ТЕСТОВОЕ ПОКРЫТИЕ ТЕХНИКИ ТЕСТ-ДИЗАЙНА

Что сегодня будет на занятии?

- Тестовое покрытие, где и зачем оно применяется
- Тест-дизайн и его основные техники
- Примеры применения техник тест-дизайна
- Плюсы и минусы различных методов тест-дизайна
- Практика

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, осуществляемую 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.
Для разработки набора тестов, обеспечивающего высокий уровень покрытия, можно использовать специальные техники тест дизайна.

Существуют следующие подходы к оценке и измерению тестового покрытия:

- **Покрытие требований (Requirements Coverage)**- оценка покрытия тестами функциональных и нефункциональных требований к продукту
- **Покрытие кода (Code Coverage)**- оценка покрытия исполняемого кода тестами, путем отслеживания непроверенных в процессе тестирования частей программного обеспечения.

Покрытие требований (Requirements Coverage)

Расчет тестового покрытия относительно требований проводится по формуле:

Tcov = (Lcov/Ltotal) * 100% где:

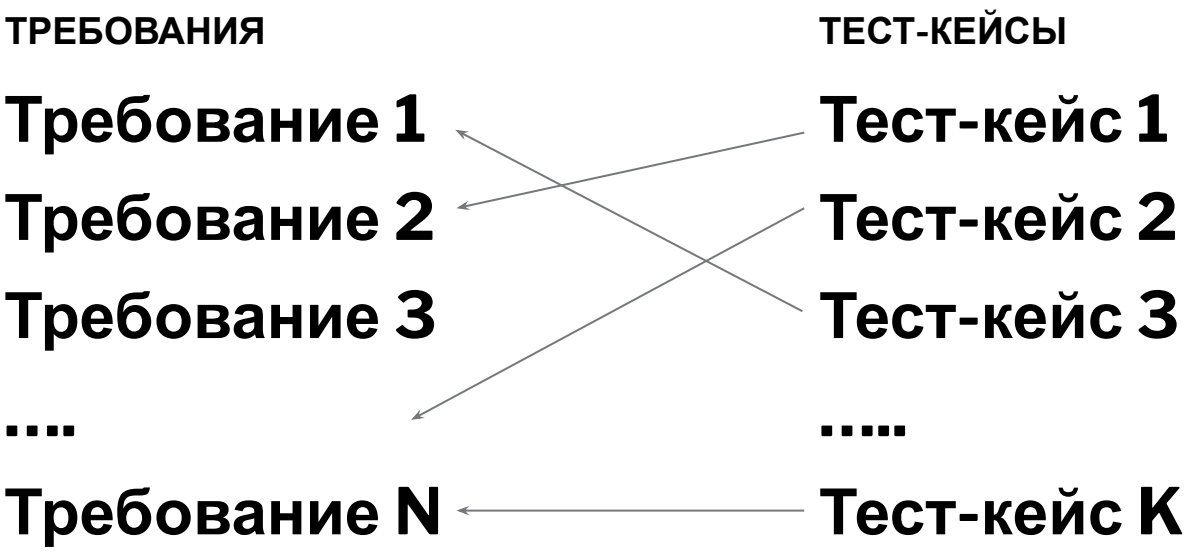
Tcov - тестовое покрытие

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.

ПОКРЫТИЕ ТРЕБОВАНИЙ



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.

ТЕСТ-ДИЗАЙН

Тест-дизайн - один из первоначальных этапов тестирования программного обеспечения, этап планирования и проектирования тестов. Тест дизайн представляет собой продумывание и написание тестовых случаев (test case), в соответствии с требованиями проекта, критериями качества будущего продукта и финальными целями тестирования.

ЦЕЛИ ТЕСТ ДИЗАЙНА

Обеспечить покрытие функционала приложения тестами:

- Тесты должны покрывать весь функционал
- Тестов должно быть минимально достаточно

Тест дизайн задачи

- Проанализировать требования к продукту
- Оценить риски возможные при использовании продукта
- Написать достаточное минимальное количество тестов
- Разграничить тесты на приемочные, критические, расширенные

ТЕХНИКИ ТЕСТ-ДИЗАЙНА

Техники тест-дизайна - это рекомендации, советы и правила, по которым стоит разрабатывать тест для проведения тестирования приложения. Это не образцы тестов, а только рекомендации к применению. В частности различные инженеры могут работая под одним и тем же проектом создать различный набор тестов. Правильным будет считаться тот набор тестов, который за меньшее количество проверок обеспечит более полное покрытие тестами.

ТЕХНИКИ ТЕСТ-ДИЗАЙНА

- Разделение на классы эквивалентности
- Анализ граничных значений
- Таблица принятия решения
- Причина – следствие
- Предугадывание ошибки

КЛАССЫ ЭКВИВАЛЕНТНОСТИ

Класс эквивалентности (equivalence class) — одно или несколько значений ввода, к которым программное обеспечение применяет одинаковую логику.

