**Decision Table Testing**

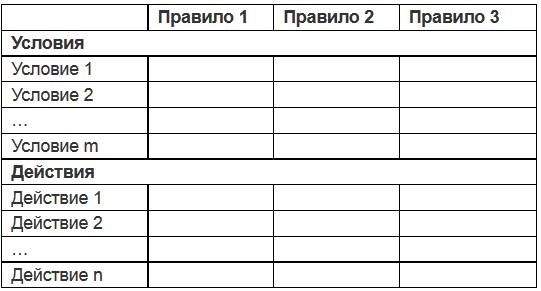
1. **Суть метода**

**Decision Table** **Testing** - класс техник тест-дизайна, которая относится к техникам «черного ящика». Она очень удобна в ситуациях, когда имеется очень много условий, в зависимости от комбинаций которых получаем различные действия или комбинации этих действий.

Если спецификация или требования написаны с использованием if-then-else логики, тогда данная таблица хорошо ложится на тестирование приложения. Данная таблица хорошо показывает причинно-следственную связь между различными условиями.

Decision Tables(Таблицы принятия решения, DT) описывают логику приложения основываясь сущностях(свойствах/условиях) состояния системы. В таблицах решений представлен набор условий, одновременное выполнение которых должно привести к определенному действию.

Общий вид таблиц решений представлен на рисунке ниже:



Условия(conditions) от 1 до m - это разные свойства системы, они представляют в таблице входные данные, которые можно ввести в систему.

Действия(actions) от 1 до n - это действия которые могут произойти с указанной комбинацией условий. Каждое правило определяет уникальный набор входных данных всех свойств, которые приводят к исполнению конкретных действий.

1. **Пример использования Decision Table**

Пусть 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. Величина скидки вычисляется с помощью decision table.

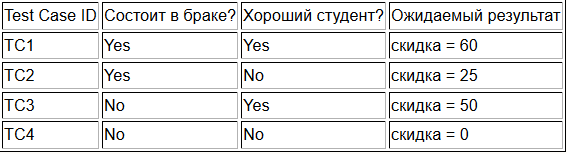
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Правило 1 | Правило 2 | Правило 3 | Правило 4 |
| **Условия** |  |  |  |  |
| Состоит в браке? | Yes | Yes | No | No |
| Хороший студент? | Yes | No | Yes | No |
| **Действия** |  |  |  |  |
| Скидка ($) | 60 | 25 | 50 | 0 |

Из таблицы видно, что выделено два условия и одно действие. Каждое из условия бинарное(принимает только два значения – true\false). В зависимости от комбинаций условия и правила получаем некоторый результат – действие. Для рассматриваемого примера таким действием является величина предоставляемой скидки.

Можно легко перевести таблицу в набор тест-кейсов. Правило для создания тест-кейсов просто – необходимо создать как минимум 1 тест-кейс на каждое правило в таблице. Тогда DT примет следующий вид:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 перевернув таблицу, получим готовые тест-кейсы в следующем виде:



1. **Достоинства метода**
2. Удобный инструмент для фиксирования требований и описания функциональности приложения
3. Заставляет лучше проанализировать тестируемый продукт, систематизировать все знания по нему. Удобно описывать логику приложения
4. Служат основой для создания тест-кейсов
5. Таблица понятна всей проектной команде
6. Эффективна для определения комбинаций, которые в других случаях могут быть не замечены.
7. Применяется на любых уровнях тестирования
8. **Недостатки метода**
9. DT не очень хорошо масштабируемы. Может потребоваться расширить или упростить таблицу.

a)Увеличить таблицу может понадобиться в связи с какими-либо новыми данными. Так, например, если рассматриваем светофор – помимо возможных значений(горит красный, зеленый, желтый) может быть введено значение «мигающий желтый». В связи с этим увеличится число колонок. Однако подобного расширения в процессе применения техники можно избежать сразу, чётко проанализировав все имеющиеся данные. Можно выписать все условия и комбинации на бумагу, чтобы проще было представить, все ли мы учли и как скомбинировать данные.

b)Упрощение происходит путем исключения из таблицы невозможных или одинаковых комбинаций. Так, например, если тестируем форму редактирования/удаления, на которой расположен radiobutton(с двумя значениями - “modify” и “delete”), то он не может быть одновременно оба этих значения. Подобные комбинации следует исключать.

1. Можно получить слишком большие таблицы. Так, для бинарных значений(true/false), получим уже 2n решений. Слишком большие таблицы необходимо разбить на более мелкие, чтобы удалить избыточность.
2. Трудно рассмотреть всевозможные альтернативы. Система не должна содержать слишком много входов, в противном случае число комбинаций станет неуправляемым. Лучше большое количество условий разделить на подмножества и проверять каждое подмножество
3. Необходимо знать, какие условия важны для тестирования. Для этого необходимо знание предметной области.

**Ответы на вопросы по технике:**

1. Decision Table Tasting лучше всего работает, когда спецификация или требования написаны с использованием if-then-else логики. DT великолепно подходят для упорядочения сложных бизнес-процессов, которые должны быть реализованы в продукте.
2. Если имеем большое количество кейсов, то не всегда возможно успеть проверить все тест-кейсы. В таком случае следует расставить приоритеты и проверять наиболее важные для нас комбинации.