичность** - требование является атомарным, если его нельзя разбить на отдельные требования без потери завершённости и оно описывает одну и только одну ситуацию.
- **Непротиворечивость, последовательность** - требование не должно содержать внутренних противоречий и противоречий другим требованиям и документам.
- **Недвусмысленность** - требование должно допускать только однозначное объективное понимание и быть атомарным в плане невозможности различной трактовки сочетания отдельных фраз.
- **Обязательность, нужность, актуальность** - если требование не является обязательным к реализации, то оно должно быть просто исключено из набора требований. Если требование нужное, но «не очень важное», для указания этого факта используется указание приоритета. Также должны быть исключены (или переработаны) требования, утратившие актуальность.

Тестирование требований

- **Прослеживаемость** - бывает вертикальной и горизонтальной. Вертикальная прослеживаемость позволяет соотносить между собой требования на различных уровнях требований, горизонтальная - соотносить требование с тест-планом, тест-кейсами, архитектурными решениями.
- **Модифицируемость** - это свойство характеризует внесения изменений в отдельные требования и в набор требований, искомую информацию легко найти, а её изменение не приводит к нарушению иных описанных в этом перечне свойств.
- **Проранжированность по важности, стабильности, срочности** - важность характеризует зависимость успеха проекта от успеха реализации требования. Стабильность характеризует вероятность того, что в ближайшем будущем в требование не будет внесено никаких изменений. Срочность определяет распределение по времени усилий проектной команды по реализации того или иного требования.
- **Корректность и проверяемость** - эти свойства вытекают из соблюдения всех вышеперечисленных. Проверяемость подразумевает возможность создания объективного тест-кейса (тест-кейсов), однозначно показывающего, что требование реализовано верно и поведение приложения в точности соответствует требованию.

Техники тестирования требований

Тестирование документации и требований относится к разряду нефункционального тестирования.
Основные техники такого тестирования в контексте требований:

- **Взаимный просмотр** - является одной из наиболее активно используемых техник тестирования требований и может быть представлен в одной из трёх следующих форм:

  - Беглый просмотр - обмен результатами работы между двумя и более авторами с тем, чтобы коллега высказал свои вопросы и замечания;
  - Технический просмотр - выполняется группой специалистов;
  - Формальная инспекция - структурированный, систематизированный и документируемый подход к анализу документации, с привличением большого количества специалистов.

- **Вопросы** - Вопросы к представителям заказчика или обращения к справочной информации.

Виды документации

Продуктная документация (product documentation, development documentation) используется проектной командой во время разработки и поддержки продукта. Она включает:

- **План проекта** (project management) и в том числе тестовый план (test).
- **Требования к программному продукту** (product requirements document) и функциональные спецификации (functional specifications document, FSD; software requirements specification, SRS).
- **Архитектуру и дизайн** (architecture and design).
- **Тест-кейсы и наборы тест-кейсов** (test cases, test suites).
- **Технические спецификации** (technical specifications), такие как схемы баз данных, описания алгоритмов, интерфейсов и т.д.

Виды документации

Проектная документация (project documentation) включает в себя как продуктную документацию, так и некоторые дополнительные виды документации и используется не только на стадии разработки, но и на более ранних и поздних стадиях (например, на стадии внедрения и эксплуатации).

- **Пользовательскую и сопроводительную документацию** (user and accompanying documentation), такую как встроенная помощь, руководство по установке и использованию, лицензионные соглашения и т.д.
- **Маркетинговую документацию** (market requirements document, MRD),которую представители разработчика или заказчика используют как на начальных этапах (для уточнения сути и концепции проекта), так и на финальных этапах развития проекта (для продвижения продукта на рынке).

White/Black/Grey Box-тестирование

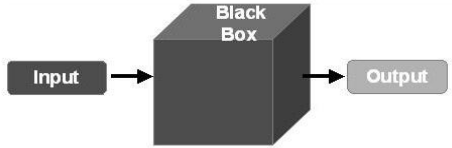
Типы тестирования, которые отличаются знанием внутреннего устройства объекта тестирования.

