**ISTQB основные теоретические понятия**

Программа обучения Базового уровня

1. **Основы тестирования**

**Ошибка > Дефект > Отказ**

Человек может сделать ошибку (просчет), которая порождает дефект (недочет, помеху) в программном коде или документе. Если код с дефектом выполнен, то система может быть не в состоянии сделать то, что должна делать (или сделать то, что от нее не ожидают), порождая отказ.

Тестирование - это возможный способ оценки качества программного обеспечения в терминах найденных дефектов, как для функциональных требований, так и для нефункциональных требований. Тестирование может породить уверенность в качестве программного обеспечения, если не найдены или найдено немного дефектов. Когда во время тестирования находятся ошибки, качество систем программного обеспечения повышается, если эти дефекты исправлены. Тестирование также является деятельностью по обеспечению качества.

**Когда заканчивать тестирование?** Для принятия решения о достаточном объеме тестирования, необходимо принимать во внимание уровень рисков, включая технические риски, риски безопасности и бизнес риски, а так же проектные ограничения, такие как время и бюджет

Тестирование должно предоставить достаточную информацию заинтересованным лицам, чтобы принять обоснованные решения о передаче программного обеспечения или системы, прошедшей тестирование, на следующий шаг разработки или передачи клиентам.

**Активности в тестировании** существуют как до, так и после выполнения самих тестов. В эти активности входят планирование и управление, выбор тестовых условий, разработка и выполнение тестовых сценариев, проверка результатов, оценка критериев выхода, создание отчетов о процессе тестирования и об испытываемой системе и закрытие после того, как фаза тестирования была выполнена. Тестирование также включает рецензирование документации (включая исходный код) и проведение статического анализа.

**Цели тестирования:**

• Обнаружение дефектов

• Повышение уверенности в уровне качества

• Предоставление информации для принятия решений

• Предотвращение дефектов

Также в тестировании на этапе разработки (таком, как компонентное, интеграционное и системное тестирование), **основная цель** может заключаться в том, чтобы вызвать как можно больше отказов, чтобы дефекты в программном обеспечении были идентифицированы и могли быть исправлены.

В приемочном тестировании **основная цель** может состоять в том, чтобы подтвердить, что система работает, как ожидалось и повысить уверенность в том, что она удовлетворяет требованиям.

Тестирование в период сопровождения в основном заключается в проверке отсутствия новых дефектов, которые могли попасть во время разработки изменений. Во время эксплуатационного тестирования **основная цель** может заключаться в том, чтобы оценить системные характеристики, такие как надежность или доступность.

**Тестирование и отладка**

Динамическое тестирование может выявить отказы, вызванные дефектами

Отладка – это действия разработчиков, которые находят, анализируют и устраняют причину отказа.

*Ответственность за тестирование обычно несут тестировщики, а за отладку - разработчики.*

**Семь принципов тестирования:**

1. Тестирование демонстрирует наличие дефектов
2. Исчерпывающее тестирование недостижимо
3. Раннее тестирование
4. Скопление дефектов
5. Парадокс пестицида
6. Тестирование зависит от контекста
7. Заблуждение об отсутствии ошибок (Обнаружение и исправление дефектов не помогут, если созданная система не подходит пользователю и не удовлетворяет его ожиданиям и потребностям.)

**Основной процесс тестирования состоит из следующих направлений деятельности**:

• Планирование и управление

Планирование тестирования - это действия, направленные на определение целей тестирования и описание задач тестирования для достижения этих целей и миссии.

Управление тестированием – это постоянное сопоставление текущего положения дел с планом и отчетность о состоянии дел, включая отклонения от плана.

• Анализ и проектирование

Анализ и проектирование тестов - это деятельность, во время которой общие цели тестирования материализуются в тестовые условия и тестовые сценарии.

• Внедрение и реализация

Реализация и выполнение тестов – это деятельность, где процедуры тестирования или автоматизированные сценарии задаются последовательностью тестовых сценариев, а также собирается любая информация, необходимая для выполнения тестов, разворачивается окружающая среда, и запускаются тесты.

• Оценка критериев выхода и создание отчетов

Оценка критериев выхода - это деятельность, где выполнение тестов оценивается согласно определенным целям. Она должна быть выполнена для каждого уровня тестирования. Написание итогового отчета о тестировании для заинтересованных лиц.

