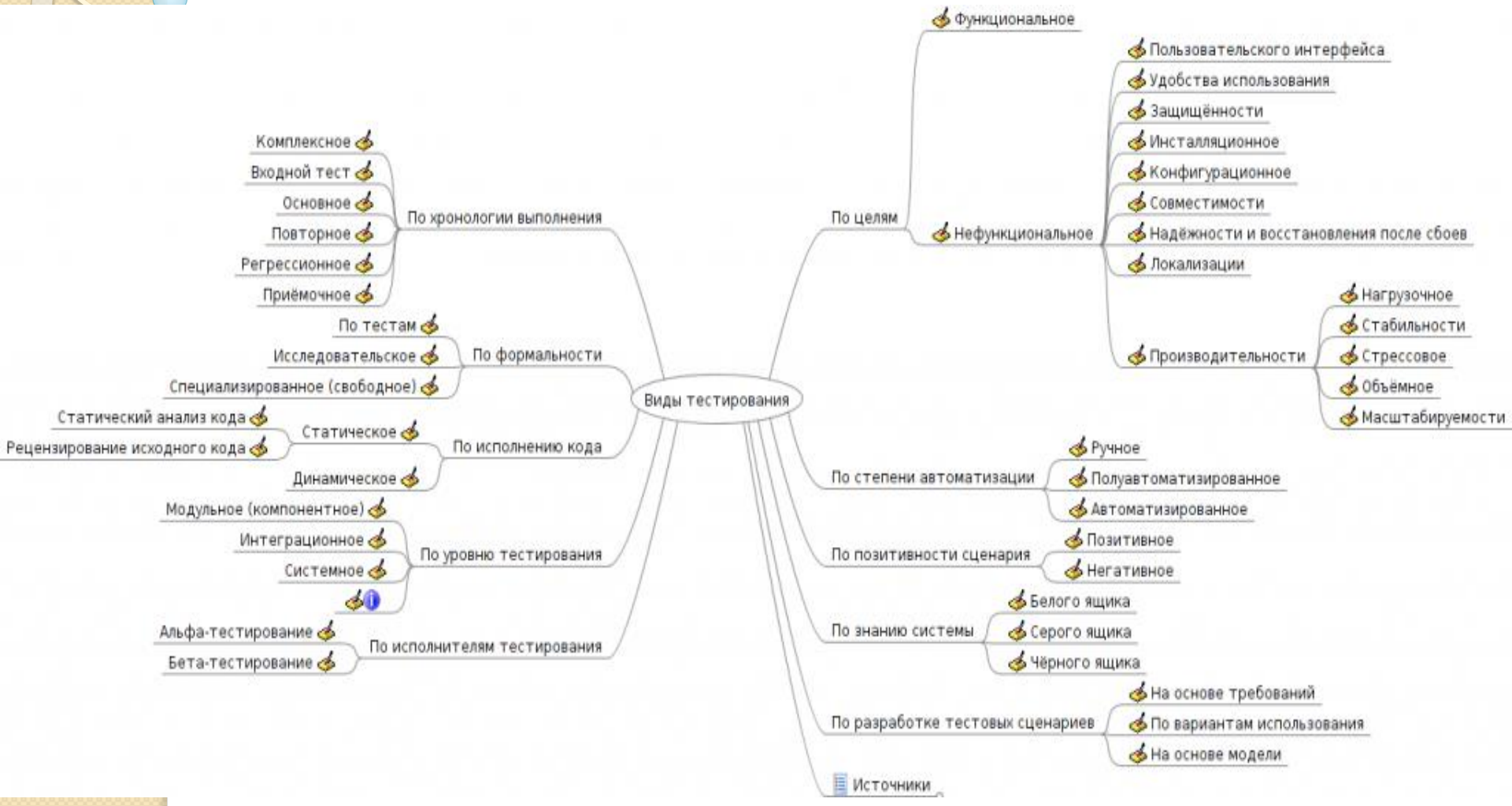
Тестирование

Направления.
Методы.
Уровни.
Виды.