Black-box тестирование:

Тестирование методом «черного ящика», также известное как тестирование, основанное на спецификации или тестирование поведения – техника тестирования, основанная на работе исключительно с внешними интерфейсами тестируемой системы.

Black-box тестирование по ISTQB:

- тестирование, как функциональное, так и нефункциональное, не предполагающее знания внутреннего устройства компонента или системы;
- тест-дизайн, основанный на технике черного ящика – процедура написания или выбора тест-кейсов на основе анализа функциональной или нефункциональной спецификации компонента или системы без знания ее внутреннего устройства.

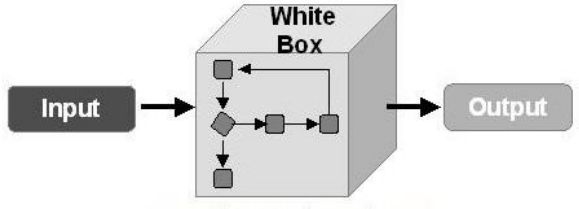


White-box тестирование:

Тестирование методом белого ящика (также прозрачного, открытого, стеклянного ящика или же основанное на коде или структурное тестирование) – метод тестирования программного обеспечения, который предполагает, что внутренняя структура/устройство/реализация системы известны тестировщику.

White-box тестирование по ISTQB:

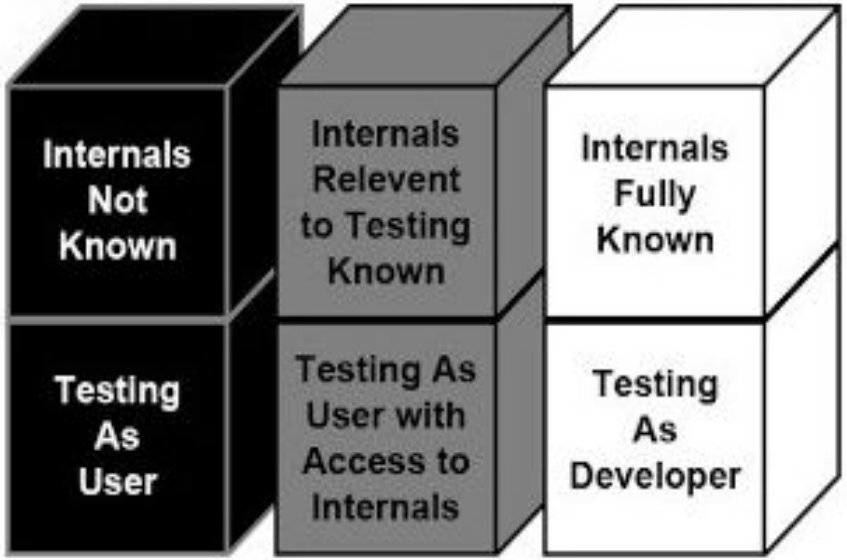
- тестирование, основанное на анализе внутренней структуры компонента или системы;
- тест-дизайн, основанный на технике белого ящика – процедура написания или выбора тест-кейсов на основе анализа внутреннего устройства системы или компонента.



Grey-box тестирование:

Тестирование методом серого ящика – метод тестирования программного обеспечения, который предполагает комбинацию White Box и Black Box подходов. То есть внутреннее устройство программы нам известно лишь частично

Предполагается, например, доступ ко внутренней структуре и алгоритмам работы ПО для написания максимально эффективных тест-кейсов, но само тестирование проводится с помощью техники черного ящика, то есть с позиции пользователя.



Статическое и динамическое тестирование

По критерию запуска программы (исполняется ли программный код) выделяют два типа тестирования:

- **Статическое тестирование** - тип тестирования, который предполагает, что программный код во время тестирования не будет выполняться. Выделяют несколько видов, таких как:

  - вычитка исходного кода программы;
  - проверка требований.

