Тест дизайн – этап процесса тестирования ПО, на котором проектируются и создаются ТК с определенными ранее критериями качества и целями тестирования

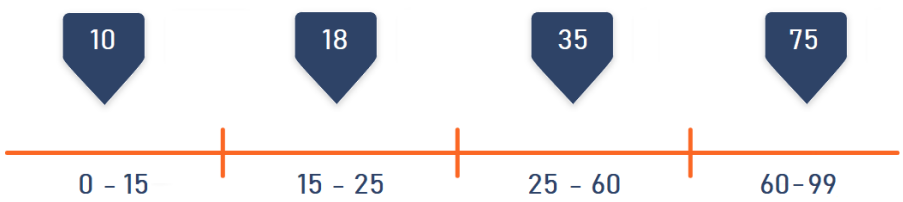
Классы эквивалентности

Метод эквивалентного разбиения позволяет минимизировать число тестов, не создавая сценарий для каждого возможного значения, а выбрав только одно значение из целого класса и приняв за аксиому, что для всех значений в данной группе результат будет аналогичным.

Например, тестируем функциональность приложения, позволяющего покупать авиа- и железнодорожные билеты онлайн. Стоимость билета будет зависеть от возраста пассажира, т.к. дети, студенты и пенсионеры относятся ко льготным категориям.

У нас есть четыре возрастных группы: младше 15 лет, от 15 до 25 лет, старше 25 и младше 60 лет и люди старше 60. При этом, в поле для ввода возраста помещается всего два символа, поэтому указать возраст более 99 лет технически невозможно.

Делаем по одному тесту для каждой возрастной группы (скажем, 10, 18, 35 и 75 лет) и один для случая, если возраст человека превышает 99 лет. Последний тест на практике невыполним (поскольку в поле возраста невозможно ввести более двух знаков), и все же не следует забывать об этой проверке.



Граничные значения

Техника граничных значений основана на предположении, что большинство ошибок может возникнуть на границах эквивалентных классов. Она тесно связана с вышеописанной техникой эквивалентного разбиения, из-за чего часто используется с ней в паре. Тогда для примера из предыдущего пункта границами будут являться значения 0, 15, 25, 60 и 99. Граничными значениями будут 0, 1, 14, 15, 16, 24, 25, 26, 59, 60, 61, 98, 99, 100.
Часто сложности возникают, если возрастные категории указаны «внахлест», например, 0-12, 12-25 лет и т.д.

Таблица принятия решений

Другое название метода – матрица принятия решений. Эта техника подходит для более сложных систем, например – двухфакторной аутентификации. Предположим, чтобы войти в систему, пользователю нужно ввести сначала логин и пароль, а затем еще подтвердить свою личность присланным в смс кодом.

Какие возможны сценарии:

1. Правильный логин и правильный пароль
2. Правильный логин, неправильный пароль
3. Неправильный логин, правильный пароль
4. Неправильный логин, неправильный пароль

Первый из этих сценариев сопровождается либо правильным, либо неправильным вводом смс-кода, итого у нас получается 5 тестов. При этом только один из сценариев приведет к положительному результату (пользователь успешно авторизуется), а остальные закончатся неудачей.

Однако, может быть так, что система выдает разные сообщения в зависимости от того, на каком этапе была допущена ошибка, скажем: invalid login, invalid password. Соответственно, групп потребуется больше, а таблица станет обширнее.

Этот метод хорош тем, что он показывает сразу все возможные сценарии в форме.

Условия	Значения	Правила				
Логин	Верный, Неверный	Верный	Верный	Верный	Неверный	Неверный
Пароль	Верный, Неверный	Верный	Верный	Неверный	Неверный	Верный
Код	Верный, Неверный, Не пришел	Верный	Неверный	Не пришел	Не пришел	Не пришел
Действия	Пользователь вошел на сайт Пользователь не авторизован	Пользователь вошел на сайт	Пользователь не авторизован	Пользователь не авторизован	Пользователь не авторизован	Пользователь не авторизован