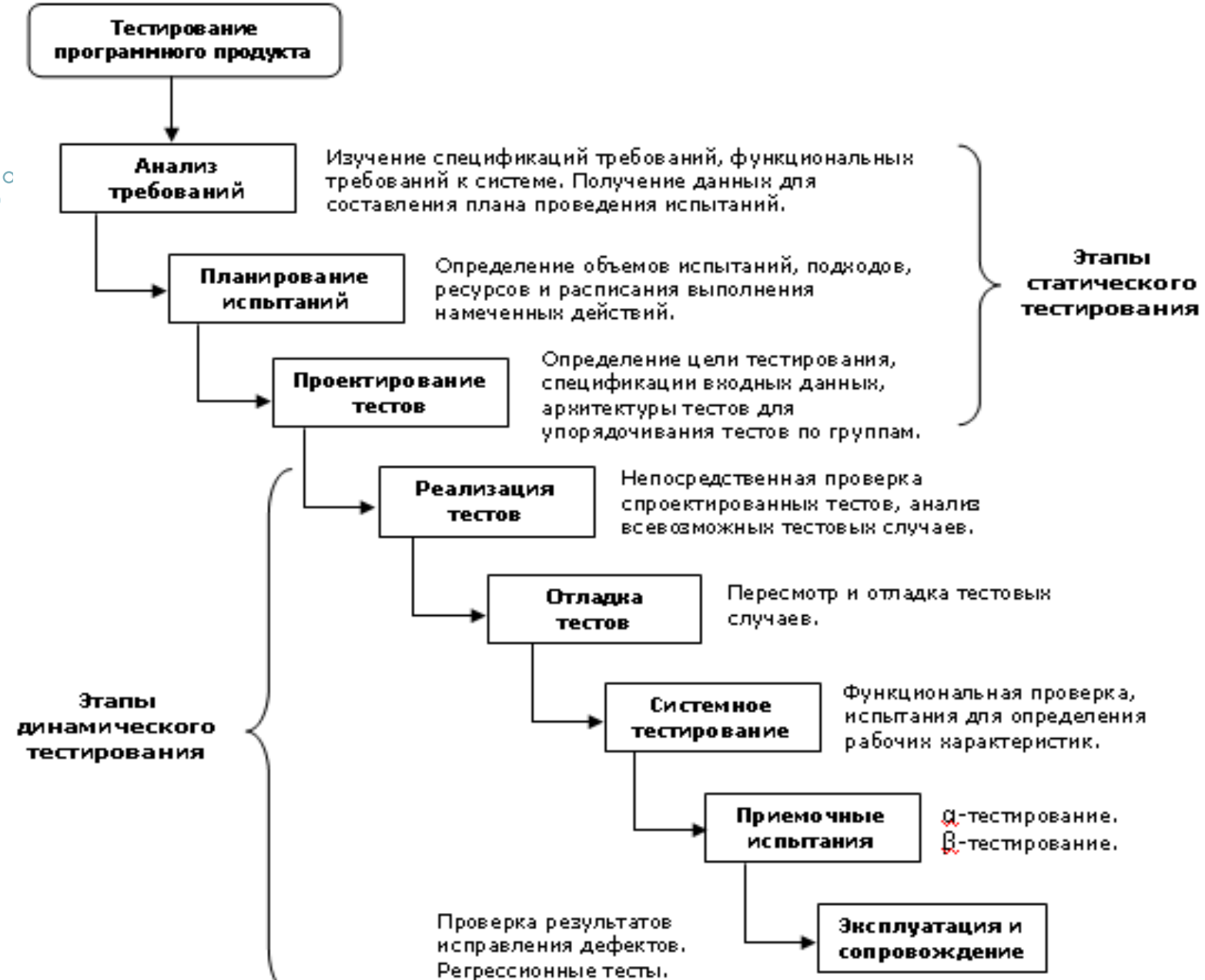
Классификация тестирования



Направления тестирования

Классификация по запуску кода на исполнение

- **Статическое** (без запуска программного кода продукта)
- **Динамическое** (при запущенной системе)


Тестирование программного продукта
Анализ требований
Изучение спецификаций требований, функциональных требований к системе. Получение данных для составления плана проведения испытаний.
Планирование испытаний
Определение объемов испытаний, подходов, ресурсов и расписания выполнения намеченных действий.
Проектирование тестов
Определение цели тестирования, спецификации входных данных, архитектуры тестов для упорядочивания тестов по группам.
Этапы статического тестирования
Реализация тестов
Непосредственная проверка спроектированных тестов, анализ всевозможных тестовых случаев.
Отладка тестов
Пересмотр и отладка тестовых случаев.
Системное тестирование
Функциональная проверка, испытания для определения рабочих характеристик.
Приемочные испытания
α-тестирование.
β-тестирование.
Эксплуатация и сопровождение
Проверка результатов исправления дефектов. Регрессионные тесты.
Этапы динамического тестирования

