Инспекция кода. Разбиение на эквивалентные части

Эквивалентное разбиение – это метод тестирования «черного ящика». Идея тестирования по методу разбиения классов эквивалентности состоит в том, чтобы исключить набор входных данных, которые заставляют систему вести себя одинаково и давать одинаковый результат при тестировании программы.

Эквивалентность. Разбиение на разделы является обычной методикой тестирования с использованием черного ящика и нацелено на сокращение числа избыточных тестовых сценариев, устраняя те, которые генерируют один и тот же результат, и не обязательно обнаруживают дефекты в функциональности программы.

Поскольку целью тестирования является обнаружение дефектов, то успешный тестовый сценарий – тот, который обнаруживает дефект.

Процесс тестирования по методу эквивалентного разбиения включает в себя идентификацию набора данных как условий ввода, которые дают одинаковый результат при выполнении программы и классифицируют их как набор эквивалентных данных (поскольку они заставляют программу вести себя одинаково и генерировать тот же результат) и происходит разбиение их на группы из другого эквивалентного набора данных.

Рассмотрим пример ниже:

Можно ожидать, что приложение, принимающее целочисленные значения (целочисленные числа) от -10,000 до +10,000, сможет обрабатывать отрицательные целые числа, ноль и положительные целые числа. Таким образом, набор входных значений можно разделить на три раздела:

От -10 000 до -1, 0 и от 1 до 10000

Более того, ожидается, что система будет вести себя одинаково для значений внутри каждого раздела. То есть способ, которым система обрабатывает -6391, будет такой же, как -9. Аналогично, положительные целые числа 5 и 3567 будут обрабатываться системой одинаково. В этом конкретном примере значение 0 является разделом с одним значением. Обычно хорошей практикой является специальный сценарий с нулевым числом.

Важно отметить, что эта методика применима не только к цифрам. Этот метод может применяться к любому набору данных, который можно рассматривать как эквивалент. Например. Приложение, которое читает изображения только из трех типов: .jpeg, .gif и .png, тогда можно идентифицировать три набора допустимых эквивалентных классов.

Изображение с расширением .jpeg

Изображение с расширением .gif

Изображение с расширением .png.

Теперь, открыв файл .jpeg, который является образом луны, ПО будет вести себя так же, как файл с изображением собаки. Поэтому, открывая только один файл типа .jpeg, хватит одного тестового сценария. Предполагается, что система будет вести себя одинаково для всех jpeg-файлов.

Тот же сценарий применяется к файлам с файлами .gif и .png. Аналогично, если приложение не может открывать файлы, отличные от разрешенных и допустимых типов, то при попытке открыть текстовый документ результат будет таким же, как при попытке открыть таблицу Excel или текстовый файл. (Ожидается, что приложение было хорошо разработано, чтобы справиться с другими типами файлов и генерирует соответствующее сообщение при попытке открыть неприемлемые типы файлов).

Они будут классифицированы как набор недействительных эквивалентных данных. Пытаться открыть приложение с недопустимыми или недействительными типами файлов – пример отрицательного тестирования, что полезно в сочетании с тестированием по методу эквивалентного разбиения, который разбивает набор эквивалентных допустимых и достоверных данных.

Инспекция кода

Инспекция кода (code inspection) или просмотр кода (code review) — это систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки. Целью просмотра является улучшение качества программного продукта и совершенствование навыков разработчика.
В процессе инспекции могут быть найдены и устранены такие проблемы, как ошибки в форматировании строк, состояние гонки (race condition), утечка памяти (memory leak) и переполнение буфера (buffer overflow), что улучшает безопасность программного продукта. Системы контроля версий дают возможность проведения совместной инспекции кода. Кроме того, существуют специальные инструментальные средства для совместной инспекции кода.
Программное обеспечение для автоматизированной инспекции кода упрощает задачу просмотра больших кусков кода, систематически сканируя его на предмет обнаружения наиболее известных уязвимостей.