Фазы тестирования: основные проблемы тестирования и задача выбора конечного набора тестов

Тестирование программного обеспечения — это процесс проверки соответствия реального поведения программы ожидаемому результату на конечном наборе тестов, выбранном определённым образом.

Фазы тестирования

Обычно процесс тестирования включает несколько ключевых этапов:

1. Анализ требований. Изучение документации, понимание бизнес-логики продукта, выявление потенциальных рисков и несоответствий.
2. Планирование тестирования. Разработка стратегии и плана тестирования, определение целей, объёма, приоритетов, необходимых ресурсов и графика.
3. Разработка тестовых сценариев. Создание тест-кейсов и чек-листов, трансформация требований в конкретные проверки.
4. Подготовка тестовой среды. Настройка необходимого окружения для проведения тестов (аппаратного, программного, сетевого).
5. Выполнение тестов. Непосредственная проверка функциональности приложения, регистрация обнаруженных дефектов.
6. Анализ результатов. Оценка выполнения требований, обнаруженных дефектов, покрытия тестами и эффективности тестирования.
7. Подготовка отчётов. Анализ результатов и составление итоговой документации.

В современных методологиях разработки (Agile, DevOps) эти этапы часто перекрываются и повторяются итеративно.

Основные проблемы тестирования

- Исчерпывающее тестирование невозможно. Проверка всех возможных входных значений, путей выполнения кода и комбинаций условий физически невыполнима из-за огромного количества возможных сценариев.
- Определение достаточности множества тестов. Нужно найти такое подмножество тестов, которое позволит сделать достоверные выводы о правильности реализации программы.
- Экономическая проблема. Необходимо выбрать конечное число тестов, которое даёт максимальную вероятность обнаружения ошибок при ограниченных ресурсах и сроках.
- Скопление дефектов. Разные модули системы могут содержать разное количество дефектов. Усилия по тестированию должны распределяться пропорционально фактической плотности дефектов (принцип Парето: 80% дефектов содержатся в 20% модулей).
- Парадокс пестицида. Если повторять одни и те же тестовые сценарии, со временем они будут находить всё меньше новых ошибок, так как ПО эволюционирует.
- Зависимость от контекста. Тестирование проводится по-разному в зависимости от специфики проекта (например, для систем с критичной безопасностью подход будет отличаться от тестирования новостного портала).
- Ограничения по времени. Часто тестировщики сталкиваются с нехваткой времени на выполнение всего объёма работ.
- Проблемы с коммуникацией. Неэффективное взаимодействие между командами разработчиков, тестировщиков и других участников проекта может приводить к ошибкам и задержкам.

Задача выбора конечного набора тестов

Основная задача — определить такой набор тестов, который обеспечит достаточную уверенность в качестве программы, минимизируя при этом затраты ресурсов.

Для решения этой задачи могут применяться:

- Анализ рисков. Определение критических областей, на которые могут повлиять новые функции или изменения.
- Техники анализа требований. Выявление ключевых требований, которые должны быть покрыты тестами.
- Экспертные знания. Использование опыта специалистов в области тестирования и прикладной области для выбора подходящих тестов.
- Методы оптимизации. Например, эквивалентное разбиение (разделение входных данных на классы с аналогичным поведением), анализ граничных значений, попарное тестирование (при большом количестве комбинаций параметров).
- Тестирование на основе конечного автомата. Построение тестов как комбинации тестов для всех состояний и переходов между ними, представленных в модели приложения.
- Случайное тестирование. Генерация тестов случайным образом по списку заданных характеристик.

Важно учитывать, что тестирование не гарантирует полного отсутствия дефектов, даже если ни один не был найден. Также необходимо регулярно пересматривать и корректировать тестовые наборы с учётом изменений в продукте