Статическое тестирование

Раннее выявление дефектов (дешевле)

Проверяется:

- код
- требования
- спецификации
- архитектура
- дизайн

Динамическое тестирование

Более позднее выявление / более дорогое устранение дефектов (спецификации написаны, архитектура разработана, интерфейс программы тоже готов)

Проверяется:

- ✓ все характеристики качества запущенного приложения

Методы тестирования

Классификация по допуску к коду (знание системы)

- **Метод белого ящика** *(white box testing)* - смотрим код
- **Метод черного ящика** *(black box testing)* - не заглядываем внутрь системы
- **Метод серого ящика** *(gray box testing)* – частично заглядываем в систему

Метод белого ящика

Тестирование "белый ящик" выполняется с целью обнаружения проблем во внутренней структуре программы.

Объектами тестирования в этом случае являются данные, полученные путем анализа логики программы.

Общая задача такого тестирования - обеспечить проверку каждого шага по алгоритму программы.

Обычно тестирование "белый ящик" проводится разработчиками.

Метод черного ящика

Под «чёрным ящиком» понимается объект исследования, внутреннее устройство которого неизвестно, т.е. как именно работает программа считается несущественным.

Этот подход до сих пор является самым распространенным.

Объектами тестирования в этом случае являются потоки входных и выходных данных. Это позволяет определять «правильность» работы ПО в соответствии с функциональными требованиями к продукту.

Метод серого ящика

В исследованиях по методу "серого ящика» объединены методы "белого ящика" и способы тестирования с помощью входных данных по методу "черного ящика".

При работе этим методом подразумевается, что тестировщик имеет доступ к внутреннему устройству программы, но тестирование производит с точки зрения конечного пользователя.

Удачным примером простого анализа по методу "серого ящика" является запуск программы внутри отладчика и подача на вход этой программы различных данных.

Уровни тестирования
Тестовое покрытие

Это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода

Уровни тестирования
Тестовое покрытие

Определяется:

-**Глубиной** тестирования
-**Шириной** тестирования
-**Целями** тестирования
-Профессиональным уровнем тестировщика

Уровни тестирования
Тестовое покрытие

Определяется:

-**Глубиной** тестирования
-**Шириной** тестирования
-**Целями** тестирования
-Профессиональным уровнем тестировщика

Уровни тестирования

по степени важности тестируемых функций «по глубине тестирования»

Отражает вид проверок, которые производились для модуля/функции

- **Дымовое** *(smoke test)*
- **Критического пути** *(critical path test)*
- **Расширенное** *(extended test)*

Дымовое тестирование

Направлено на проверку самой главной, самой важной, самой ключевой функциональности, неработоспособность которой делает бессмысленной дальнейшее тестирование.

Самый первый и быстрый уровень.

Если тесты этого уровня не проходят, то тестирование прекращается.

Пример: запускается ли программа, устанавливается или удаляется ли она вообще.

Тест критического пути

Проверка функциональности, используемой типичными пользователями в типичной повседневной деятельности.

В рамках данного тестирования, как правило, проверяется большинство требований, предъявляемых к программному продукту.

Т.е. проверяем правильность работы часто используемых функций, *например*: возможность войти на сайт, выбрать аккаунты для публикации, создать и опубликовать пост.

Расширенное тестирование

Проверка всей (остальной) функциональности, заявленной в требованиях.

Это углубленный тест, при котором проверяется нестандартное использование программного продукта.

Прогоняются сложные, логически запутанные сценарии, совершаются действия, которые конечный пользователь будет совершать редко.

Пример: границы переполнения массивов, ввод специальных символов

Классификация тестирования по (убыванию) степени важности тестируемых функций

Уровни тестирования
по уровню детализации приложения («вширь»)

Отражает количество модулей/функций приложения, которые были протестированы.

- **Компонентное/модульное** *(component / unit testing)*
- **Интеграционное** *(integration testing)*
- **Системное** *(system testing)*
- **Приёмочное** *(Acceptance testing)*

Компонентное / модульное тестирование

Направлено на проверку отдельных небольших частей приложения, которые (как правило) можно исследовать изолированно от других подобных частей.

При выполнении данного тестирования могут проверяться отдельные функции или методы классов, сами классы, взаимодействие классов, небольшие библиотеки, отдельные части приложения.

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

Интеграционное тестирование

Направлено на проверку взаимодействия между несколькими частями приложения (каждая из которых, в свою очередь, проверена отдельно на стадии модульного тестирования).

