1. Что такое Тест-дизайн?

Тест-дизайн – это процесс проектирования и создания тестовых случаев (тест-кейсов) для проверки программного обеспечения. Цель тест-дизайна – обеспечить максимально эффективное покрытие функциональности приложения, выявить возможные дефекты и убедиться в соответствии продукта требованиям. Хороший тест-дизайн позволяет создать тесты, которые с высокой вероятностью найдут ошибки и при этом потребуют минимальных усилий на их выполнение.

2. Что такое классы эквивалентности? Приведите пример

Классы эквивалентности – это техника тест-дизайна, которая разделяет входные данные на группы, в пределах которых ожидается, что система будет вести себя одинаково. Считается, что если один тестовый случай из класса эквивалентности проходит, то и все остальные тестовые случаи из этого класса тоже пройдут (и наоборот, если один падает, то и остальные упадут).

3. Что представляют из себя граничные значения и как их применение способно облегчить процесс тестирования?

Граничные значения – это значения, находящиеся на границах классов эквивалентности. То есть это минимальные и максимальные значения, а также значения непосредственно рядом с ними. Например, если у нас есть диапазон чисел от 1 до 100, то граничные значения будут: 0, 1, 2, 99, 100, 101.

4. Для чего используют технику попарного тестирования и что это такое?

Попарное тестирование (Pairwise Testing, All-Pairs Testing) используется для сокращения количества тестовых случаев при проверке систем с большим количеством входных параметров. Полное тестирование всех комбинаций входных параметров может быть очень затратным по времени и ресурсам. Попарное тестирование позволяет проверить все возможные \*пары\* значений входных параметров, что часто выявляет значительное количество дефектов, при этом существенно сокращая количество тестовых случаев по сравнению с полным перебором.

Попарное тестирование – это техника тест-дизайна, которая создает тестовые случаи таким образом, чтобы каждая возможная пара значений любых двух входных параметров была протестирована хотя бы один раз.

Пример:

Предположим, у нас есть система с тремя входными параметрами:

• A: Значения A1, A2

• B: Значения B1, B2

• C: Значения C1, C2

Полное тестирование потребовало бы 2 × 2 × 2 = 8 тестовых случаев:

1. A1, B1, C1

2. A1, B1, C2

3. A1, B2, C1

4. A1, B2, C2

5. A2, B1, C1

6. A2, B1, C2

7. A2, B2, C1

8. A2, B2, C2

С помощью попарного тестирования можно сократить количество тестовых случаев. Один из возможных наборов тестов, покрывающих все пары значений, может выглядеть так:

1. A1, B1, C1

2. A1, B2, C2

3. A2, B1, C2

4. A2, B2, C1

Здесь мы протестировали все возможные пары:

• (A1, B1), (A1, B2), (A2, B1), (A2, B2)

• (A1, C1), (A1, C2), (A2, C1), (A2, C2)

• (B1, C1), (B1, C2), (B2, C1), (B2, C2)

Для создания наборов тестов для попарного тестирования часто используют специальные инструменты (например, AllPairs, PICT).

В заключение, понимание и применение техник тест-дизайна критически важно для обеспечения высокого качества программного обеспечения. Выбор конкретной техники зависит от специфики проекта, доступных ресурсов и уровня риска.