Техника анализа классов эквивалентности

это техника, при которой мы разделяем функционал (часто диапазон возможных вводимых значений) на группы эквивалентных по своему влиянию на систему значений. Такое разделение помогает убедиться в правильном функционировании целой системы — одного класса эквивалентности, проверив только один элемент этой группы. Эта техника заключается в разбиении всего набора тестов на классы эквивалентности с последующим сокращением числа тестов.

ПРИЗНАКИ ЭКВИВАЛЕНТНОСТИ ТЕСТОВ:

- направлены на поиск одной и той же ошибки;
- если один из тестов обнаруживает ошибку, другие скорее всего, тоже её обнаружат;
- если один из тестов не обнаруживает ошибку, другие, скорее всего, тоже её не обнаружат;
- тесты используют схожие наборы входных данных;
- для выполнения тестов мы совершаем одни и те же операции;
- тесты генерируют одинаковые выходные данные или приводят приложение в одно и то же состояние;
- все тесты приводят к срабатыванию одного и того же блока обработки ошибок;
- ни один из тестов не приводит к срабатыванию блока обработки ошибок.

АЛГОРИТМ ИСПОЛЬЗОВАНИЯ ЭКВИВАЛЕНТНОСТИ КЛАССОВ

1. Определить классы эквивалентности.

Это главный шаг техники, т.к. во многом от него зависит эффективность её применения.

2. Выбрать одного представителя от каждого класса эквивалентности.

На этом этапе следует выбрать один тест из эквивалентного набора тестов.

3. Выполнение тестов.

На этом шаге следует выполнить тесты от каждого класса эквивалентности.

КЛАССИЧЕСКИЙ ПРИМЕР АНАЛИЗА КЛАССОВ ЭКВИВАЛЕНТНОСТИ

Есть поле ввода с диапазоном допустимых значений от 1 до 100

1. **Что делаем первым делом?** Определяем классы эквивалентности
2. **Дальше?** Выбираем одного представителя от каждого класса
3. **И последний шаг?** Проводим тест на выбранных значениях

ПЛЮСЫ И МИНУСЫ ТЕХНИКИ АНАЛИЗА ЭКВИВАЛЕНТНЫХ КЛАССОВ

К плюсам можно отнести отсеивание огромного количества значений ввода, использование которых просто бессмысленно.

К минусам можно отнести неправильное использование техники, из-за которого есть риск упустить баги.

ТЕХНИКА АНАЛИЗА ГРАНИЧНЫХ ЗНАЧЕНИЙ

Граничные значения — это те места, в которых один класс эквивалентности переходит в другой.

Это техника проверки поведения продукта на крайних (граничных) значениях входных данных. Граничное тестирование также может включать тесты, проверяющие поведение системы на входных данных, выходящих за допустимый диапазон значений. При этом система должна определённым (заранее оговоренным) способом обрабатывать такие ситуации. Например, с помощью исключительной ситуации или сообщения об ошибке.

АЛГОРИТМ ИСПОЛЬЗОВАНИЯ ТЕХНИКИ ГРАНИЧНЫХ ЗНАЧЕНИЙ:

1. Выделить классы эквивалентности;

Как и в предыдущей технике, этот шаг является очень важным и от того, насколько правильным будет разбиение на классы эквивалентности, зависит эффективность тестов граничных значений.

2. Определить граничные значения этих классов;

3. Понять, к какому классу будет относиться каждая граница;

4. Провести тесты по проверке значения до границы, на границе и сразу после границы.

КЛАССИЧЕСКИЙ ПРИМЕР ПРИМЕНЕНИЯ ТЕХНИКИ ГРАНИЧНЫХ ЗНАЧЕНИЙ

В поле ввода можно внести только цифры от 0 до 10 000.

1. Определяем классы эквивалентности
2. Выбираем представителя из каждого класса
3. Проводим тесты

ТАБЛИЦА ПРИНЯТИЯ РЕШЕНИЙ

Это хороший инструмент для фиксирования требований и описания функциональности приложения.

Этими таблицами очень удобно описывать бизнес логику приложения, и в добавок они могут служить отличной **основой для создания тест кейсов**.