«На стыке» их взаимодействия часто возникают проблемы, которые выявляет интеграционное тестирование.

Системное тестирование

Направлено на проверку всего приложения как единого целого, собранного из частей, проверенных на двух предыдущих стадиях.

Здесь не только выявляются дефекты «на стыках» компонентов, но и появляется возможность полноценно взаимодействовать с приложением с точки зрения конечного пользователя.

Во время тестирования рекомендуется использовать окружение максимально приближенное к тому, на которое будет установлен продукт после выдачи.

Приёмочное тестирование

Формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:

- определения удовлетворяет ли система приемочным критериям;
- вынесения решения заказчиком или другим уполномоченным лицом принимается приложение или нет.

Приемочное тестирование выполняется на основании набора типичных тестовых случаев и сценариев, разработанных на основании требований к данному приложению.

Связь уровней тестирования

Классификация по привлечению конечных пользователей

Альфа-тестирование (*alpha testing*) выполняется внутри организации-разработчика с возможным частичным привлечением конечных пользователей. Может являться формой внутреннего приёмочного тестирования. Суть этого вида: продукт уже можно периодически показывать внешним пользователям, но он ещё достаточно «сырой», потому основное тестирование выполняется организацией-разработчиком.

Бета-тестирование (*beta testing*) выполняется вне организации-разработчика с активным привлечением конечных пользователей/заказчиков. Может являться формой внешнего приёмочного тестирования. Суть этого вида: продукт уже можно открыто показывать внешним пользователям, он уже достаточно стабилен, но проблемы всё ещё могут быть, и для их выявления нужна обратная связь от реальных пользователей.

Гамма-тестирование (*gamma testing*) — финальная стадия тестирования перед выпуском продукта, направленная на исправление незначительных дефектов, обнаруженных в бета-тестировании. Как правило, также выполняется с максимальным привлечением конечных пользователей/заказчиков. Может являться формой внешнего приёмочного тестирования. Суть этого вида: продукт уже почти готов, и сейчас обратная связь от реальных пользователей используется для устранения последних недоработок.

Виды тестирования
Классификация по целям тестирования

Виды тестирования по преследуемым целям разбиваются на группы:

1. **Функциональные виды**
2. **Нефункциональные виды**
3. **Виды, связанные с изменениями**

1. Функциональные виды

Тестируются функции и взаимодействие с другими системами

Разработка тестов на всех уровнях тестирования:

- компонентное/интеграционное/системное;
- **приемочное**/тестирование критического пути/расширенное тестирование

Изучается, как внешне ведет себя система

1.Функциональные виды

Одни из самых распространенных видов функциональных тестов:

- **Функциональное тестирование**
  (Functional testing)
- **Тестирование безопасности**
  (Security and Access Control Testing)
- **Тестирование взаимодействия**
  (Interoperability Testing)

1.Функциональные виды.

Функциональное тестирование

Процесс проверки программного обеспечения, сконцентрированный на анализе соответствия программного обеспечения требованиям и спецификациям.

Цели функционального тестирования:

- ✓ тестирование каждой функции в отдельности;
- ✓ обнаружение дефекта в программном продукте;
- ✓ определение степени соответствия программного продукта требованиям и ожиданиям заказчика;
- ✓ принять решение о возможности передачи продукта заказчику.

1.Функциональные виды.

Тестирование безопасности

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.

http://protesting.ru/testing/types/security.html

1.Функциональные виды.

Тестирование взаимодействия

Это функциональное тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя:

- тестирование совместимости *(compatibility testing)*
- интеграционное тестирование *(integration testing)*.

2. Нефункциональные виды

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами.

В целом, это тестирование того, "Как" система работает.

2. Нефункциональные виды

- Тестирование производительности (*performance testing*)
- Тестирование установки (*installation testing*)
- Тестирование удобства пользования (*usability testing*)
- Тестирование на отказ и восстановление (*failover & recovery testing*)
- Конфигурационное тестирование (*configuration testing*)
- Тестирование интернационализации (*internationalization testing*)
- Локализационное тестирование (*localization testing*)
- Тестирование документации (*document testing*)
- Тестирование совместимости (*compatibility testing*)

2.Нефункциональные виды.

Тестирование производительности

Тестирование, которое проводится с целью определения, как быстро работает система или её часть под определённой нагрузкой.