• Действия по завершению тестов

Действия по завершению тестирования собирают данные о завершенных испытаниях для объединения опыта, тестового обеспечения, фактов и цифр. Действия по завершению тестирования происходят на тех этапах проекта, когда система программного обеспечения выпущена, тестирование завершено (или прервано), этап был завершен, или релиз по сопровождению был закончен.

1. **Место тестирования в жизненном цикле (ЖЦ) разработки ПО**

**Модели разработки ПО:**

* V-модель (Последовательная модель разработки), используют четыре уровня тестирования: компонентное, интеграционное, системное и приемочное.
* Итеративно-инкрементные модели разработки (Итеративно-инкрементный процесс разработки – это процесс, основывающийся на требованиях, дизайне, сборке и тестировании системы, созданной в результате серии коротких циклов разработки)
* Waterfall
* Спиральная (используется в экспериментальной разработке)

**Уровни тестирования**

* Компонентное тестирование

Компонентное тестирование (также известное как модульное) занимается поиском дефектов и верификацией функционирования программных модулей, программ, объектов, классов и т.п., которые можно протестировать изолированно. В процессе могут быть использованы заглушки, драйвера и эмуляторы. На практике компонентное тестирование обычно производится разработчиками, которые пишут код. Дефекты обычно исправляются сразу после того, как становятся известны, без занесения их в базу дефектов. Один из подходов к компонентному тестированию – составить автоматизированные тестовые сценарии до кодирования. Это называется разработкой, управляемой тестированием.

* Интеграционное тестирование

Интеграционное тестирование проверяет интерфейсы между компонентами, взаимодействие различных частей системы, таких как операционная системы, файловая система, аппаратное обеспечение, и интерфейсы между системами.

Интеграционное тестирование может состоять из одного или более уровней и может быть выполнено на тестовых объектах разного размера следующим образом:

1. Компонентное интеграционное тестирование проверяет взаимодействие между программными компонентами и производится после компонентного тестирования

2. Системное интеграционное тестирование проверяет взаимодействие между системами или между аппаратным обеспечением и может быть выполнено после системного тестирования. В этом случае, разработчики могут управлять только одной стороной интерфейса.

Тестирование специфичных нефункциональных характеристик ( например, производительности), может быть включено в интеграционное тестирование, наравне с функциональным.

На каждой стадии интеграции тестировщики концентрируют все внимание именно на интеграции как таковой. Например, если интегрируется модуль А с модулем В, они проверяют взаимодействие модулей, а не функциональность каждого из них, т.к. она должна быть проверена во время компонентного тестирования.

Подходы к интеграционному тестированию:

- Большой взрыв

- Снизу вверх

- Сверху вниз

* Системное тестирование

Системное тестирование сконцентрировано на поведении тестового объекта как целостной системы или продукта. Область тестирования должна быть четко определена.

Во время системного тестирования тестовое окружение должно быть как можно ближе к предполагаемому эксплуатационному окружению системы для минимизации риска пропуска отказов, связанных с эксплуатационным окружением системы.

Системное тестирование должно заниматься исследованием функциональных и нефункциональных требований к системе и качеством обрабатываемых данных.

Системное тестирование чаще всего выполняет независимая тестовая команда.

* Приемочное тестирование

Приемочным тестированием системы чаще всего занимаются заказчики или пользователи системы, а также другие заинтересованные лица.

**Основная цель** приемочного тестирования – проверка работоспособности системы, частей системы или отдельных нефункциональных характеристик системы. Поиск дефектов не является главной целью приемочного тестирования.

Приемочное тестирование оценивает готовность системы к развертыванию и использованию.

**Типичные виды приемочного тестирования:**

- Пользовательское приемочное тестирование

Обычно проверяет готовность системы для использования в бизнесе.

Эксплуатационное (приемочное) тестирование

Приемочное тестирование, проводимое системными администраторами, включает:

• Тестирование резервного копирования \ восстановления

• Восстановление после сбоев

• Управление пользователями

• Задачи сопровождения

• Задачи загрузки и миграции данных

• Периодическая проверка уязвимостей системы

Контрактное и правовое приемочное тестирование

Контрактное приемочное тестирование выполняется для проверки требований, предъявляемых контрактом к разрабатываемому ПО.