- **Динамическое тестирование** - тип тестирования, который предполагает запуск программного кода. Таким образом, анализируется поведение программы во время ее работы. Включает разные подвиды, каждый из которых зависит от:

  - доступ к коду (Black\White\Grey box тестирование);
  - уровень тестирования (системное , приемочное тестирование);
  - сферы использования приложения (функциональное, нагрузочное

Ручное и автоматизированное тестирование

Выделяют три типа тестирования:

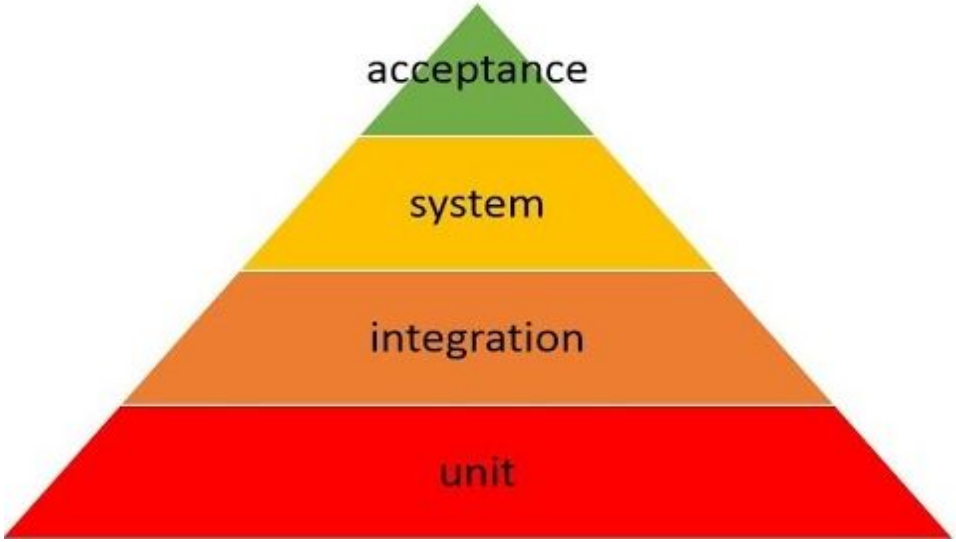
- **Ручное тестирование** (manual) - выполнение кейсов выполняется вручную.
- **Полуавтоматизированное тестирование** (half-automated)- ручное тестирование сочетается с автоматическим.
- **Автоматизированное** (automated) - предполагает использование специального программного обеспечения (помимо тестируемого) для контроля выполнения тестов и сравнения ожидаемого фактического результата работы программы. Существует несколько видов автоматизированного тестирования:
  - автоматизация тестирования кода (code-driven testing);
  - автоматизация тестирования графического пользовательского интерфейса (graphical user interface testing);
  - автоматизация тестирования API (Application Programming Interface).

Уровни тестирования

Тестирование на разных уровнях производится на протяжении всего жизненного цикла разработки и сопровождения программного обеспечения. Уровень тестирования определяет то, над чем производятся тесты: над отдельным модулем, группой модулей или системой, в целом. Проведение тестирования на всех уровнях системы - это залог успешной реализации и сдачи проекта.

Классификация по ISTQB:

- Unit testing
- Integration testing
- System testing
- Acceptance testing



Unit or Component testing

Модульное или компонентное 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.).

Обычно компонентное (модульное) тестирование проводится вызывая код, который необходимо проверить и при поддержке сред разработки, таких как фреймворки (frameworks - каркасы) для модульного тестирования или инструменты для отладки.

Все найденные дефекты, как правило исправляются в коде без формального их описания в системе менеджмента багов (Bug Tracking System).

Разница между компонентным и модульным тестированием - в компонентном тестировании, в качестве параметров функций, используют реальные объекты и драйверы, а в модульном тестировании - конкретные значения.

Integration testing

Интеграционное тестирование предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами).

