По знанию системы:

Метод тестирования черного ящика – когда мы тестируем, но не знаем код, белый – когда имеем, серый – совмещенный режим.

По уровню тестирования:

Требования к ПО – Системный тест

Архитектура – Интеграционный тест

Спецификация модуля – Модульный тест

Системное тестирование:

* Проверка соответствия установленным требованиям
* Выявление дефектов, связанных с работой системы в целом:
* Отсутствующая функциональность
* Неверное использование ресурсов системы
* Непредусмотренные комбинации данных пользовательского уровня
* Несовместимость с окружением
* Непредусмотренные сценарии использования
* Неудобство в применении и т.п.
* Метод «Черного ящика»

Интеграционное тестирование:

* Проверка связи между компонентами, а также взаимодействия с различными частями системы (ОС, оборудование и т.д.)
* Поиск дефектов, связанных с ошибками в реализации и интерпретации взаимодействия между модулями
* Требует создания тестового окружения
* Метод «Белого ящика»

Модульное тестирование:

* Проверка отдельно взятых модулей, функций или классов
* Выявление локализованных в модуле ошибок в алгоритмах или реализации алгоритмов
* Требует создания тестового окружения
* Метод «Белого ящика»

По методам сборки модулей:

Монолитный

Инкрементальный (два вида: сверху-вниз и снизу-вверх)

Сравнение методов

Монолитное:

- Большие трудозатраты, связанные со сложностью идентификации ошибок

+ Большие возможности распараллеливания работ

Инкрементальное:

- Требует затрат на разработку заглушек и драйверов

+ Связано с меньшей трудоемкостью идентификации ошибок за счет локализации тестируемого кода

Недостатки Сверху-вниз

* Проблема разработки достаточно интеллектуальных заглушек, т.е. заглушек, способных к использованию различных режимов работы
* Сложность организации и работы среды для реализации исполнения модулей в нужной последовательности
* Параллельная разработка модулей верхним и нижних модулей не всегда приводит к эффективной реализации модулей из-за подстройки нижних уровней к уже протестированным верхним

Недостатки Снизу-вверх

* Запаздывание проверки концептуальных особенностей тестируемого комплекса

Комбинирование уровней тестирования

* В каждом конкретном проекте должны быть определены задачи, ресурсы и технологии для каждого уровня тестирования
* Задача тестировщика и менеджеров – оптимально распределить ресурсы между тремя уровнями тестирования так, чтобы каждый из возможных типов дефектов был «адресован»
* Например, перенос усилий на поиск фиксированного типа дефектов из области системного в область модульного может сэкономить время и деньги тестирования

По степени автоматизации:

Ручное

Автоматизированное

Автоматическое

По целям:

Функциональное

Не функциональное: производительности, инсталляционное, конфигурационное, безопасности, локализации, удобства.

Функциональное тестирование – рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом. Может приводится в аспекте требований и бизнес процессов. Может приводится на всех уровнях тестирования.

План:

Ящики

(По уровню) Системный – интеграционный – модульный

(По методам сборки модулей) Монолитный – инкрементальный

Совмещение уровней тестирования

Степень автоматизации

По целям

Тестирование производительности

Это автоматизированное тестирование, имитирующее работу определенного количества бизнес пользователей на каком-либо общем ресурсе.

Виды:

* Производительности и стабильности – с какой скоростью работает приложение с каким-то количеством пользователей
* Нагрузочное – сможет ли система работать при неожиданном потоке пользователей
* Объемное – сможет ли система работать так же, как сейчас, если будет расширяться
* Стрессовое – какое количество пользователей необходимо до отказа
* Инсталляционное – проверка успешной установки и настройки, удаления обновления ПО при различном программном и аппаратном окружении, призванное также оценить работоспособность системы после завершения работы инсталлятора.
* Проверяет:
* Установку
* Корректность файлов
* Регистрацию приложения в ОС
* Обратную совместимость данных
* Прерывание установки, установка при запущенном приложении и т.д.
* Удаление
* Конфигурации – направлено на проверку работы ПО при различных конфигурациях системы
* Заявленных платформах
* Поддерживаемых драйверов
* При различных конфигурациях ПК
* При различных конфигурациях системного ПО и т.д.
* Обычно автоматизированное
* Безопасности – используется 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
* Локализации – проверка правильности перевода элементов интерфейса пользователя, проверка правильности перевода системных сообщений и ошибок, проверка перевода раздела Помощь/справка и сопроводительной документации
* Встроен ли текст в программный код?
* Перевод длиннее исходного текста
* Символы, входящие в ANSI
* Клавиатура («мертвые клавиши»)
* Фильтрация ввода
* Клавиши вызова
* Сборные сообщения
* Идентификатор сообщения об ошибке
* Формат данных, единицы измерений
* Сортировка, правила перевода регистра
* Удобства использования – направлено на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.
* Способы проведения:
* Экспертная оценка
* Анкеты, пользовательское тестирование
* Математические оценки
* Чит-листы, гайдлайны и стандарты

