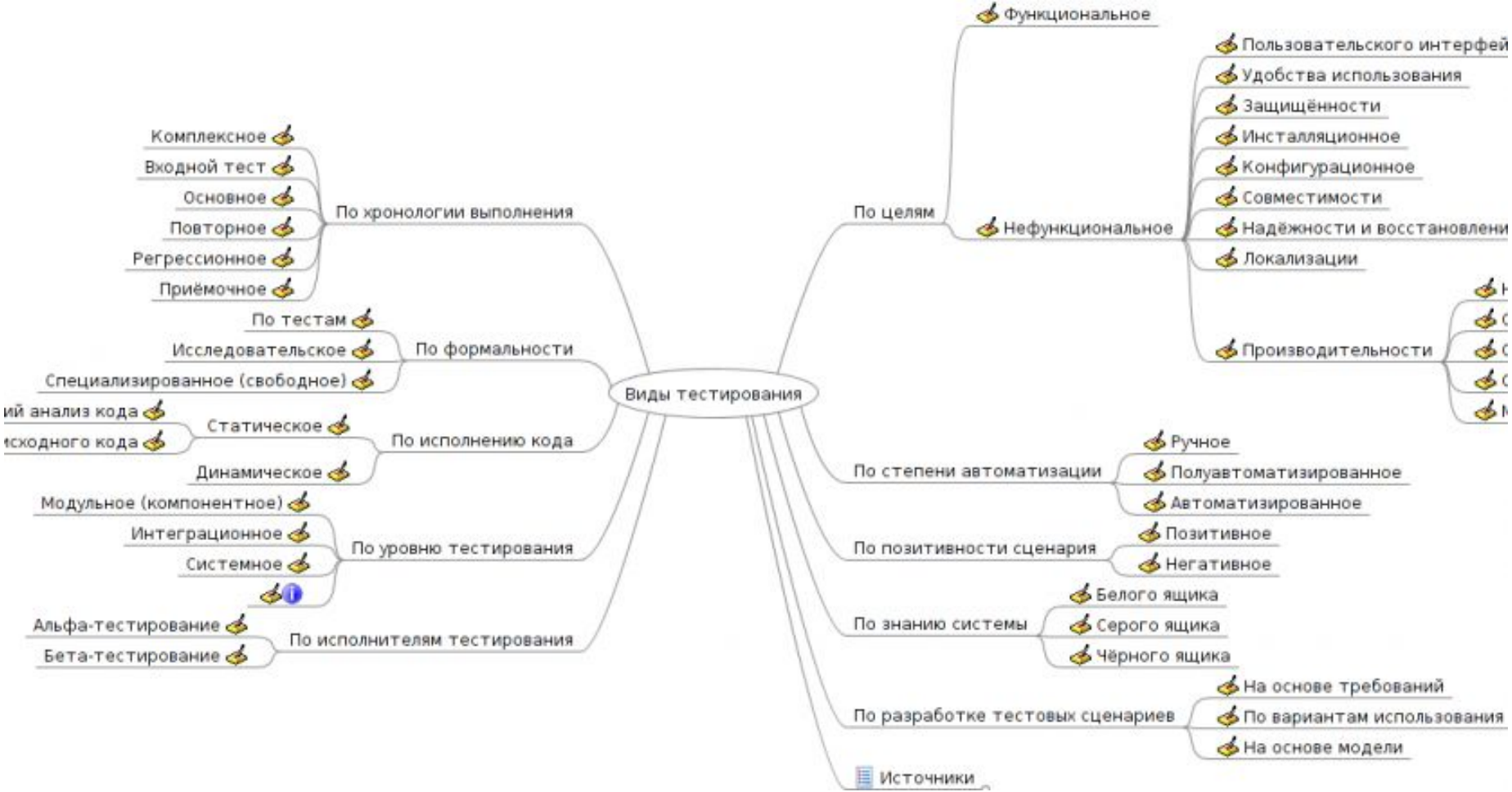
Виды тестирования


Виды тестирования
По целям
Функциональное
Нефункциональное
Пользовательского интерфей
Удобства использования
Защищённости
Инсталляционное
Конфигурационное
Совместимости
Надёжности и восстановлени
Локализации
Производительности
По степени автоматизации
Ручное
Полуавтоматизированное
Автоматизированное
По позитивности сценария
Позитивное
Негативное
По знанию системы
Белого ящика
Серого ящика
Чёрного ящика
По разработке тестовых сценариев
На основе требований
По вариантам использования
На основе модели
Источники
По хронологии выполнения
Комплексное
Входной тест
Основное
Повторное
Регрессионное
Приёмочное
По формальности
По тестам
Исследовательское
Специализированное (свободное)
По исполнению кода
Статическое
Динамический анализ кода
Анализ исходного кода
Динамическое
По уровню тестирования
Модульное (компонентное)
Интеграционное
Системное
По исполнителям тестирования
Альфа-тестирование
Бета-тестирование

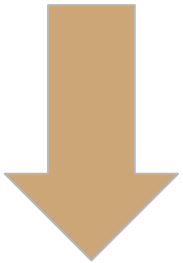
Функциональное тестирование