Уровни интеграционного тестирования:

- **Компонентный интеграционный уровень** (Component integration level) - проверяется взаимодействие между компонентами системы после проведения компонентного тестирования;
- **Системный интеграционный уровень** (System integration level) - проверяется взаимодействие между разными системами после проведения системного тестирования.

Integration testing

Подходы к интеграционному тестированию:

- **Снизу вверх (**Bottom Up integration) - Все низкоуровневые модули собираются воедино и затем тестируются. После чего собирается следующий уровень модулей.
- **Сверху вниз** (Top Down integration) - Сначала тестируются все высокоуровневые модули, затем постепенно добавляются низкоуровневые.
- **Большой взрыв** (“Big Bang integration) - Все разработанные модули собираются вместе в виде законченной системы, а затем проводится интеграционное тестирование.

System testing

Основной задачей **системного тестирования** является проверка как функциональных, так и не функциональных требований , наличие дефектов в системе в целом.

Можно выделить два подхода к системному тестированию:

- **На базе требований** (requirements based) - для каждого требования пишутся тестовые случаи (test cases), проверяющие выполнение данного требования.
- **На базе случаев использования** (use case based) - на основе представления о способах использования продукта создаются случаи использования системы (Use Cases). По конкретному случаю использования можно определить один или более сценариев. На проверку каждого сценария пишутся тест-кейсы (test cases), которые должны быть протестированы.

Acceptance testing

Приемочное тестирование это формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:

- определения удовлетворения системой приемочным критериям;
- вынесения решения заказчиком или другим уполномоченным лицом принятия приложения.

Виды приемочного тестирования:

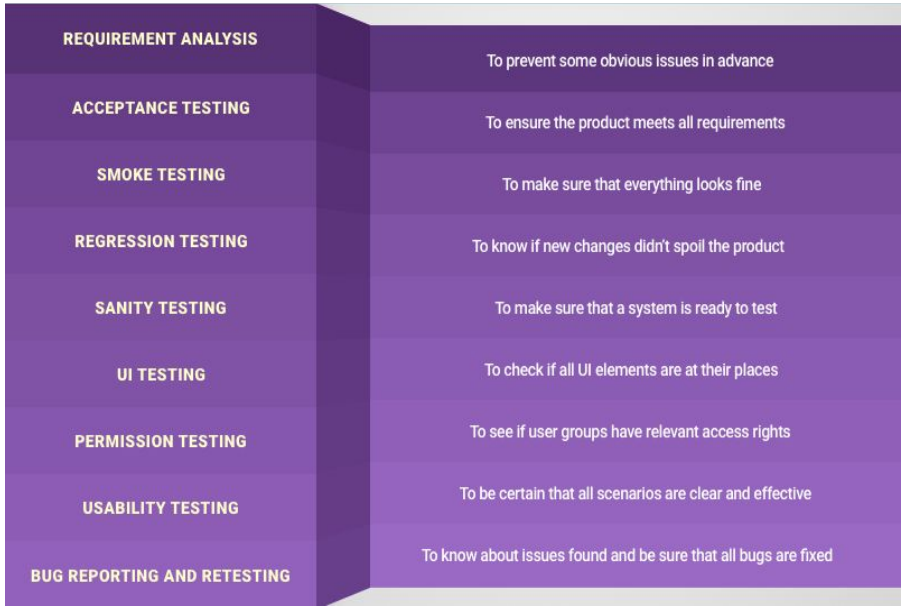
- Пользовательское приемочное тестирование (User acceptance testing);
- Альфа тестирование (Alpha testing) - проводится разработчиками или командой тестировщиков на на поздней стадии разработки с имитацией реального использования продукта;
- Бета тестирование (Beta testing) - проводится небольшой группой авторизированных пользователей, с целью собрать первые фидбеки.

