Техники тест-дизайна

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.

Основная цель тест-дизайна — *обеспечить покрытие всего функционала тестируемого продукта тестами*, при этом тестов должно быть минимально достаточно.

Техники тест-дизайна — это *рекомендации, советы и правила, по которым стоит разрабатывать тест* для проведения тестирования продукта. Это не образцы тестов, а только рекомендации к применению.

*Правильным будет считаться тот набор тестов, который за меньшее количество проверок обеспечит более полное покрытие тестами.

Тестирование Классами Эквивалентности (Equivalence Class Testing)

Тестовые данные разбиваются на определенные классы допустимых значений. В рамках каждого класса выполнение теста с любым значением тестовых данных приводит к эквивалентному результату. После определения классов необходимо выполнить хотя бы один тест в каждом классе.

Продемонстрируем это на конкретном примере. Представим, что мы тестируем модуль для HR, который определяет возможность принятия на работу кандидата в зависимости от его возраста.

Установлены следующие условия отбора:

1. при возрасте от 0 до 16 лет — не нанимать;
2. при возрасте от 16 до 18 лет — можно нанять только на part time;
3. при возрасте от 18 до 55 лет — можно нанять на full time;
4. при возрасте от 55 до 99 лет — не нанимать.

Стоит ли проверять каждое значение? Более разумным видится тестирование диапазона каждого условия. Собственно, это и есть наши классы эквивалентности:

1. класс эквивалентности NO: 0-16;
2. класс эквивалентности PART: 17-18;

3. класс эквивалентности FULL: 19-55;
4. класс эквивалентности NO: 56-99.

Как уже было указано, после определения классов эквивалентности мы должны выполнить тест с любым значением из диапазона класса. Таким образом, у нас остается только 4 позитивных тест-кейса вместо первоначальных 100 (0-99), например:

1. 10 — не нанимать;
2. 17 — нанимать на неполный день;
3. 40 — нанимать на полный день;
4. 80 — не нанимать.

Тестирование Граничных Значений (Boundary Value Testing)

Эта техника основана на том факте, что одним из самых слабых мест любого программного продукта является область граничных значений. Для начала выбираются диапазоны значений — как правило, это классы эквивалентности. Затем определяются границы диапазонов. На каждую из границ создается 3 тест-кейса: первый проверяет значение границы, второй — значение ниже границы, третий — значение выше границы.

Нужно помнить, что «выше» и «ниже» — понятия относительные. Например, если мы говорим о границе 6$, то значение «ниже» будет 5$, а значение «выше» — 7$. Если речь идет о границе 6.00$, то значение «ниже» будет 5.99$, а значение «выше» — 6.01$. Не исключено, что значение «ниже» или «выше» границы может быть другим классом эквивалентности, уже охваченным нами. В этом случае нет смысла создавать дубликаты тест-кейсов.

Вернемся к примеру, рассмотренному нами в технике классов эквивалентности:

1. при возрасте от 0 до 16 лет — не нанимать;
2. при возрасте от 16 до 18 лет — можно нанять только на part time;
3. при возрасте от 18 до 55 лет — можно нанять на full time;
4. при возрасте от 55 до 99 лет — не нанимать.

Представим, что соответствующий код выглядит так:

```
If (applicantAge >= 0 && applicantAge <=16)
hireStatus="NO";
If (applicantAge >= 16 && applicantAge <=17)
hireStatus="PART";
If (applicantAge >= 18 && applicantAge <=54)
hireStatus="FULL";
If (applicantAge >= 55 && applicantAge <=99)
hireStatus="NO";
```

При предварительной оценке границ диапазонов требований сразу видна ошибка — диапазоны пересекаются, накладываются. По сути, условие должно быть записано по-другому:

при возрасте от 0 до 15 лет — не нанимать;

при возрасте от 16 до 17 лет — можно нанять только на part time;

при возрасте от 18 до 54 лет — можно нанять на full time;

при возрасте от 55 до 99 лет — не нанимать.