Функциональное тестирование выполняется чтобы убедиться, что каждая функция программного приложения ведет себя так, как указано в документе с требованиями.

Задача:

проверяем ЧТО наша система делает

Функциональное тестирование



функциональное

безопасности

Если продукт отвечает за безопасность, тогда это функциональное.

Этапы функционального тестирования:

1. Определении функциональности продукта, который необходимо протестировать. Включает в себя тестирование основных функций, условий ошибок и сообщений, тестирование удобства использования, то есть, является ли продукт удобным для пользователя или нет, и т. д.

2. Создание входных данных для проверяемой функциональности в соответствии со спецификацией требований.

3. Позже, из спецификации требований, определяется результат для тестируемой функциональности.

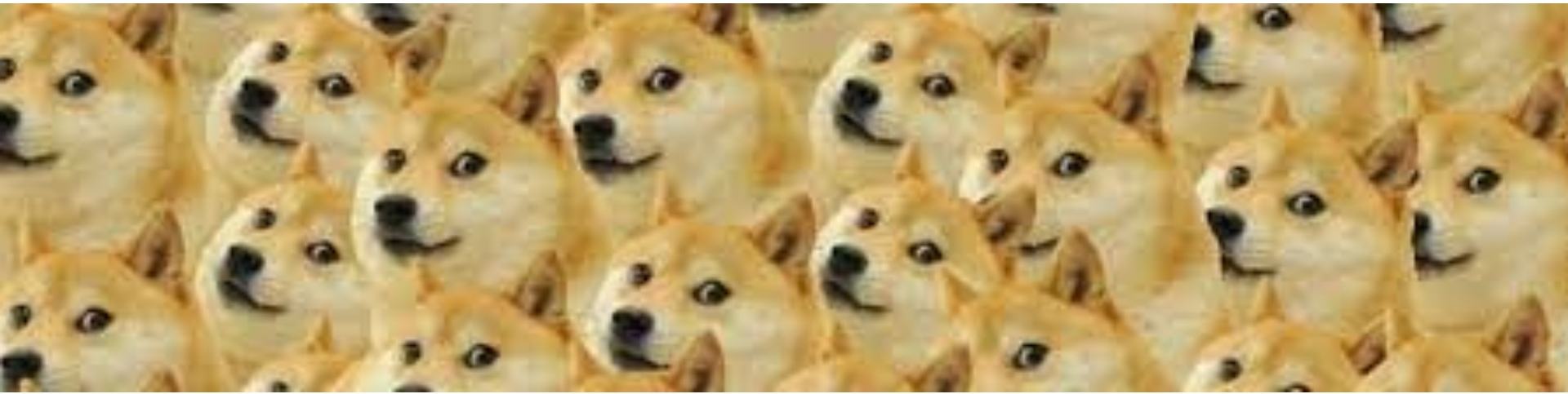
4. Подготовленные тест-кейсы исполняются.

5. Фактический результат, то есть результат после выполнения тест-кейса, и ожидаемый результат (определенный из спецификации требований) сравниваются, чтобы определить, работает ли функциональность должным образом или нет.