В таблицах решений представлен набор условий, одновременное выполнение которых должно привести к определённому действию

ШАБЛОН ТАБЛИЦЫ РЕШЕНИЙ СЛЕДУЮЩИЙ

	Правило 1	Правило 2	…	Правило p
Сущность				
Свойство-1				
Свойство-2				
…				
Свойство-m				
Действия				
Действие-1				
Действие-2				
…				
Действие-n				

УПРОСТИМ ШАБЛОН ДЛЯ ПОНИМАНИЯ

	Тест кейс 1	Тест кейс 2	…	Тест кейс p
Входные данные				
Свойство-1				
Свойство-2				
…				
Свойство-m				
Ожидаемый результат				
Действие-1				
Действие-2				
…				
Действие-n				

ТАБЛИЦА РЕШЕНИЙ НА ПРИМЕРЕ

Представим, что тестируем приложение для страховой компании. Это приложение вычисляет скидку на страхование автомобилей, взаимозависимости от того, был ли водитель хорошим студентом и состоит ли он в браке

Всего 2 сущности

	Правило 1	Правило 2	Правило 3	Правило 4
Сущность				
Состоит в браке?	Yes	Yes	No	No
Хороший студент?	Yes	No	Yes	No

В ЗАВИСИМОСТИ ОТ КОМБИНАЦИИ ЗНАЧЕНИЙ НАШИХ СУЩНОСТЕЙ У НАС ВЫЧИСЛЯЕТСЯ СКИДКА

	Правило 1	Правило 2	Правило 3	Правило 4
Сущность				
Состоит в браке?	Yes	Yes	No	No
Хороший студент?	Yes	No	Yes	No
Действия				
Скидка ($)	60	25	50	0

ТЕПЕРЬ МОЖНО СОЗДАВАТЬ ТЕСТ-КЕЙСЫ:

	Тест кейс 1	Тест кейс 2	Тест кейс 3	Тест кейс 4
Входные данные				
Состоит в браке?	Yes	Yes	No	No
Хороший студент?	Yes	No	Yes	No
Ожидаемый результат				
Скидка ($)	60	25	50	0

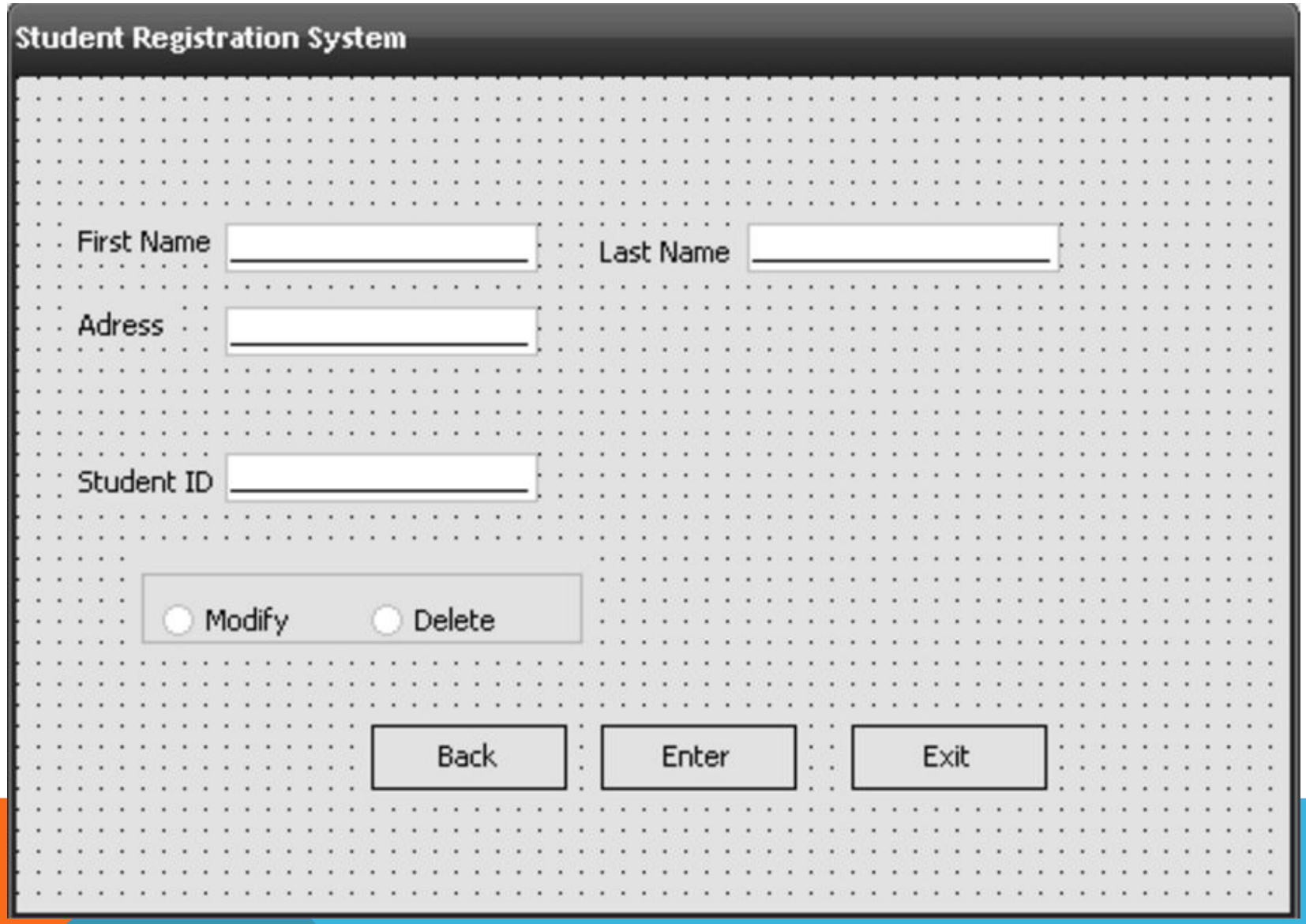
или вот так:

Test Case ID	Состоит в браке?	Хороший студент?	Ожидаемый результат
TC1	Yes	Yes	скидка = 60
TC2	Yes	No	скидка = 25
TC3	No	Yes	скидка = 50
TC4	No	No	скидка = 0

ПРИМЕР НА БОЛЬШЕМ КОЛИЧЕСТВЕ ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ.

Представим, что мы тестируем форму регистрации студентов.
С помощью этой формы мы можем создавать, редактировать и удалять студентов из БД.
Чтобы создать нового студента нужно ввести его имя, фамилию, адрес и нажать кнопку enter. Запись о студенте будет сохранена в БД и приложение вернет ID новосозданного студента.

Для того чтобы отредактировать или удалить данные студента нужно ввести ID студента и выбрать радиобаттон Modify или Delete соответственно.


Student Registration System
First Name
Last Name
Adress
Student ID
Modify
Delete
Back
Enter
Exit

КАК БУДЕТ ВЫГЛЯДЕТЬ ТАБЛИЦА РЕШЕНИЙ?

	Правило 1	Правило 2	Правило 3	Правило 4	Правило 5	Правило 6	Правило 7	Правило 8	Правило 9	Правило 10	Правило 11	Правило 12	Правило 13	Правило 14	Правило 15	Правило 16
Сущность																
Введены данные о студенте?	No	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Введен ID студента?	No	No	No	No	Yes	Yes	Yes	Yes	No	No	No	No	Yes	Yes	Yes	Yes
Выбран Modify?	No	No	Yes	Yes	No	No	Yes	Yes	No	No	Yes	Yes	No	No	Yes	Yes
Выбран Delete?	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes
Действие																
Создать нового студента	No	No	No	No	No	No	No	No	Yes	No	No	No	No	No	No	No
Отредактировать данные студента	No	No	No	No	No	No	Yes	No	No	No	Yes	No	No	No	No	No
Удалить студента	No	No	No	No	No	Yes	No	No	No	No	No	No	No	No	No	No

МОЖНО УПРОСТИТЬ!

	Тест Кейс 1	Тест Кейс 2	Тест Кейс 3	Тест Кейс 4	Тест Кейс 5	Тест Кейс 6	Тест Кейс 7	Тест Кейс 8	Тест Кейс 9	Тест Кейс 10	Тест Кейс 11	Тест Кейс 12
Входные данные												
Введены данные о студенте?	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Введен ID студента?	No	No	No	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes
Выбран Modify?	No	No	Yes	No	No	Yes	No	No	Yes	No	No	Yes
Выбран Delete?	No	Yes	No	No	Yes	No	| No	Yes	No	No	Yes	No
Ожидаемый Результат												
Создать нового студента	No	No	No	No	No	No	Yes	No	No	No	No	No
Отредактировать данные студента	No	No	No	No	No	Yes	No	No	Yes	No	No	No
Удалить студента	No	No	No	No	Yes	No	No	No	No	No	No	No

ПРИЧИНА/СЛЕДСТВИЕ

Причина / Следствие (**Cause/Effect - CE**). Это, как правило, ввод комбинаций условий (причин), для получения ответа от системы (Следствие). Например, вы проверяете возможность добавлять клиента, используя определенную экранную форму. Для этого вам необходимо будет ввести несколько полей, таких как "Имя", "Адрес", "Номер Телефона" а затем, нажать кнопку "Добавить" - эта "Причина". После нажатия кнопки "Добавить", система добавляет клиента в БД и показывает его номер на экране - это "Следствие".

ПРЕДУГАДЫВАНИЕ ОШИБКИ

Предугадывание ошибки (**Error Guessing - EG**). Это когда тест аналитик использует свои знания системы и способность к интерпретации спецификации на предмет того, чтобы "предугадать" при каких входных условиях система может выдать ошибку. Например, спецификация говорит: "пользователь должен ввести код". Тест аналитик, будет думать: "Что, если я не введу код?", "Что, если я введу неправильный код? ", и так далее. Это и есть предугадывание ошибки.

ПРАКТИКА

Использование техник тест-дизайна