Тестирование программного обеспечения: основные понятия и принципы

Что такое тестирование программного обеспечения?

Тестирование ПО — это процесс проверки соответствия фактического поведения программы ожидаемым результатам, используя заранее подготовленные тестовые сценарии.

Основные цели тестирования:

Обнаружение ошибок до выпуска продукта для пользователей.

Предоставление информации о качестве продукта.

Подтверждение соответствия программного обеспечения заданным требованиям.

Принципы тестирования

Тестирование показывает наличие дефектов

Тестирование помогает выявить ошибки, но не может гарантировать их полное отсутствие. Оно лишь снижает вероятность их наличия.

Полное тестирование невозможно

Из-за огромного количества возможных комбинаций входных данных проверить все сценарии невозможно. Тестирование всегда выборочно.

Баги и их жизненный цикл

Что такое баг?

Баг — это отклонение фактического результата работы программы от ожидаемого.

Жизненный цикл бага:

Новый — баг обнаружен и зарегистрирован.

Назначен — передан разработчику на исправление.

Исправлен — разработчик устранил ошибку.

Проверка — тестировщик подтверждает исправление.

Закрыт — баг устранён.

Переоткрыт — если ошибка осталась.

Отклонён/Отложен — если баг неактуален или его исправление отложено.

Разница между UX и UI дизайном

UX (User Experience) — фокусируется на удобстве использования, логике взаимодействия и достижении целей пользователя.

UI (User Interface) — отвечает за визуальную часть: цвета, шрифты, кнопки и расположение элементов.

Пример:

UX-дизайнер проектирует путь пользователя при оформлении заказа.

UI-дизайнер определяет, как будет выглядеть кнопка "Купить".

Жизненный цикл разработки ПО (SDLC)

Анализ требований — определение целей и задач.

Планирование — оценка ресурсов и сроков.

Проектирование — создание архитектуры.

Разработка — написание кода.

Тестирование — проверка качества.

Развёртывание — выпуск продукта.

Модели разработки ПО

Водопадная модель:

Этапы выполняются последовательно.

Подходит для проектов с чёткими требованиями.

Не позволяет вернуться к предыдущему этапу.

V-образная модель:

Тестирование планируется параллельно с разработкой.

Акцент на проверку соответствия требованиям.

Используется в критичных системах (например, медицинских).

Сравнение:

Критерий Водопадная модель V-образная модель

Гибкость Низкая Средняя

Тестирование В конце проекта Параллельно

Риски Высокие Управляемые

Agile: основные ценности

Agile — гибкий подход к разработке, основанный на итерациях и сотрудничестве с заказчиком.

Ценности Agile:

Люди и взаимодействие важнее процессов.

Рабочий продукт важнее документации.

Сотрудничество с заказчиком важнее формальных договорённостей.

Готовность к изменениям важнее следования плану.

Разница между Scrum и Kanban

Scrum:

Работа делится на спринты (1-4 недели).

Фиксированные роли (Scrum-мастер, Product Owner).

Цель — завершить задачи спринта.

Kanban:

Непрерывный поток задач.

Визуализация на доске (To Do, In Progress, Done).

Цель — завершить конкретную задачу.

Разница:

В Scrum нельзя менять задачи во время спринта, в Kanban — можно.

Scrum подходит для проектов с чёткими целями, Kanban — для гибких процессов.

User Story: описание и оформление

User Story — описание функциональности с точки зрения пользователя.

Формат:

«Как [роль], я хочу [действие], чтобы [цель]».

Пример:

«Как клиент, я хочу фильтровать товары по цене, чтобы быстро найти нужный вариант».

Критерии качества (INVEST):

Независимая, Обсуждаемая, Полезная, Оцениваемая, Небольшая, Тестируемая.

Жизненный цикл тестирования (STLC)

Анализ требований — определение тестируемых функций.

Планирование — стратегия и ресурсы.