Условия	Значения	Правила			
Логин	Верный, Неверный	Верный	Неверный	Верный, Неверный	Верный
Пароль	Верный, Неверный	Верный	Верный, Неверный	Неверный	Верный
Код	Верный, Неверный, Не пришел	Верный	Верный, Неверный, Не пришел	Верный, Неверный, Не пришел	Неверный
Действия	Пользователь вошел на сайт Пользователь не авторизован	Пользователь вошел на сайт	Пользователь не авторизован	Пользователь не авторизован	Пользователь не авторизован

Попарное тестирование

Суть этого метода, также известного как pairwise testing, в том, что каждое значение каждого проверяемого параметра должно быть протестировано на взаимодействие с каждым значением всех остальных параметров. После составления такой матрицы мы убираем тесты, которые дублируют друг друга, оставляя максимальное покрытие при минимальном необходимом наборе сценариев.

Попарное тестирование позволяет обнаружить максимум ошибок без избыточных проверок.

Для Parwise достаточно, чтобы каждое значение всех параметров хотя бы единожды сочеталось с другими значениями остальных параметров. Таким образом, матрицу можно значительно сократить. Например:

№	Браузер	Операционная система	Язык
1	Opera	Windows	RU
2	Google	Linux	RU
3	Opera	Linux	EN
4	Google	Windows	EN

При составлении матрицы принятия решений для двух браузеров, двух ОС и двух языков было бы нужно 8 сценариев. При попарном тестировании достаточно четырех.

Все это можно просчитать и вручную, но не обязательно – гораздо удобнее автоматизировать процесс. Для этого существует программа попарного независимого комбинированного тестирования – Pairwise Independent Combinatorial Testing (PICT). Для проведения тестирования создается текстовый файл с перечислением и их возможных значений, а затем запускает PICT через командную строку. Скомбинированные тесты отображаются в виде таблицы в самой консоли. Так же результаты по желанию можно выгрузить в файл Excel.

Google Chrome	Windows	RU
Google Chrome	Windows	EN
Google Chrome	Linux	RU
Google Chrome	Linux	EN
Opera	Windows	RU
Opera	Windows	EN
Opera	Linux	RU
Opera	Linux	EN

Google Chrome	Windows	RU
Google Chrome	Linux	EN
Opera	Windows	EN
Opera	Linux	RU

Причина и следствие

Простая проверка базовых действий и их результата. Например, если нажать крестик в правом верхнем углу окна (причина), оно закроется (следствие), и т.д. Этот метод позволяет проверить все возможности системы, а также обнаружить баги и улучшить техническую документацию продукта.

Примерный алгоритм использования техники:

1. Выделяем причины и следствия в спецификациях
2. Связываем причины и следствия
3. Учитываем «невозможные» сочетания причин и следствий
4. Составляем «таблицу решений», где в каждом столбце указана комбинация входов и выходов, т.е. каждый столбец – это готовый тестовый сценарий
5. Расставляем приоритеты

Эта техника помогает:

- определить минимальное количество тестов для нахождения максимума ошибок

- выяснить все причины и следствия – таким образом, мы убедимся, что на любые манипуляции с системой у системы будет ответ

- найти возможные недочеты в логике описания приложения (что, в свою очередь, поможет улучшить документацию)

Например, протестируем приложение типа “записная книжка”. После ввода всех данных нового контакта и нажатия кнопки Создать (причина) приложение должно автоматически создать карточку с номером телефона, фотографией и ФИО человека (следствие). Тесты покажут, можно ли оставлять одно или несколько полей пустыми, распознает ли система кириллицу, латиницу или оба алфавита, а также другие параметры.

Предугадывание ошибок

Используя свои знания о системе, можно «предугадать», при каких входных условиях есть риск ошибок. Для этого важно иметь опыт, хорошо знать продукт и уметь выстроить коммуникации с коллегами.

Например, в спецификации указано, что поле должно принимать код из четырех цифр. В числе возможных тестов:

- Что произойдет, если не ввести код?
- Что произойдет, если не ввести спецсимволы?
- Что произойдет, если ввести не цифры, а другие символы?
- Что произойдет, если ввести не четыре цифры, а другое количество?

Преимущества:

1. Эта проверка эффективна в качестве дополнения к другим техникам

2. Выявляет тестовые случаи, которые “никогда не должны случиться”

Недостатки:

1. Техника в значительной степени основана на интуиции

2. Необходим опыт в тестировании подобных систем

3. Малое покрытие тестами