Приемочное тестирование на соответствие стандартам выполняется для проверки соответствия стандартам государственным, юридическим или стандартам безопасности.

Альфа и бета тестирование (или тестирование в условиях эксплуатации)

Разработчики рыночного, или коробочного, ПО часто хотят получить отзывы от потенциальных или существующих заказчиков до того, как начнется продажа продукта.

Альфа тестирование выполняется организацией, разрабатывающей продукт, но не группой разработчиков.

Бета тестирование, или тестирование в условиях эксплуатации, выполняется покупателями или потенциальными заказчиками на их собственных мощностях.

**Типы тестирования**

Типы тестирования определяются целями тестирования, которые могут быть следующими:

• Функция, выполняемая программой

• Нефункциональная характеристика качества, такая как надежность или удобство использования

• Структура или архитектура программы или системы

**Тестирование функций (Функциональное тестирование)**

Эти функции описывают, «что» эта система делает.

Функциональные тесты разрабатываются на основе функций и возможностей системы (описанных в документах или понятных тестировщикам) и их взаимодействия со специфичными системами и могут быть выполнены на всех уровнях тестирования (например, тесты для компонентов могут основываться на спецификациях компонентов).

Один из типов функционального тестирования, тестирование безопасности, исследует функции (например, брандмауэр) касающиеся обнаружения угроз, таких как вирусы, поступающих извне. Другой тип функционального тестирования, тестирование возможности взаимодействия, оценивает способность программного продукта взаимодействовать с одним или более указанными компонентами или системами.

**Тестирование нефункциональных характеристик (Нефункциональное тестирование)**

Нефункциональное тестирование включает, но не ограничивается, нагрузочное тестирование, тестирование производительности, стресс-тестирование, тестирование удобства использования, тестирование восстановления, тестирование надежности и тестирование переносимости. Это тестирование того, «как» система работает. Нефункциональное тестирование может выполняться на всех уровнях тестирования.

Нефункциональное тестирование рассматривает внешнее поведение программного обеспечения и в большинстве случаев использует разработку тестов методом черного ящика.

**Тестирование структуры/архитектуры программного обеспечения (Структурное тестирование)**

Покрытие – это часть структуры программы, которая была охвачена тестированием, выраженная в процентах. Если покрытие не равно 100%, то необходимо разрабатывать дополнительные тесты для покрытия пропущенных участков программы.

**Тестирование изменений: подтверждающее и регрессионное тестирование**

После того, как дефект обнаружен и исправлен, программу необходимо перепроверить, чтобы убедиться, что исходный дефект успешно устранен. Это называется подтверждением. Отладка (локализация и исправление дефекта) относится к процессу разработки, а не тестирования.

**Регрессионное тестирование** – это повторное тестирование уже протестированных программ после внесения в них изменений, чтобы обнаружить дефекты, внесенные или пропущенные в результате этих действий.

Тесты должны быть повторяемыми, если они должны использоваться для подтверждающего или регрессионного тестирования.

Регрессионное тестирование может выполняться на всех уровнях тестирования и включает функциональное, нефункциональное и структурное тестирование. Регрессионные наборы тестов запускаются множество раз и меняются медленно, поэтому регрессионное тестирование является хорошим кандидатом на автоматизацию.

**Тестирование в период сопровождения**

После установки система программного обеспечения обычно находится в эксплуатации в течение многих лет. В это время сама система, ее конфигурация или среда исполнения часто изменяются или расширяются. Ранее планирование релизов крайне важно для успешного тестирования в период сопровождения. При этом необходимо отличать запланированные выпуски и срочные исправления. Тестирование в период сопровождения выполняется на текущей ОС и может быть вызвано модификацией, переносом или прекращение эксплуатации данной системы.

Тестирование в период сопровождения при переносе (например, между платформами) должно включать как эксплуатационные тесты нового окружения, так и изменения в самой программе.

В дополнение к тестированию изменений, тестирование периода сопровождения включает регрессионное тестирование тех частей программы, которые не подвергались изменениям.