Нефункциональное тестирование

Нефункциональное тестирование проводится для проверки нефункциональных требований приложения, таких как производительность, безопасность, совместимость, надежность, удобство использования и т. д.

Виды нефункционального тестирования:



1. Тестирование безопасности

- только достоверный пользователь может войти в систему (Аутентификация)
- пользователь должен иметь возможность входить в те модули, для которых он авторизован или к которым пользователю был предоставлен доступ (Авторизация)
- пароль должен соответствовать требованиям (то есть длине, специальным символам, числам и т. д.)
- тайм-аут: если приложение неактивно, оно должно истечь по таймауту в указанное время;
- резервное копирование данных: резервное копирование данных должно быть выполнено в указанное время и данные должны быть скопированы в безопасное место;
- вся коммуникация должна быть зашифрована

2. Тестирование интерфейса (UI)

UI (user interface) — пользовательский интерфейс: все кнопки, таблички, поля ввода текста и другие способы взаимодействия юзера с сайтом, приложением или иным IT-сервисом.

Например:

• соблюдение единого стиля

• интерфейс соответствует дизайну (шрифт, цвет, размер, расположение на экране)

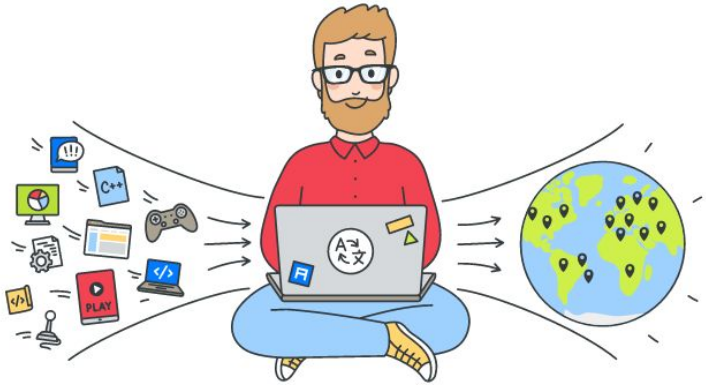
UI UX

3. Тестирование удобства использования (Usability)

Usability-тестирование предназначено для определения того, насколько программный продукт понятен, легок в освоении, прост в эксплуатации и привлекателен для пользователей при определенных условиях и требованиях.

- Как быстро пользователь достиг цели?
- Удобно ли попасть по кнопке?
- Не длинный ли путь?
- Порог вхождения (как быстро пользователь поймет как пользоваться программой)

4. Тестирование локализации



•перевод контента (аудио, видео, документация)
•учет региональных особенностей
•функциональные изменения (система мер, формат даты, часовой пояс, начало недели)
•перевод текста

В данный вид проверки входит необходимость выполнения работ по переводу всего контента ПО для конечного пользователя. Во время перевода должны учитываться иконки, информационная графика, справочные материалы, техническая документация и иные культурные особенности регионов.

5. Тестирование интернационализации

Тестирование продукта на одинаковую работу в разных регионах и культурах мира.

Задача:

может ли программный код работать со всей международной поддержкой без нарушения функциональности, что может привести к потере данных или проблемам целостности информации.

- проверка языковой совместимости: может ли продукт правильно работать в определенной языковой среде?
- правильно ли отображаются и принимаются на ввод валюта, дата, время, индекс и т.п.
- интерфейс: любые визуальные проблемы, такие как проблемы с графикой, наложение текста

6. Тестирование совместимости

Цель:

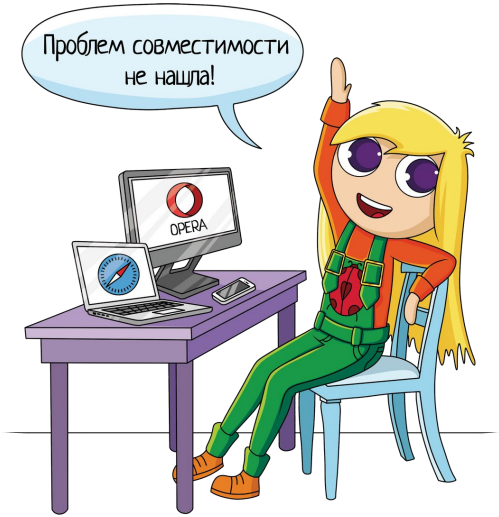
проверка корректной работы продукта в определенном окружении.

