**Классификация тестирования по уровням.**

Классификация тестирования программного обеспечения по уровням делится на следующие категории:

**Модульное тестирование (Unit Testing):**

Проверка отдельных модулей или компонентов системы.

Проводится разработчиками.

**Интеграционное тестирование (Integration Testing):**

Проверка взаимодействия между модулями.

Выявление проблем взаимодействия.

**Системное тестирование (System Testing):**

Полная проверка всей системы в целом.

Проводится командами качества.

**Приемочное тестирование (Acceptance Testing):**

Проверка системы на соответствие требованиям клиента.

Может проводиться непосредственно пользователями.

\_\_\_\_\_

**1. Модульное тестирование (Unit Testing):**

Цель: Проверка корректности работы отдельных модулей, компонентов или функций кода.

Кто выполняет: Обычно разработчики.

Объект тестирования: Отдельные функции, методы, классы или небольшие модули.

Методы: Как правило, автоматизированное тестирование с использованием фреймворков для модульного тестирования (например, JUnit для Java, pytest для Python, NUnit для .NET).

Фокус: Изолированная проверка логики, входных и выходных данных, обработка исключений.

Преимущества:

Раннее обнаружение дефектов.

Упрощение отладки (баги локализованы в конкретных модулях).

Улучшение дизайна кода (требует модульного дизайна).

Облегчение рефакторинга.

Недостатки:

Не проверяет взаимодействие между модулями.

Требует хорошего понимания кода.

Может быть сложно писать тесты для некоторых модулей.

**2. Интеграционное тестирование (Integration Testing):**

Цель: Проверка корректности взаимодействия между интегрированными модулями, компонентами или подсистемами.

Кто выполняет: Обычно команда тестирования или разработчики (в зависимости от организации).

Объект тестирования: Комбинации интегрированных модулей, API, сервисы.

Методы:

Сверху вниз (Top-Down): Начинается с тестирования верхнего уровня модулей и постепенно переходит к нижним. Требуются “заглушки” (stubs) для имитации поведения нижних уровней.

Снизу вверх (Bottom-Up): Начинается с тестирования нижнего уровня модулей и постепенно переходит к верхним. Требуются “драйверы” (drivers) для имитации поведения верхних уровней.

Большой взрыв (Big Bang): Все модули интегрируются одновременно и тестируются как единое целое. Сложно отлаживать.

Инкрементное тестирование: Модули интегрируются и тестируются постепенно, по одному или небольшими группами. Самый распространенный и рекомендуемый подход.

Фокус: Проверка передачи данных между модулями, обработка ошибок при взаимодействии, соответствие интерфейсов.

Преимущества:

Обнаружение дефектов, связанных с интеграцией.

Проверка соответствия интерфейсов.

Уменьшение риска серьезных ошибок на более поздних этапах.

Недостатки:

Более сложно отлаживать, чем модульное тестирование.

Требует более сложной подготовки тестового окружения.

**3. Системное тестирование (System Testing):**

Цель: Проверка соответствия всей системы требованиям и спецификациям.

Кто выполняет: Независимая команда тестирования (тестировщики).

Объект тестирования: Полностью интегрированная система.

Методы: Черный ящик (Black-Box testing) – тестирование без знания внутренней структуры системы. Тестирование проводится на основе требований и спецификаций.

Типы тестирования, часто выполняемые на этом уровне:

Функциональное тестирование (например, тестирование сценариев использования, тестирование граничных значений).

Нефункциональное тестирование (например, тестирование производительности, тестирование безопасности, тестирование юзабилити).

Фокус: Проверка сквозной функциональности, производительности, безопасности и других системных характеристик.

Преимущества:

Проверка соответствия системы требованиям заказчика.

Выявление серьезных дефектов, которые могут повлиять на работу системы.

Недостатки:

Может быть трудоемким и дорогостоящим.

Сложно выявить причину обнаруженных дефектов.

Требует полностью интегрированной системы.

**4. Приемочное тестирование (Acceptance Testing):**

Цель: Проверка готовности системы к эксплуатации и соответствия ожиданиям конечных пользователей или заказчиков.

Кто выполняет: Конечные пользователи, заказчики или представители заказчика.

Объект тестирования: Полностью интегрированная система, готовая к развертыванию.

Методы:

Тестирование, определяемое пользователем (User Acceptance Testing - UAT): Реальные пользователи тестируют систему в реальных условиях.

Тестирование по контракту (Contract Acceptance Testing): Проверка соответствия системы условиям контракта.