1. **Статические методы**

**Статические методы и процесс тестирования**

В отличие от динамического тестирования, которое требует запуска ПО, при статическом тестировании код или проектная документация исследуется вручную (рецензирование) или с помощью автоматизированных средств анализа (статический анализ) без исполнения кода.

**Рецензирование** - вид тестирования ПО (включая код), который может проводиться перед динамическим тестированием. Исправление дефектов, обнаруженных во время рецензирования на ранних этапах жизненного цикла ПО (например, дефектов, найденных в требованиях), часто обходится значительно дешевле по сравнению с дефектами, найденными во время выполнения тестов и исполнения кода.

Главной составляющей ручного процесса является исследование и комментирование программного продукта.

Преимущества рецензирования: раннее обнаружение и исправление дефектов, улучшение продуктивности разработки, уменьшение времени разработки, уменьшение времени и стоимости тестирования, сокращение стоимости жизненного цикла, уменьшение числа дефектов и улучшение коммуникаций. Во время рецензирования могут быть найдены упущения, например, в требованиях, которые маловероятно найти во время динамического тестирования.

У рецензирования, статического анализа и динамического тестирования общая цель – обнаружение дефектов. Методы дополняют друг друга, так как с разной эффективностью находят различные типы дефектов.

В отличие от динамического тестирования статические методы находят причины сбоя (дефекты), а не сами отказы.

Типичные дефекты, которые легче найти при рецензировании, чем при динамическом тестировании: отклонения от стандартов, дефекты в требованиях или дизайне, недостаточная пригодность к сопровождению и некорректные спецификации интерфейса.

**Процесс рецензирования**

**Введение**

Типы рецензирования варьируются от неформальных, когда нет письменных инструкций для экспертов, до формальных, которые предполагают участие команды, документирование результатов рецензирования и процесса ее проведения.

Стиль проведения рецензирования зависит от согласованных целей (например, нахождение дефектов, достижение понимания, обучение тестировщиков или новых членов команды, или обсуждение и принятие согласованного решения).

**Действия (шаги) формального рецензирования**

Типичное формальное рецензирование включает следующие действия:

1. Планирование (определение критерия рецензирования, выбор участников, распределение ролей, определение критериев входа и выхода, отбор частей документации для рецензирования, проверка критерия входа.
2. Старт (Распределение документов, Объяснение целей, процесса участникам)
3. Индивидуальная подготовка (Подготовка к экспертному собранию - анализ документов, Конспектирование потенциальных дефектов, вопросов и комментариев)
4. Конспектирование потенциальных дефектов, вопросов и комментариев
5. Повторная обработка (Исправление найденных дефектов, Запись обновленного статуса дефектов)
6. Отслеживание (Проверка того, что дефекты были назначены, Сбор метрик, Проверка критерия выхода)

**Роли и Обязанности в формальном рецензировании**

• Менеджер: принимает решение о проведении, выделяет время, определяет, были ли достигнуты цели.

• Модератор: руководит проведением рецензирования документа или набора документов, включая планирование рецензирования, проведение встречи и отслеживание.

• Автор: автор или главный ответственный за документ(ы) для рецензирования.

• Эксперты: люди со специальным техническим или бизнес опытом и знаниями (часто называются проверяющими или инспекторами), которые после необходимой подготовки, определяют и описывают проблемы и вопросы, найденные в проверяемом ПО.

• Секретарь: документирует все проблемы и открытые вопросы, которые были определены во время экспертной встречи

**Типы рецензирований**

* Неформальное рецензирование (Отсутствие формального процесса, может принимать форму парного программирования, Результаты могут быть документированы, Эффективность зависит от экспертов, результат при минимуме затрат)
* Сквозной контроль (Встреча проводится автором, может быть в форме сценариев)
* Технический анализ (Документированный процесс, В идеале проводится обученным модератором (не автором), подготовка экспертов, Подготовка отчета о рецензировании, который включает список найденных проблем и вопросов, заключение, соответствует ли программный продукт требованиям, а также необходимые рекомендации, Главные цели: обсуждение, принятие решений, оценка альтернатив, нахождение дефектов, решение технических проблем и проверка соответствия требованиям, планам, нормам и стандартам)
* Инспекция (Проводится обученным модератором, Роли определены, Включает сбор метрик, Формальный процесс основан на правилах и контрольных списках, Определены критерии входы и выхода, Инспекционный отчет, Главная цель: поиск дефектов)

