**Доменный анализ** – техника тест-дизайна, в основе которой лежит работа с доменами. Домен – совокупность элементов, которая имеет свои границы (за пределами которых свойства элементов отличаются от свойств элементов в границах).

**Два теста считаются эквивалентными**, если:

● они тестируют подобные друг другу сущности (функцию, модуль, часть системы),

● они приводят к одному и тому же результату (выполнив два любых теста из одного класса эквивалентности, получим один и тот же результат).

**Если два теста эквивалентны**, то:

● если первый тест ловит ошибку, то второй тоже её поймает,

● если первый тест пропускает ошибку, то и второй её пропустит.

При этом классы эквивалентности могут быть линейными и нелинейными. Линейные классы эквивалентности можно расположить на одной числовой прямой или в виде упорядоченных множеств. Нелинейные классы – это все остальные классы, которые нельзя представить в виде упорядоченных множеств.

**Стратегии тестирования домена**

**Random Testing** – выбор значений для тестирования класса случайным образом из области допустимых значений класса. Например, выбрав класс «день недели», можно выбрать любое из 7 допустимых значений (понедельник, вторник, среда, четверг, пятница, суббота, воскресенье).

**Метод пропорционального разбиения** – по сути, продолжение метода случайного тестирования, с той лишь разницей, что количество тестовых значений для классов определяется:

* стоимостью дефекта в классе (насколько опасно иметь дефект в том или ином классе). Чем сложнее система или её функция, тем большее количество тестов необходимо для её проверки;
* величиной класса (чем больше область допустимых значений класса, тем больше тестовых значений потребуется для тестирования). Например, для тестирования класса «дни недели» достаточно проверки одного значения, а для тестирования класса «Фамилии» – нет, и нужно подбирать категории значений. Например, это могут быть категория простых фамилий (Иванов) и категория двойных фамилий (Петрова-Моисеенко);
* классы разбиваются на подклассы, и из каждого подкласса выбирается по одному значению. Например, выбран класс «Буквы». Буквы могут быть в двух регистрах: строчные и заглавные. Таким образом, разбиваем множество букв на 2 подкласса – строчные и заглавные, затем выбираем случайным образом представителя каждого подкласса и проверяем эти значения.

**Risk-based** – группа техник тестирования, которые фокусируются на выборе значений в потенциально уязвимых местах

Например, для класса «дата» потенциально уязвимыми местами могут быть значения последних дней месяца, поскольку в разных месяцах различное количество дней. Соответственно, тестовыми значениями должны быть последние дни месяца

А если класс «Фамилия», то узким местом может быть двухсоставная фамилия, такая как Зачеши-Гриву.

**Анализ граничных значений (Boundary Value Testing)** – популярная техника доменного анализа, при которой определяются границы классов эквивалентности и проверяются элементы, близкие к границе.

Эта техника основана на рисках и начинается с идеи о том, что программа может сломаться в области граничных значений, потому что:

* давно замечено, что при разработке большое число проблем возникает на границах входных переменных;
* даже если классы эквивалентности найдены правильно, то граничные значения могут быть ошибочно отнесены к другому классу.

Анализ граничных значений можно разделить на:

1. **Специальные значения для тестирования** – выбор дополнительных значений исходя из специфики предметной области (Бизнес требования). Например, есть функция, которая возвращает дату следующего дня. Здесь необходимо проверять последнюю дату короткого и длинного месяцев, последнюю дату года и т. п.
2. **Steeplechase Testing** – ещё одна разновидность работы с граничными значениями; кроме границ, накладываемых бизнес-требованиями, нужно проверять технические границы. Например, если число должно содержать не более 3 символов, то нужно также проверить ввод числа, содержащего 4 символа.
3. **Анализ границ выходных значений (Robustness Testing)** – техника, при которой исследуются выходные значения, на основании чего выбираются входные значения для тестов. Например, есть функция c=a\*b, где a, b, c – вещественные числа типа float. Если a=max float, b=10, то с=10\*max float, то есть это потенциальный дефект.