В рамках тестирования производительности выделяют следующие подвиды:

a)нагрузочное (*performance & load testing*)

b)стрессовое (*stress testing*)

c)тестирование стабильности / надежности (*stability / reliability testing*)

d)тестирование объемами (*volume testing*)

2.Нефункциональные виды.

Тестирование производительности

Нагрузочное тестирование *(performance & load testing)* – это автоматизированное тестирование, которое имитирует одновременную работу множества пользователей над тестируемым приложением.

Проверяется способность приложения сохранять заданные показатели качества при нагрузке в допустимых пределах и некотором превышении этих пределов (определение «запаса прочности»).

2.Нефункциональные виды.

Тестирование производительности

Стрессовое тестирование *(stress testing)* позволяет проверить насколько приложение и система в целом работоспособны при нештатных изменениях нагрузки или в ситуациях недоступности значительной части необходимых ПП ресурсов, а также оценить способность системы к регенерации, т.е. к возвращению к нормальному состоянию после прекращения воздействия стресса.

Целью стрессового тестирования является оценка работоспособности системы в условиях повышенных и предельных нагрузок.

2.Нефункциональные виды.

Тестирование производительности

Тестирование стабильности / надежности *(stability / reliability testing)* - проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки.

2.Нефункциональные виды.

Тестирование производительности

Тестирование объемами *(volume testing)* - получение оценки производительности при увеличении объемов данных в базе данных приложения, при этом происходит:

- измерение времени выполнения выбранных операций при определенных интенсивностях выполнения этих операций
- может производиться определение количества пользователей, одновременно работающих с приложением

2.Нефункциональные виды.

Установочное тестирование

- **Тестирование установки** ***(инсталляционное тестирование)*** - направлено на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.

В общем случае проверяет множество сценариев и аспектов работы инсталлятора в таких ситуациях, как:

- ✓ новая среда исполнения, в которой приложение ранее не было инсталлировано;
- ✓ обновление существующей версии («апгрейд»);
- ✓ изменение текущей версии на более старую («даунгрейд»);
- ✓ повторная установка приложения с целью устранения возникших проблем («переинсталляция»);
- ✓ повторный запуск инсталляции после ошибки, приведшей к невозможности продолжения инсталляции;
- ✓ удаление приложения;
- ✓ установка нового приложения из семейства приложений;
- ✓ автоматическая инсталляция без участия пользователя.

2.Нефункциональные виды.

Тестирование удобства пользования

Вид тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.

Исследуется, насколько конечному пользователю нравится использовать данный продукт.

Очень часто успех продукта зависит именно от эмоций, которые он вызывает у пользователей.

2.Нефункциональные виды. Тестирование на отказ и восстановление

Тестирование отказоустойчивости *(failover testing)* — тестирование, заключающееся в эмуляции или реальном создании критических ситуаций с целью проверки способности приложения задействовать соответствующие механизмы, предотвращающие нарушение работоспособности, производительности и повреждения данных.

Тестирование восстанавливаемости *(recoverability testing)* — тестирование способности приложения восстанавливать свои функции и заданный уровень производительности, а также восстанавливать данные в случае возникновения критической ситуации, приводящей к временной (частичной) утрате работоспособности приложения.

2.Нефункциональные виды.

Конфигурационное тестирование

Специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы (заявленных платформах, операционных системах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.)

2.Нефункциональные виды.

Тестирование интернационализации

Проверяет готовность приложения к работе с различными языковыми интерфейсами. Этот вид тестирования не подразумевает проверки качества соответствующей адаптации (этим занимается *тестирование локализации*), оно сфокусировано именно на проверке возможности такой адаптации.

В частности, проверяется способность корректно отображать шрифты, пункты меню, производить поиск, сортировку, способность приложения обрабатывать файлы, поименованные на различных языках.

Следующей стадией, как правило, является *локализационное тестирование*.

2.Нефункциональные виды.

Тестирование локализации

Проверяет, насколько корректно продукт адаптирован к работе на том или ином языке: всё ли переведено и переведено правильно, не нарушилась ли логика построения интерфейса и обработки данных и т.д. Это тестирование следует за *тестированием интернационализации* и проверяет корректность перевода и адаптации продукта, а не готовность продукта к таким действиям.

Для локализационного тестирования рекомендуется обязательно приглашать в команду носителя того языка, перевод на который тестируется.