Разработка тест-кейсов — создание сценариев.

Выполнение тестов — запуск и фиксация багов.

Отчётность — анализ результатов.

Виды тестирования

Функциональное — проверка работы функций (например, авторизация).

Нефункциональное — проверка производительности, безопасности и удобства.

Статическое — анализ кода без запуска (например, code review).

Динамическое — тестирование с запуском программы.

Регрессионное — проверка после изменений.

Примеры:

Дымовое тестирование — проверка основных функций.

Нагрузочное тестирование — оценка работы под нагрузкой.

Разница между статическим и динамическим тестированием

Статическое:

Проверка кода и документации без запуска программы.

Часть верификации.

Динамическое:

Тестирование работающего ПО.

Часть валидации.

Валидация и верификация

Верификация:

«Мы делаем продукт правильно?» (соответствие требованиям).

Валидация:

«Мы делаем правильный продукт?» (удовлетворение потребностей пользователя).

Пример:

Верификация: кнопка соответствует макету.

Валидация: пользователь успешно отправляет форму.

Требования к ПО

Типы требований:

Бизнес-требования (цели).

Пользовательские (задачи пользователя).

Функциональные (поведение системы).

Свойства хороших требований:

Полнота, однозначность, непротиворечивость, тестируемость.

Методы тестирования

Чёрный ящик — тестирование без доступа к коду (через интерфейс).

Белый ящик — тестирование с доступом к коду.

Серый ящик — комбинация чёрного и белого ящика.

Позитивное и негативное тестирование

Позитивное — проверка корректных сценариев (например, ввод валидного пароля).

Негативное — проверка некорректных сценариев (например, ввод неверного пароля).

Разница между фронтенд и бэкенд тестированием

Фронтенд — тестирование интерфейса (кнопки, формы).

Бэкенд — тестирование логики, API, базы данных.

Функциональное и нефункциональное тестирование

Функциональное — проверка функций (например, поиск).

Нефункциональное — проверка атрибутов (скорость, безопасность).

Дымовое и регрессионное тестирование

Дымовое — быстрая проверка основных функций.

Регрессионное — полная проверка после изменений.

Санитарное тестирование

Санитарное тестирование — углублённая проверка конкретной функции после изменений.

Тестовая документация

Виды:

Тест-план, чек-лист, тест-кейс, баг-репорт, отчёт о тестировании.

Чек-лист:

Список проверок (например, "Проверить авторизацию").

Тест-кейс:

Подробное описание шагов, ожидаемого и фактического результата.

Отличие:

Чек-лист — что проверять.

Тест-кейс — как проверять.

Тест-план

Тест-план — документ, описывающий цели, объём и стратегию тестирования.

Вопросы:

Что тестировать? Как? Когда? Какие риски?

Баг-репорт

Атрибуты:

Заголовок, шаги воспроизведения, ожидаемый/фактический результат, серьёзность, приоритет.

Severity vs Priority:

Severity — влияние на систему (критичный, средний, низкий).

Priority — очерёдность исправления (высокий, средний, низкий).

Пример:

Опечатка в логотипе: низкая severity, но высокий priority.

Техники тест-дизайна

Классы эквивалентности:

Группы данных с одинаковым поведением (например, возраст 0-14).

Граничные значения:

Тестирование на границах классов (например, 14, 15).

Исследовательское и ad-hoc тестирование

Исследовательское — тестирование по сценарию с возможностью адаптации.

Ad-hoc — свободное тестирование без плана.

Тестирование мобильных приложений

Особенности:

Разные типы приложений (нативные, веб, гибридные).

Тестирование прерываний (звонки, SMS).

Проверка на разных устройствах и ОС.

API тестирование

API тестирование — проверка взаимодействия между компонентами системы.

Протоколы:

REST, SOAP, JSON-RPC.

Пример:

REST API — обмен данными через HTTP-методы (GET, POST).