Виды тестирования

Функциональные:
- Функциональные тесты
- Тестирование безопасности
- Тестирование взаимодействия

Нефункциональные:
- Тестирование производительности
- Тестирование установки
- Тестирование удобства пользования
- Тестирование на отказ и восстановление
- Конфигурационное тестирование

Связанные с изменениями:
- Дымовое тестирование
- Регрессионное тестирование
- Тестирование сборки
- Санитарное тестирование или проверка согласованности/исправности



Функциональные виды тестирования

По ISTQB: Тестирование, основанное на анализе спецификации функциональности компонента или системы.

Функциональность - Способность программного продукта обеспечивать функции, которые соответствуют установленным и предполагаемым потребностям, при использовании ПО в определенных условиях.

Преимущества функционального тестирования:
- имитирует фактическое использование системы.

Недостатки функционального тестирования:
- возможность упущения логических ошибок в программном обеспечении;
- вероятность избыточного тестирования.

Достаточно распространенной является автоматизация функционального тестирования.

Функциональные тесты

Основываются на функциях, выполняемых системой, и могут проводиться на всех уровнях тестирования (компонентном, интеграционном, системном, приемочном). Как правило, эти функции описываются в требованиях, функциональных спецификациях или в виде случаев использования системы (use cases).

Может проводиться в двух аспектах:

- Требования.
- Бизнес-процессы.

Тестирование в аспекте «требования» использует спецификацию функциональных требований к системе, как основу для дизайна тестовых случаев (Test Cases). Это позволит сфокусироваться и не упустить при тестировании наиболее важный функционал.

Тестирование в аспекте «бизнес-процессы» использует знание бизнес-процессов, которые описывают сценарии ежедневного использования системы. В этом аспекте тестовые сценарии (test scripts), как правило, основываются на случаях использования системы (use cases).

Тестирование безопасности

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.

Общая стратегия безопасности основывается на трех основных принципах:

- Конфиденциальность.
- Целостность.
- Доступность.

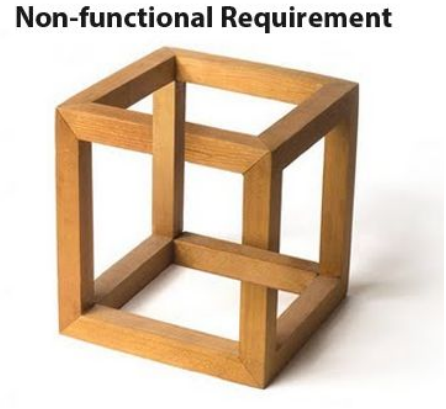
Тестирование взаимодействия

Тестирование взаимодействия (Interoperability Testing) – это функциональное тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя тестирование совместимости (compatibility testing) и интеграционное тестирование (integration testing).

Программное обеспечение с хорошими характеристиками взаимодействия может быть легко интегрировано с другими системами, не требуя каких-либо серьезных модификаций. В этом случае, количество изменений и время, требуемое на их выполнение, могут быть использованы для измерения возможности взаимодействия.

Нефункциональные виды тестирования

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами. То есть, тестирование атрибутов компонента или системы, не относящихся к функциональности, например надежность, эффективность, практичность, сопровождаемость и переносимость. В целом, это тестирование того, как система работает.



Тестирование производительности

Задачей тестирования производительности (Performance testing) является определение масштабируемости приложения под нагрузкой.

Виды:

- Стрессовое тестирование (Stress testing).
- Объемное тестирование (Volume testing).
- Тестирование стабильности или надежности(Stability / Reliability Testing).
- Нагрузочное тестирование (Load).

Тестирование установки

Тестирование установки (Installation testing) направлено на действия, которые нужно совершить пользователю для установки и настройки ПО.

Может включать в себя:

- полную или частичную установку;
- модификацию установки/удаления.

Тестирование удобства использования

Тестирование удобства пользования (Usability Testing) - это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.

Тестирование удобства пользования дает оценку уровня удобства использования приложения по следующим пунктам:

- Производительность, эффективность (efficiency).
- Правильность (accuracy).
- Активизация в памяти (recall).
- Эмоциональная реакция (emotional response).