В итоге корректный код должен выглядеть следующим образом:

```
If (applicantAge >= 0 && applicantAge <=15)
hireStatus="NO";
If (applicantAge >= 16 && applicantAge <=17)
hireStatus="PART";
If (applicantAge >= 18 && applicantAge <=54)
hireStatus="FULL";
If (applicantAge >= 55 && applicantAge <=99)
hireStatus="NO";
```

Таким образом, набор значений, для которых будут составлены тесты, будет выглядеть так:

{-1, 0, 1}, {15, 16, 17}, {17, 18, 19}, {54, 55, 56}, {98, 99, 100}.

Тестирование переходов состояний (state transition testing)

Тестирование переходов состояний - разработка тестов методом черного ящика, в котором сценарии тестирования строятся на основе выполнения корректных и некорректных переходов состояний.

Чтобы понять принцип данной техники предлагаю сразу же перейти к примеру составления диаграммы состояний по требованиям.

Требование 1: сразу после получения сообщений с почтового сервера они помечены как непрочитанные.

Требование 2: когда пользователь кликает на сообщение, чтобы открылось его содержание, сообщение становится прочитанным.

Требование 3: пользователь может пометить ранее прочитанные сообщения как непрочитанные.

Требование 4: пользователь может удалить сообщения (как прочитанные, так и непрочитанные), вернуть сообщения после удаления нельзя.

По этим требованиям можно получить такую диаграмму для сообщений:

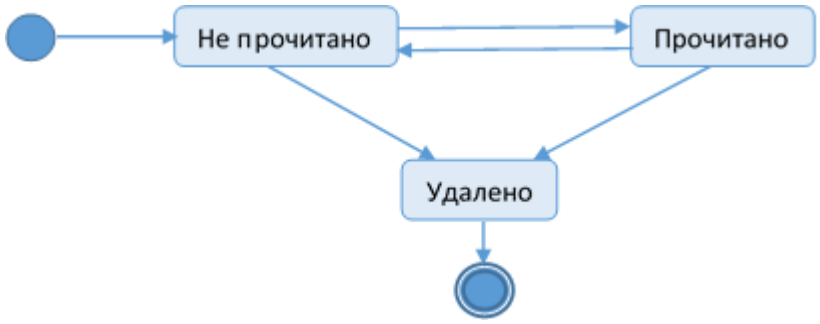

Рисунок 1. Диаграмма переходов состояний

Простые позитивные тесты по диаграммам состояний представляют собой проверку валидных переходов между состояниями, т.е. возможность выполнить действия, обозначенные стрелками. Негативные тесты — проверка невозможности выполнения никаких других действий. Таким образом, легко посчитать количество необходимых позитивных тестов — количество стрелок на диаграмме. Количество негативных тестов тоже несложно подсчитать:

$$T^- = N^2 - T^+,$$

где T^- — количество негативных тестов, N — количество состояний, T^+ — количество позитивных тестов.

Таким образом, для этой диаграммы будет четыре простых позитивных теста и пять негативных. Если проектировать негативные тесты для почтового веб-клиента, они будут выглядеть приблизительно так:

Тест-кейс «Не прочитано – Не прочитано»

Шаги:

1. Открыть список сообщений на двух вкладках браузера.
2. Найти прочитанное сообщение и пометить его непрочитанным на первой вкладке.

3. Не обновляя вторую вкладку, пометить непрочитанным сообщение, выбранное на Шаге 2.

Ожидаемый результат:

Сообщение на обеих вкладках помечено как непрочитанное, на Шаге 3 не возникает сообщения об ошибке.

Тест-кейс «Удалено – Удалено»

Шаги:

1. Открыть список сообщений на двух вкладках браузера.
2. Найти произвольное сообщение и удалить его на первой вкладке.
3. Не обновляя вторую вкладку, удалить сообщение, выбранное на Шаге 2.

Ожидаемый результат:

Сообщение на обеих вкладках удалено (пропало из списка сообщений), на Шаге 3 не возникает сообщения об ошибке.

Пример позитивного тест-кейса представлен ниже.

Тест-кейс «Не прочитано – Прочитано»

Шаги:

1. Открыть список непрочитанных сообщений.
2. Открыть произвольное сообщение.

Ожидаемый результат:

Сообщение помечено как прочитанное.