Альфа-тестирование: Тестирование пользователями или тестировщиками, но в контролируемой среде разработчика.

Бета-тестирование: Тестирование реальными пользователями в реальной среде.

Фокус: Проверка удобства использования, соответствия бизнес-процессам, соответствия требованиям заказчика.

Преимущества:

Гарантия соответствия системы ожиданиям заказчика.

Улучшение удовлетворенности пользователей.

Снижение риска проблем после развертывания.

Недостатки:

Зависит от доступности конечных пользователей.

Может быть сложно организовать и контролировать.

**4. Приемочное тестирование (Acceptance Testing):**

**4.1 Альфа-тестирование (Alpha Testing):**

Цель: Оценка продукта в контролируемой среде разработчика, часто внутренними пользователями или специально нанятыми тестировщиками. Выявление багов, проблем юзабилити и общей функциональности перед выпуском для более широкой аудитории.

Кто выполняет: Внутренние тестировщики, разработчики, сотрудники компании, либо специально нанятые тестировщики, работающие в офисе разработчика.

Среда: Контролируемая среда, часто в лаборатории или офисе разработчика.

Фокус: Поиск критических багов, проверка общей функциональности, юзабилити и соответствия требованиям.

Преимущества:

Раннее выявление проблем, которые могут повлиять на пользовательский опыт.

Возможность получить обратную связь от пользователей до публичного релиза.

Недостатки:

Может не отражать реальное использование продукта.

Требует значительных ресурсов на организацию и проведение.

**4.2 Бета-тестирование (Beta Testing):**

Цель: Оценка продукта реальными пользователями в реальной среде. Получение обратной связи о функциональности, юзабилити, производительности и надежности в условиях, максимально приближенных к реальным.

Кто выполняет: Конечные пользователи (бета-тестеры), которые используют продукт в своей обычной среде.

Среда: Реальная среда, где пользователи будут использовать продукт в повседневной жизни.

Фокус: Получение обратной связи о пользовательском опыте, выявление багов, которые не были обнаружены на предыдущих этапах, оценка производительности и надежности в реальных условиях.

Преимущества:

Получение ценной обратной связи от реальных пользователей.

Выявление проблем, которые трудно обнаружить в контролируемой среде.

Повышение удовлетворенности пользователей и снижение риска негативных отзывов после релиза.

Недостатки:

Сложно контролировать процесс бета-тестирования.

Может быть сложно обработать всю полученную обратную связь.

**4.3 Пользовательское приемочное тестирование** (UAT - User Acceptance Testing): (Описание как в предыдущем ответе, но теперь более конкретно как подвид приемочного тестирования)

**4.4 Операционное приемочное тестирование** (OAT - Operational Acceptance Testing): (Проверка готовности системы к операционной поддержке, включая резервное копирование, восстановление, мониторинг и т.д.)

**5 Выходное тестирование (Exit Testing/Release Testing/Production Readiness Testing):**

Цель: Подтверждение готовности программного обеспечения к выпуску в производственную среду. Это последний этап тестирования перед развертыванием, направленный на проверку, что все критические функции работают корректно, а риски развертывания минимальны. Некоторые считают это подмножеством системного или приемочного тестирования.

Кто выполняет: Команда тестирования, DevOps-инженеры, или команда Release Management.

Объект тестирования: Полностью развернутая система в тестовой среде, максимально приближенной к производственной. Проверяются процессы развертывания, конфигурации и интеграции.

Методы:

Повторное выполнение ключевых тестовых сценариев, которые были пройдены на предыдущих этапах.

Проверка процессов развертывания и отката изменений.

Тестирование интеграции с другими системами.

Оценка производительности и масштабируемости в условиях, приближенных к производственным.

Проверка безопасности и соответствия нормативным требованиям.

Фокус: Минимизация рисков при развертывании, проверка готовности системы к реальной нагрузке, подтверждение корректной работы всех критических функций. Убедиться, что система развертывается правильно и работает стабильно.

Преимущества:

Снижение риска проблем в производственной среде.

Повышение уверенности в качестве программного обеспечения.

Улучшение удовлетворенности пользователей.

Недостатки:

Требует хорошо настроенной тестовой среды, максимально приближенной к производственной.

Может быть сложно организовать и провести, особенно для сложных систем.

\_\_\_\_\_

**Разработать модульные тесты для программного модуля MathFunctions, содержащий методы вычисления косинуса и синуса. Код для программного модуля находится «Resorce/ModuleC#/». Оформить тест-кейсы для модульного тестирования.**