**Предмет: 05.03 Тестирование ИС**

**Группа: 2Б ИСиП**

**Дата: 2.05.2025**

**Задание: законспектировать в тетради**

**Тема: Тестовые сценарии, тестовые варианты**

Для разработки тестовых сценариев и выполнения тестов используются системы управления тестированием, существенно повышающие производительность тест-дизайнеров и тестировщиков, а также обеспечивающие видимость уровня качества приложений среди всех участников проекта.

Тестовые сценарии неразрывно связаны с требованиями, изменения в которых должны своевременно отражаться в тестовой документации, что позволяет сделать система управления жизненным циклом разработки приложений, при помощи механизма трассировок.

При выполнении теста тестировщик отмечает результат прохождения одного шага или всего тестового сценария, прикрепляет обнаруженные ошибки и другую вспомогательную информацию: скриншоты, дампы, логи и т.п.

Тестовые сценарии удобно объединять в тест-планы по назначению:

· тестирование релиза, то есть очередной версии продукта;

· тестирование развертывания;

· тестирование удобства использования;

· конфигурационное тестирование;

· тестирование безопасности и т.п.

Зачастую ручное тестирование превращается в рутину и занимает значительное время, что отрицательно сказывается на скорости выпуска релизов. Автоматизация тестирования позволяет:

· высвободить ресурсы для проведения более сложных видов тестирования;

· снизить количество дефектов, доходящих до стадии контроля качества;

· ускорить выпуск релизов.

Сведение результатов автоматических и ручных тестов в системе управления качеством, позволяет всем участникам проекта видеть уровень качества очередного релиза, контролировать его изменение и опираться на эту информацию при планировании своей работы.

**Получение результатов тестирования и их анализ.**

**Получение результатов тестирования и их анализ.**

Получение результатов тестирования напрямую зависит от средств тестирования. В моем случае это был встроенный в 1С механизм отладки и система контроля ошибок.

Перед получением результата программа проходит несколько уровней:

· Тестирование компонентов — тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. Часто тестирование компонентов осуществляется разработчиками программного обеспечения.

· Интеграционное тестирование — тестируются интерфейсы между компонентами, подсистемами или системами. При наличии резерва времени на данной стадии тестирование ведётся итерационно, с постепенным подключением последующих подсистем.

· Системное тестирование — тестируется интегрированная система на её соответствие требованиям.

· Альфа-тестирование — имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком. Чаще всего альфа-тестирование проводится на ранней стадии разработки продукта, но в некоторых случаях может применяться для законченного продукта в качестве внутреннего приёмочного тестирования. Иногда альфа-тестирование выполняется под отладчиком или с использованием окружения, которое помогает быстро выявлять найденные ошибки. Обнаруженные ошибки могут быть переданы тестировщикам для дополнительного исследования в окружении, подобном тому, в котором будет использоваться программа.

· Бета-тестирование — в некоторых случаях выполняется распространение предварительной версии (в случае проприетарного программного обеспечения иногда с ограничениями по функциональности или времени работы) для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

Часто для свободного и открытого программного обеспечения стадия альфа-тестирования характеризует функциональное наполнение кода, а бета-тестирования — стадию исправления ошибок. При этом как правило на каждом этапе разработки промежуточные результаты работы доступны конечным пользователям.

Тестовое покрытие

**Тестовое покрытие** - это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.

Если рассматривать тестирование как "проверку соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов", то именно этот конечный набор тестов и будет определять тестовое покрытие:

*Чем выше требуемый уровень тестового покрытия, тем больше тестов будет выбрано, для проверки тестируемых требований или исполняемого кода.*

Сложность современного программного обеспечения и инфраструктуры сделало невыполнимой задачу проведения тестирования со 100% тестовым покрытием. Поэтому для разработки набора тестов, обеспечивающего более менее высокий уровень покрытия можно использовать специальные инструменты либо техники тест дизайна.

Существуют следущие подходы к оценке и измерению тестового покрытия:

1. [**Покрытие требований (Requirements Coverage)**](http://www.protesting.ru/testing/testcoverage.html#requirements)- оценка покрытия тестами функциональных и нефункциональных требований к продукту путем построения матриц трассировки (traceability matrix).
2. [**Покрытие кода (Code Coverage)**](http://www.protesting.ru/testing/testcoverage.html#code)- оценка покрытия исполняемого кода тестами, путем отслеживания непроверенных в процессе тестирования частей программного обеспечения.
3. [**Тестовое покрытие на базе анализа потока управления**](http://www.protesting.ru/testing/testcoverage.html#flow)- оценка покрытия основанная на определении путей выполнения кода программного модуля и создания выполняемых тест кейсов для покрытия этих путей.