1. Аппаратная платформа;

2. Операционная система (Windows, MacOS, …)

3. Браузеры (Internet Explorer, Firefox, Opera, Chrome, Safari)

4. Различные расширения экрана



Кроссбраузерное тестирование – это вид тестирования, направленный на поддержку и корректное отображение продукта в различных браузерах, мобильных устройствах, планшетах и экранах разных размеров.

7. Конфигурационное тестирование

Данный вид тестирования применяется, если известно, что продукт будет использоваться на разных платформах, в различных браузерах, будет поддерживать разные версии драйверов и т.п.

Цель:

• Определить оптимальную конфигурацию оборудования, обеспечивающую требуемые характеристики производительности и времени реакции тестируемой системы.

Как отличить совместимость от конфигурации?

При конфигурационном тестировании тестируют различные конфигурации железа, софта, которые поддерживает система. Например, сайт по спецификации должен работать на веб-браузерах Google Chrome, Mozilla Firefox и Apple Safari. Или приложение должно запускаться с 500 мб оперативной памяти.

Тестирование совместимости проверяет совместимость с объектами вне зависимости от того поддерживает ли их система.

Пример. Приложение разработано для работы с семейством Windows. При тестировании его на различных видах Windows (XP, 7, 10, ...), это будет конфигурационное тестирование. Но когда мы начнем тестировать его на Linux, то это уже будет тестирование совместимости.

8. Инсталляционное тестирование

Успешная установка, настройка, обновление и удаление ПО (как десктопного, так и мобильного)

Инсталляционное тестирование

Установка:

- должна начаться после клика по кнопке
- успешна во всех поддерживаемых окружениях
- подсчитывает ли свободное место перед установкой и выдает ли предупреждение если недостаточно
- наличие созданных ярлыков

Обновление:

- поддерживает ли приложение функцию обновления
- поведение с более ранней версией
- сохраняются ли настройки пользователя
- запуск приложения после обновления
- откат к предыдущей версии

Удаление:

- не остается ли в системе никаких папок/файлов/ярлыков после полного удаления приложения
- корректно ли работает система после установки и последующего удаления приложения