Тестирование на отказ и восстановление

Тестирование на отказ и восстановление (Failover and Recovery Testing) проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками программного обеспечения, отказами оборудования или проблемами связи (например, отказ сети).

Целью данного вида тестирования является проверка систем восстановления (или дублирующих основной функционал систем), которые, в случае возникновения сбоев, обеспечат сохранность и целостность данных тестируемого продукта.

Конфигурационное тестирование

Конфигурационное тестирование(Configuration Testing) — специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.)

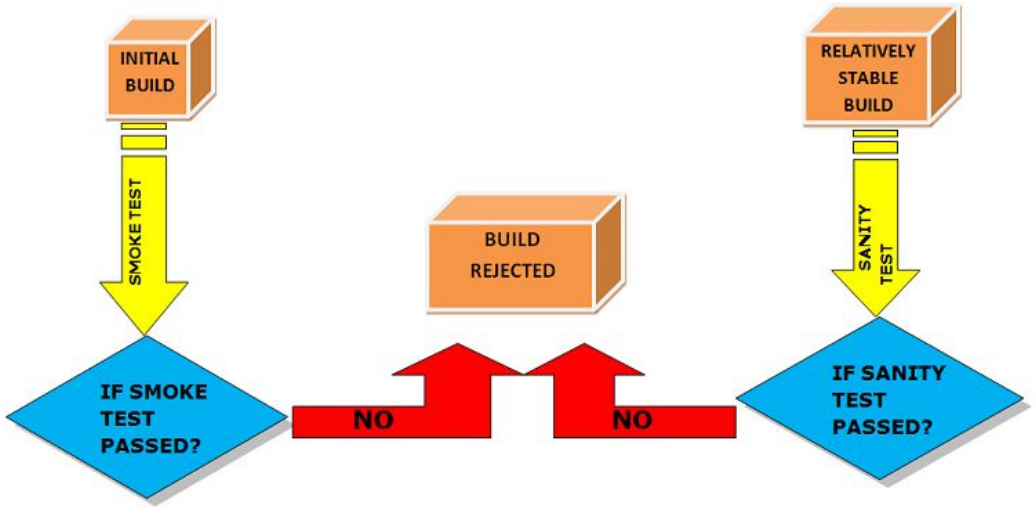
В зависимости от типа проекта конфигурационное тестирование может иметь разные цели:

- Проект по профилированию работы системы.
- Проект по миграции системы с одной платформы на другую.

Примечание: В ISTQB Syllabus вообще не говорится о таком виде тестирования, как конфигурационное. Согласно глоссарию, данный вид тестирования рассматривается там как тестирование портируемости(portability testing: The process of testing to determine the portability of a software product.).

Связанные с изменениями виды тестирования

После проведения необходимых изменений, таких как исправление багов/дефектов, программное обеспечение должно быть перетестировано для подтверждения того факта, что проблема была действительно решена. Ниже перечислены виды тестирования, которые необходимо проводить после установки программного обеспечения, для подтверждения работоспособности приложения или правильности осуществленного исправления дефекта.



Дымовое тестирование

Smoke testing это выборка из общего числа запланированных тестовых сценариев, покрывающая основную функциональность компонента или системы. Проводится с целью удостовериться, что базовые функции программы в целом работают корректно, без углубления в детали.

Аналогами дымового тестирования являются Build Verification Testing и Acceptance Testing, выполняемые на функциональном уровне командой тестирования, по результатам которых делается вывод о том, принимается или нет установленная версия программного обеспечения в тестирование, эксплуатацию или на поставку заказчику.

Для облегчения работы, экономии времени и людских ресурсов рекомендуется внедрить автоматизацию тестовых сценариев для дымового тестирования.

Санитарное тестирование или проверка согласованности/исправности

Санитарное тестирование (Sanity Testing) - это узконаправленное тестирование, достаточное для доказательства того, что конкретная функция работает согласно заявленным
в спецификации требованиям.
Является подмножеством регрессионного тестирования. Используется для определения работоспособности определенной части приложения после изменений произведенных в ней или окружающей среде. Обычно выполняется вручную.