# Различия:

Метод покрытия требований сосредоточен на проверке соответствия набора проводимых тестов требованиям к продукту, в то время как анализ покрытия кода - на полноте проверки тестами, разработанной части продукта (исходного кода), а анализ потока управления - на прохождении путей в графе или модели выполнения тестируемых функций (Control Flow Graph).

# Ограничения:

Метод оценки покрытия кода не выявит нереализованные требования, так как работает не с конечным продуктом, а с существующим исходным кодом.

Метод покрытия требований может оставить непроверенными некоторые участки кода, потому что не учитывает конечную реализацию.

# Покрытие требований (Requirements Coverage)

Расчет тестового покрытия относительно требований проводится по формуле:

Tcov = (Lcov/Ltotal) \* 100%

где:

**Tcov** - тестовое покрытие

**Lcov** - количество требований, проверяемых тест кейсами

**Ltotal** - общее количество требований

Для измерения покрытия требований, необходимо проанализировать требования к продукту и разбить их на пункты. Опционально каждый пункт связывается с тест кейсами, проверяющими его. Совокупность этих связей - и является матрицей трассировки. Проследив связи, можно понять какие именно требования проверяет тестовый случай.

Тесты не связанные с требованиями не имеют смысла. Требования, не связанные с тестами - это "белые пятна", т.е. выполнив все созданные тест кейсы, нельзя дать ответ реализовано данное требование в продукте или нет.

Для оптимизации тестового покрытия при тестировании на основании требований, наилучшим способом будет использование стандартных техник тест дизайна.

# Покрытие кода (Code Coverage)

Расчет тестового покрытия относительно исполняемого кода программного обеспечения проводится по формуле:

Tcov = (Ltc/Lcode) \* 100%

где:

**Tcov** -тестовое покрытие

**Ltc** - кол-ва строк кода, покрытых тестами

**Lcode** - общее кол-во строк кода.

В настоящее время существует инструментарий (например: [**Clover**](http://www.atlassian.com/software/clover/)), позволяющий проанализировать в какие строки были вхождения во время проведения тестирования, благодаря чему можно значительно увеличить покрытие, добавив новые тесты для конкретных случаев, а также избавиться от дублирующих тестов. Проведение такого анализа кода и последующая

оптимизация покрытия достаточно легко реализуется в рамках тестирования белого ящика (white-box testing) при модульном, интеграционном и системном тестировании; при тестировании же черного ящика (black-box testing) задача становится довольно дорогостоящей, так как требует много времени и ресурсов на установку, конфигурацию и анализ результатов работы, как со стороны тестировщиков, так и разработчиков.

# Тестовое покрытие на базе анализа потока управления

**Тестирование потоков управления** (Control Flow Testing) - это одна из техник тестирования белого ящика, основанная на определении путей выполнения кода программного модуля и создания выполняемых тест кейсов для покрытия этих путей.

Фундаментом для тестирования потоков управления является построение графов потоков управления (Control Flow Graph), основными блоками которых являются:

* блок процесса - одна точка входа и одна точка выхода
* точка альтернативы - одна точка входа, две и более точки выхода
* точка соединения - две и более точек входа, одна точка выхода

Для тестирования потоков управления определены разные **уровни тестового покрытия**:

|  |  |  |
| --- | --- | --- |
| **Уровень** | **Название** | **Краткое описание** |
| Уровень  0 | -- | “Тестируй все что протестируешь, пользователи протестируют остальное” На английском языке это звучит намного элегантнее: *“Test whatever you test, users will test the rest”* |
| Уровень  1 | Покрытие операторов | Каждый оператор должен быть выполнен как минимум один раз. |
| Уровень  2 | Покрытие альтернатив [**[2]**](http://www.protesting.ru/testing/testcoverage.html#%5B2%5D)/ Покрытие ветвей | Каждый узел с ветвлением (альтернатива) выполнен как минимум один раз. |
| Уровень  3 | Покрытие условий | Каждое условие, имеющее TRUE и FALSE на выходе, выполнено как минимум один раз. |
| Уровень  4 | Покрытие условий альтернатив | Тестовые случаи создаются для каждого условия и альтернативы |
| Уровень  5 | Покрытие множественных условий | Достигается покрытие альтернатив, условий и условий альтернатив (Уровни 2, 3 и 4) |
| Уровень  6 | “Покрытие бесконечного числа путей” | Если, в случае зацикливания, количество путей становится бесконечным, допускается существенное их сокращение, ограничивая количество циклов выполнения, для уменьшения числа тестовых случаев. |
| Уровень  7 | Покрытие путей | Все пути должны быть проверены |

# Таблица 1. Уровни тестового покрытия

Основываясь на данных этой таблицы, вы сможете спланировать необходимый уровень тестового покрытия, а также оценить уже имеющийся.