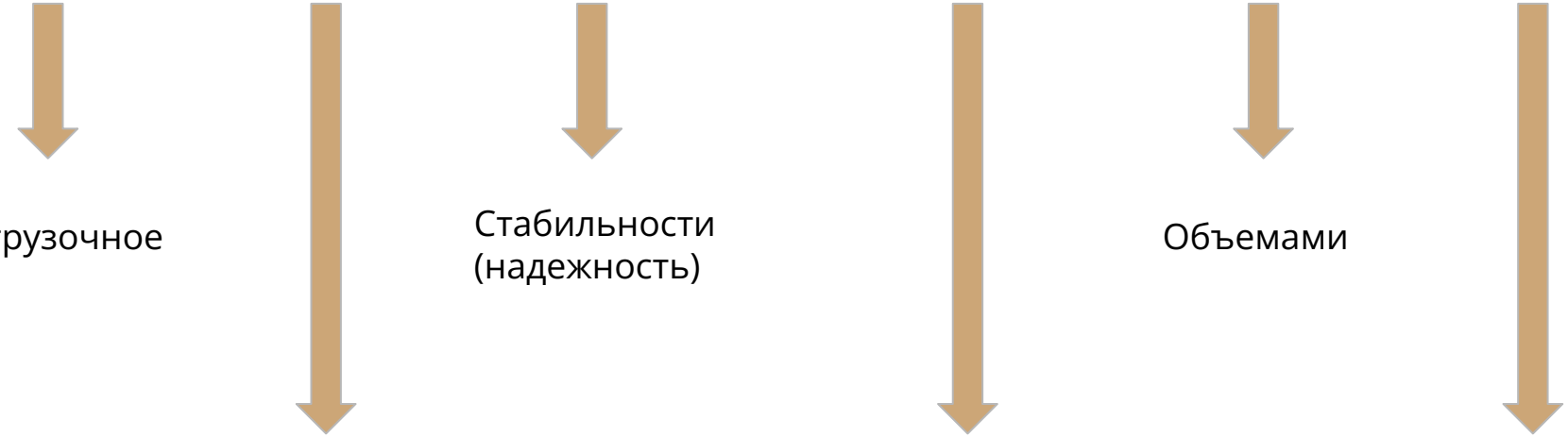
9. Тестирование производительности

Используется для проверки скорости, времени отклика, стабильности, надежности, масштабируемости и использования ресурсов приложения при определенной рабочей нагрузке, обычно регрессионным образом, когда в приложение ежедневно или еженедельно вносятся небольшие изменения.

Цель:

выявить и устранить узкие места производительности в программном приложении.

Тестирование производительности



Нагрузочное

Стрессовое

Стабильности (надежность)

Масштабируемости

Объемами

Конкурентное

9.1. Нагрузочное

Это проверка того, как работает система под разными уровнями нагрузки до предельного значения, которое она должна выдерживать (предельное значение должно быть прописано в нефункциональных требованиях по нагрузке)

В процессе замеряется:

- время отклика системы

- скорость обработки запросов от пользователей (например, как быстро открываются и прогружаются страницы сайта, как быстро система выполняет расчеты, выдает результаты поиска и т.д.)

- сколько ресурсов “съедает” система – сетевых, процессорных, памяти.

Цель:

создав определенную нагрузку наблюдать за показателями производительности системы.

9.2. Стрессовое

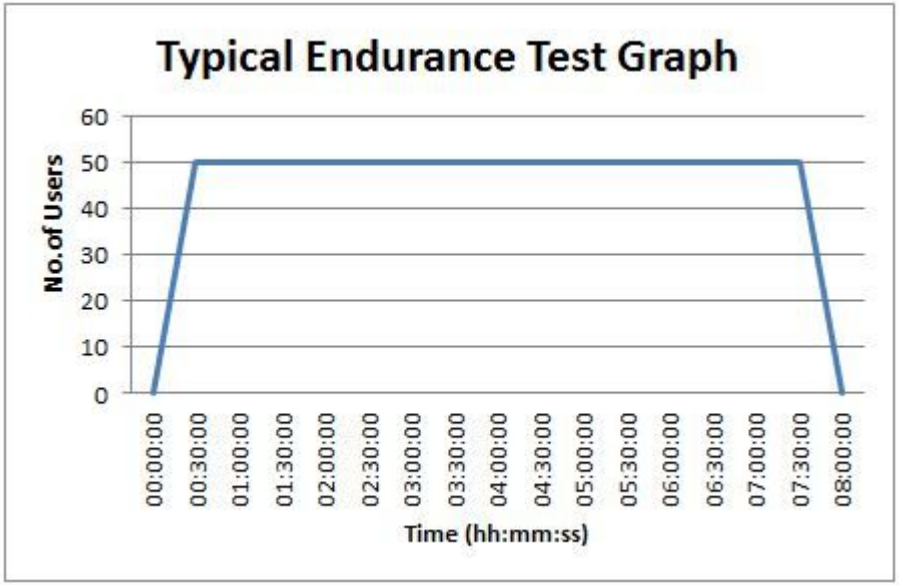
Оценивает скорость работы системы, если нагрузка на систему выше нормы, описанной в требованиях, или же в состоянии ограниченных ресурсов (память, скорость интернета).

Как отличить стресс от нагрузки?

Нагрузочное – это тестирование ***в пределах значений нагрузки***, которые должна выдерживать система, а стрессовое – это тестирования ***за ее пределами***.