**Факторы успешного проведения рецензирования:**

• Каждое рецензирование должно иметь четкие, определенные заранее цели

• Вовлечение правильных людей в соответствии с целями рецензирования

• Тестировщики - ценные эксперты, которые вносят свой вклад в рецензирование, а также изучают продукт, что позволяет создавать тесты раньше

• Найденные дефекты ожидаемы и выражены объективно

• Необходимо учитывать человеческий фактор и психологические аспекты (например, создание положительного опыта для автора)

• Рецензирование должно проводиться в атмосфере доверия, результаты не должны использоваться для оценки участников

• Должны применяться те методы рецензирования, которые подходят для достижения поставленных целей, соответствуют типу и уровню программного продукта и экспертов

• Использование контрольных списков или ролей

• Обучение методам рецензирования, особенно для более формальных типов рецензирования, таких как инспекция

• Руководство поддерживает хороший процесс рецензирования (например, выделяет соответствующее время в графиках разработки проекта)

• Акцент на изучении и улучшении процесса

**Статический анализ с помощью инструментальных средств**

Цель статического тестирования - нахождение дефектов в коде или моделях ПО. Фактически статический анализ – это исследование ПО с помощью специального инструмента без его запуска, при динамическом тестировании ПО требуется запуск кода. Статический анализ выявляет дефекты, которые сложно найти при динамическом тестировании.Cтатический анализ больше находит дефекты, чем сбои. Инструментальные средства статического анализа анализируют код программы (например, потоки управления и поток данных), а так же сгенерированный код, например, HTML или XML.

**Типичные дефекты, которые могут быть найдены при статическом анализе:**

• Обращение к переменной, которой не присвоено значение

• Несоответствие интерфейсов между модулями и компонентами

• Переменные, которые не используются или некорректно объявлены

• Невыполняемые ветки кода

• Пропущенная или неверная логика (например, бесконечные циклы)

• Излишне сложные конструкции

• Отклонение от стандартов программирования

• Уязвимость в безопасности

• Нарушение синтаксиса в коде или моделях ПО

Инструментальные средства статического анализа используются разработчиками до или во время компонентного или интеграционного тестирования или при добавлении кода в инструменте управления конфигурацией, а также дизайнерами во время моделирования ПО.

**4. Методы проектирования тестов**

Ожидаемые результаты должны создаваться как часть спецификаций тестовых сценариев и включать в себя выходные данные, изменения в данных и состояниях, и любые иные последствия теста. Если ожидаемые результаты не были определены, правдоподобные, но ошибочные результаты могут быть приняты за корректные.

В идеальных условиях ожидаемые результаты должны быть определены до момента выполнения теста.

Во время реализации теста тестовые сценарии разрабатываются, реализуются, получают приоритеты и формируют спецификацию процедуры тестирования

Различные процедуры тестирования и автоматические тестовые сценарии собираются в расписание выполнения тестов, определяющее, в какой очередности, когда и кем эти тестовые процедуры и сценарии должны быть выполнены.

Расписание выполнения тестов должно учитывать такие факторы, как регрессионные тесты, приоритеты и технические и логические зависимости.

**Категории методов проектирования тестов**

Целью метода проектирования тестов является определение тестовых условий и тестовых сценариев. Классическим является разделение методов тестирования на методы черного и белого ящиков.

Методы черного ящика - это способ определить и выбрать тестовые условия или сценарии для компонента или системы (как функциональные, так и не функциональные), на основе анализа базиса тестирования и опыта разработчиков, тестировщиков и пользователей, без отсылки к внутренней структуре компонента или системы.

Методы белого ящика - основываются на анализе структуры компонента или системы.

Также рассматриваются методы создания тестов на основе опыта:

• для определения тестовых сценариев используются человеческие знания и опыт.

• знания тестировщиков, разработчиков, пользователей и заинтересованных лиц о программном продукте, его использовании и окружении, являются одним из источников информации.

• знания о вероятных дефектах и их распределении являются другим источником информации

**Методы, основанные на спецификациях, или методы черного ящика**