Различие между дымовым и санитарным тестированием

Эти виды тестирования имеют "Вектора движения", направления в разные стороны. Санитарное тестирование (**Sanity testing**) направлено **вглубь** проверяемой функции, в то время как дымовое (**Smoke testing**) направлено **вширь**, для покрытия тестами как можно большего функционала в кратчайшие сроки.

Регрессионное тестирование

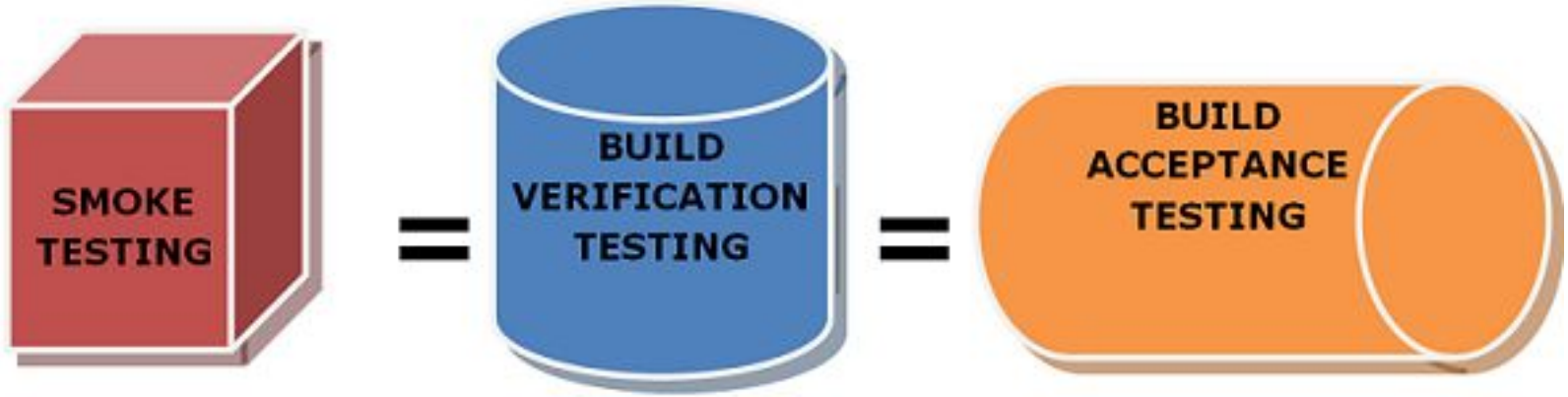
Регрессионное тестирование (Regression Testing) - это вид тестирования, направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, веб-сервер или сервер приложения), для подтверждения того факта, что существующая ранее функциональность работает как и прежде. Регрессионными могут быть как функциональные, так и нефункциональные тесты.

Сам по себе термин "регрессионное тестирование", в зависимости от контекста использования, может иметь разный смысл. Сэм Канер, к примеру, описал 3 основных типа регрессионного тестирования:

- Регрессия багов (Bug regression).
- Регрессия старых багов (Old bugs regression).
- Регрессия побочного эффекта (Side effect regression).

Тестирование сборки

Тестирование сборки (Build Verification Test) , направленное на определение соответствия выпущенной версии критериям качества для начала тестирования. По своим целям является аналогом дымового тестирования, направленного на приемку новой версии в дальнейшее тестирование или эксплуатацию. Вглубь оно может проникать дальше, в зависимости от требований к качеству выпущенной версии.



Спасибо за внимание!

www.andersenlab.com

Список литературы

- https://qaevolution.ru/yuzabiliti-testirovanie/kratkij-konspekt-poleznyx-znanij-po-testirovaniyu-dokumentacii/
- https://habr.com/post/346290/
- https://www.altexsoft.com/blog/business/software-documentation-types-and-best-practices/
- https://www.guru99.com/types-of-software-testing.html