По хронологии выполнения:

* Дымовое – короткий цикл тестов, выполняемый для подтверждения того, что после сборки кода устанавливаемое приложение стартует и выполняет основные функции, обычно автоматическое
* Основное – круг выполнения всех запланированных проверок для новой и или измененной разработки. В зависимости от целей может совмещать в себе функциональное тестирование, тестирование пользовательского интерфейса и удобства использования. Также, может быть тестированием производительности или безопасности.
* Регрессионное – Направлено на подтверждение того факта, что существующая ранее функциональность работает, как и прежде после изменений, сделанных в приложении или окружающей среде. Чаще автоматическое.
* Приемочное – формальный процесс тестирования, который проверяет соответствие системы потребностям, требованиям и бизнес процессам пользователя. Проводится для вынесения заказчиком или другим уполномоченным лицом, принимается приложение или нет.

Критерий покрытия тестирования:

* Метрика для оценки качества тестирования
* Измеряет долю классов ситуаций, представители которых попали в тестовый набор
* Основан на коде: критерий белого ящика, но не выявит нереализованные требования. Основан на спецификации: критерий черного ящика, но может оставить непроверенными некоторыми участки кода
* Критерии черного ящика:
* Тестирование функций
* Тестирование классов входных данных
* Тестирование классов выходных данных
* Тестирование области допустимых значений
* Тестирование длины набора данных
* Тестирование упорядоченности набора данных
* Критерии белого ящика:
* Покрытия операторов
* Покрытия ветвей
* Покрытия путей
* Покрытия решений и условий
* Комбинаторного покрытия условий

Требования – условие или возможность требуемая пользователем для решения задач или достижения целей

Требования – условие или возможность которые должны удовлетворяться системой или компонентом системы, или которыми система или компонент должна обладать для обеспечения условий контракта, стандартов, спецификаций или др. регулирующими документами

Требования – документальная репрезентация условий или возможностей, перечисленных в предыдущих пунктах

Источники требований

* Стандарты
* Федеральное законодательство
* Нормативное обеспечение организации
* Представления и ожидания потребителей и пользователей системы
* Конкурирующие программные продукты

Методы выявления требований:

* Общение с заказчиком
* Общение с авторами продукта
* Анализ стандартов и нормативной документации
* Анализ выявленных проблем в баг-трекере

Виды требований:

* Бизнес-требования
* Требования пользователей
* Функциональные требования
* Нефункциональные требования
* Предположения и ограничения
* Требования, связанные с внедрением

Проблемы требований:

* Неполные
* Частные изменения
* Неверно трактовали
* Изменения в последний момент

Показатели качества требований:

* Корректность или адекватность
* Целостность
* Недвусмысленность
* Полнота
* Непротиворечивость
* Упорядоченность по приоритету и стабильности
* Проверяемость
* Модифицируемость
* Прослеживаемость в ходе разработке

Тестирование требований:

* Когда? Как только появилось хотя бы одно требование
* Зачем? Уменьшение количества доработок и изменений, сокращение рисков, ознакомление и согласование задач между разработчиками
* Доколе? Достаточно информации для начала разработки/тестирования
* Кто? Аналитики, тестировщики, менеджер, разработчики

Тест №1. Проверяем требования на полноту

* Содержат ли требования выражения типа «подлежит определению», «и так далее», «и прочее», «но»…
* Ссылаются ли требования на несуществующие источники?
* Ссылается ли на ещё не определенные источники?

Тест №2. Выявить неполные, неизмеримые требования

* Определяем меру качества для каждого требования
* Верно ли, что каждое требование имеет критерий качества, который можно использовать для проверки того, удовлетворяет ли какое-либо решение требованию?
* Обращаем внимание на неоднозначные требования (быстро, нормально, оптимально)

Тест №3. Проверяем отслеживаемость требований

* Рассматриваем каждое требование как отдельно распознаваемую, измеряемую сущность: каждое ли требование однозначно распознаваемо?

Тест №4. Однозначно ли требование?

* Всякая ли ссылка на термин, определенный в спецификации требований, согласуется с этим определением?
* Используется ли один и тот же термин для одних и тех же понятий?