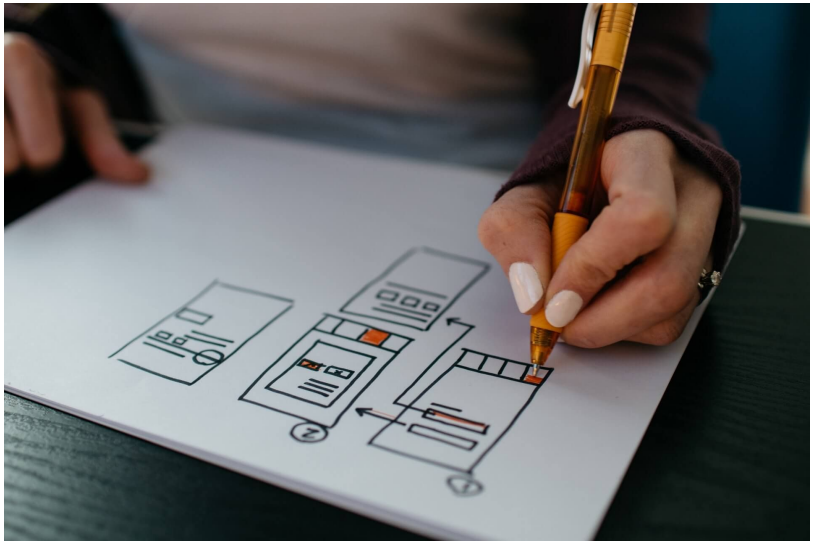
9.3. Стабильности (надежности)

Это тестирование системы со значительной нагрузкой в течение длительного периода времени, чтобы выяснить, как система ведет себя при длительном использовании. То есть для обеспечения того, чтобы производительность и / или время отклика после некоторого длительного периода устойчивой активности были не хуже, чем в начале теста.



9.4. Масштабируемости

Проводится для определения способности приложения масштабироваться с точки зрения пользовательской нагрузки, количества транзакций, объема данных и т. д



9.5. Объемное

Предназначено для прогнозирования того, может ли система / приложение обрабатывать большой объем данных в плане проверки объема данных, обрабатываемых базой данных.

9.6. Конкурентное

Проверяется поведение системы в момент, когда одновременно происходят два или более событий, или выполняется одновременный вход нескольких пользователей в систему с выполнением одного и того же действия.

Цели:

Определить влияние одновременного доступа к одним и тем же записям базы данных, модулям или коду приложения;

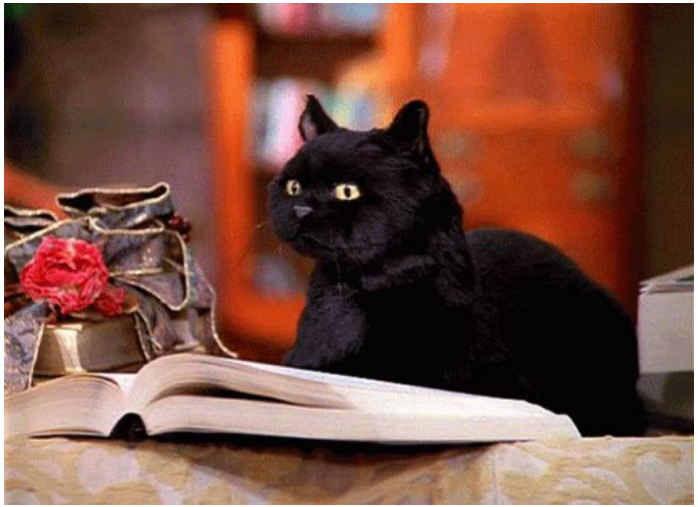
Определить и измерить уровень взаимоблокировки, блокировки и использования однопоточного кода и ограничения доступа к общим ресурсам;

10. На отказ и восстановление

Проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками ПО, отказами оборудования или проблемами связи/сети.

11. Тестирование документации -

это процесс проверки качества документации, который может начаться на ранних этапах жизненного цикла разработки. Чем раньше начнется тестирование, тем раньше будет выявлен баг, и тем дешевле будет его починить.



Включает в себя такие документы, как ***инструкции для пользователя и администратора, замечания по релизу, тестовые техники, список требований***, и т. д.

Цель:

помогает повысить качество продуктов и связано с любой информацией