2.Нефункциональные виды. Тестирование интернационализации и локализации

В ходе тестирования локализации и интернационализации проверяются следующие аспекты:

- *Соответствие приложения стандартам оформления пользовательского интерфейса*
- *Отсутствие непереведенных сообщений*
- *Проверка правильности перевода системных сообщений и ошибок.*
- *Проверка правильности перевода согласно тематике*
- *Совместимость приложения со стандартами различных регионов*
- *Проверка правильности оформления документации и вспомогательных файлов*

2.Нефункциональные виды.

Тестирование документации

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 сложноустранимые и дорогостоящие последствия.

2.Нефункциональные виды.

Тестирование совместимости

Основной целью является проверка корректной работы продукта в определенном окружении.

Окружение может включать в себя следующие элементы:

- ✓Совместимость с аппаратной платформой, операционной системой и сетевой инфраструктурой;
- ✓Совместимость с браузерами и их версиями;
- ✓Совместимость с мобильными устройствами;
- ✓И так далее (например: Базы данных).

3. Связанные с изменениями виды тестирования

После проведения необходимых изменений, таких как исправление бага/дефекта, программное обеспечение должно быть перетестировано для подтверждения того факта, что проблема была действительно решена.

3. Связанные с изменениями виды тестирования

Виды тестирования, которые необходимо проводить после установки программного обеспечения, для подтверждения работоспособности приложения или правильности осуществленного исправления дефекта:

- **Дымовое тестирование** *(Smoke Testing)*
- **Регрессионное тестирование** *(Regression Testing)*
- **Тестирование сборки** *(Build Verification Test)*
- **Санитарное тестирование или проверка согласованности/исправности** *(Sanity Testing)*

3. Связанные с изменениями виды тестирования

Дымовое тестирование

Дымовое тестирование рассматривается как короткий цикл тестов, выполняемый для подтверждения того, что после сборки кода (нового или исправленного) устанавливаемое приложение, стартует и выполняет основные функции.

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

3. Связанные с изменениями виды тестирования

Регрессионное тестирование

Это вид тестирования, осуществляемый для подтверждения факта, что существующая ранее функциональность работает как и прежде после сделанных в приложении или окружающей среде исправлений или дополнений (устранение дефекта, слияние кода, миграция на другую операционную систему, базу данных, веб-сервер или сервер приложения).

3. Связанные с изменениями виды тестирования

Тестирование сборки

Тестирование 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.

3. Связанные с изменениями виды тестирования

Санитарное тестирование

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.

Виды тестирования

Классификация по степени автоматизации

- **Ручное тестирование** *(manual testing)* - тест-кейсы выполняет человек
- **Автоматизированное тестирование** *(automated testing)* - тест-кейсы полностью выполняет специальное инструментальное средство.
- **Полуавтоматизированное тестирование** *(semiautomated testing)* - тест-кейсы частично выполняет специальное инструментальное средство.

Виды тестирования

Классификация по признаку позитивности сценариев

- **Позитивное тестирование** *(positive testing)* - направлено на исследование приложения в ситуации, когда все действия выполняются строго по инструкции без каких бы то ни было ошибок, отклонений, ввода неверных данных и т.д.
- **Негативное тестирование** *(negative testing)* - направлено на исследование работы приложения в ситуациях, когда с ним выполняются (некорректные) операции и/или используются данные, потенциально приводящие к ошибкам (деление на ноль)

Классификация по времени проведения тестирования

Тестирование новой функциональности

Проверка того, что заявленный в данном билде новый функционал работает должным образом.

"Все обещанное на месте и работает".

Тесты по подготовленности

Свободное тестирование (Интуитивное тестирование, Спонтанное тестирование) (*Ad hoc testing*) – это

Тестирование, выполняемое неформально, без формальной подготовки тестов, формальных методов проектирования тестов, определения ожидаемых результатов и руководства по выполнению тестирования.

Исследовательское тестирование (*Exploratory testing*) – это

Неформальный метод проектирования тестов, при котором тестировщик активно контролирует проектирование тестов в то время, как эти тесты выполняются, и использует полученную во время тестирования информацию для проектирования новых и улучшенных тестов

Контрольные вопросы:

1. Описать направления тестирования.
2. Сравнить методы тестирования: черный, белый и серый ящик.
3. Описать уровни тестирования «вглубь» и «вширь».
4. Цели функционального тестирования.
5. Перечислить нефункциональные виды тестирования.
6. Виды тестирования, связанные с изменениями.
7. Классификация тестирования по степени автоматизации и позитивности сценариев.
8. Описать альфа- , бета- и гамма-тестирование.