Методы проектирования тестов

Категории методов проектирования тестов

Цель метода проектирования тестов, включая перечисленные ниже, заключается в определении тестовых условий, тестовых сценариев и тестовых данных.

Выбор метода проектирования тестов

Выбор конкретного метода проектирования тестов зависит от множества факторов, включая:

- Тип системы или компонента
- Сложность системы или компонента
- Нормативные документы
- Требования пользователей или контракта
- Уровни рисков
- Типы рисков
- Задачи тестирования
- Доступную документацию
- Знания и опыт тестировщиков
- Доступные инструменты
- Ресурсы времени и бюджет
- Жизненный цикл разработки
- Ожидаемое использование системы
- Прошлый опыт использования методов проектирования тестов для компонента или системы
- Ожидаемые типы дефектов компонента или системы

Некоторые методы применимы к конкретным ситуациям и уровням тестирования, другие применимы на всех уровнях. Обычно, тестировщики используют комбинации различных методов в процессе создания тестовых сценариев для достижения наилучших результатов. Применение тех или иных методов на этапе анализа, дизайна и реализации тестов может варьироваться от неформального (минимум либо полное отсутствие документации) до строго

формального. Уровень формальности зависит от контекста, включая зрелость процессов разработки и тестирования, временных ограничений, требований безопасности и требований регуляторов, знания и опыта вовлеченных людей, а также используемой модели жизненного цикла.

Категории методов проектирования тестов и их характеристики.

Здесь и далее методы проектирования тестов делятся на методы черного ящика, методы белого ящика и методы, основанные на опыте. Методы черного ящика (поведенческие, или методы, основанные на поведении) основываются на анализе соответствующего базиса тестирования (формальных требований, спецификаций, сценариев использования, пользовательских историй или бизнес-процессов). Эти методы применимы как для функционального, так и для нефункционального тестирования. Методы черного ящика сосредотачиваются на связи входных данных и выходных результатов объекта тестирования, а не на его внутренней структуре. Методы белого ящика (структурные, или методы, основанные на структуре) основаны на анализе архитектуры, детального проектирования, внутренней структуры или кода компонента либо системы. В отличие от методов черного ящика методы белого ящика сосредотачиваются на структуре и обработке внутри объекта тестирования. Методы, основанные на опыте, используют опыт разработчиков, тестировщиков и пользователей для проектирования, реализации и выполнения тестов. Их часто совмещают с методами черного и белого ящиков. Общие характеристики методов черного ящика:

● Тестовые условия, тестовые сценарии и тестовые данные получаются из базиса тестирования, который может включать в себя требования, спецификации, сценарии использования и пользовательские истории

● Тестовые сценарии могут использоваться для определения несоответствий и отклонений между требованиями и реализацией

● Измерение покрытия основано на элементах базиса тестирования и методе проектирования, применяемом к базису тестирования Общие характеристики методов белого ящика:

● Тестовые условия, тестовые сценарии и тестовые данные получаются из базиса тестирования, который может включать в себя код, архитектуру, детальную архитектуру или любой другой источник информации о структуре программного обеспечения

● Измерение покрытия основано на элементах структуры (коде, интерфейсах и т.д.)

● Спецификации используются как источник дополнительной информации для определения ожидаемых результатов тестовых сценариев Общие характеристики методов, основанных на опыте:

● Тестовые условия, тестовые сценарии и тестовые данные получаются из базиса тестирования, который может включать в себя знания и опыт тестировщиков, разработчиков, пользователей и других заинтересованных лиц Эти знания и опыт включают в себя возможное использование ПО, его окружение, возможные дефекты и их распределение. Международный стандарт ИСО (ISO/IEC/IEEE 29119-4) содержит описание методов проектирования тестов и соответствующие им методы измерения покрытия (см. также [Craig 2002] и [Copeland 2004] для описания дополнительных методов).

Методы черного ящика

Эквивалентное разбиение

Эквивалентное разбиение делит данные на группы (классы эквивалентности), которые, как предполагается, обрабатываются схожим образом [Kaner 2013], [Jorgensen 2014]. Области эквивалентности могут быть как для правильных, или позитивных, так и неправильных, или негативных, значений.

● Позитивные значения – это значения, которые должны быть приняты. Класс, содержащий позитивные значения, называется «действительный класс эквивалентности».

● Негативные значения – значения, которые должны быть отвергнуты. Класс, содержащий негативные значения, называется «недействительный класс эквивалентности».

● Классы могут быть определены для любых данных, относящихся к объекту тестирования, включая: входные данные, выходные данные, внутренние данные, данные, связанные со временем (например, до или после события), интерфейсные параметры (например, в случае интеграционного тестирования компонентов).

● Любой класс может быть при необходимости разделен на подклассы

● Каждое значение должно принадлежать только одному классу эквивалентности

● Во избежание маскирования дефектов негативные классы эквивалентности в тестовых сценариях следует использовать по отдельности, то есть избегать комбинаций одних негативных классов с другими. Дефекты могут быть маскированы, если при наличии нескольких дефектов обнаруживается только один из них. Для достижения 100% покрытия с помощью этого метода, тестовые сценарии должны покрывать все позитивные и негативные классы, проверяя хотя бы одно значение из каждого класса. Покрытие вычисляется как отношение количества тестируемых классов к общему числу классов. Метод эквивалентного разбиения применим на всех уровнях тестирования.

Анализ граничных значений

Метод анализа граничных значений является продолжением метода эквивалентного разбиения, но может быть применим, только если классы состоят из упорядоченных числовых значений. Максимальное и минимальное значение класса являются его границами [Beizer 1990]. Для примера, предположим, что некоторое поле ввода принимает положительное целое число от 0 до 9; ввод осуществляется через клавиатуру, поэтому нечисловые входные значения исключены. Допустимы значения от 1 до 5 включительно. Таким образом, можно выделить три области эквивалентности: негативная

(слишком маленькие), позитивная, негативная (слишком большие). Для позитивной области эквивалентности значения 1 и 5 будут граничными. Для области с негативными большими значениями границами будут значения 6 и 9. Для области с негативными малыми значениями будет только одна граница – 0, поскольку эта область состоит из одного элемента. В рассмотренном выше примере мы можем определить по два граничных значения на каждую из границ. Граница между негативными малыми и позитивными значениями дает нам тестовые значения 0 и 1. Граница между позитивными и негативными большими значениями дает нам тестовые значения 5 и 6. Вариация данного метода определяет три граничных значения на каждую из границ: значения перед, на и сразу после границы. Для рассматриваемого примера применение этого метода даст следующие тестовые значения: для нижней границы – 0,1,2; для верхней границы – 4,5,6 [Jorgensen 2014]. Некорректное поведение более вероятно на границах класса, чем внутри класса. Важно помнить, что специфицированные и реализованные границы могут быть смещены вверх или вниз относительно истинных значений, они могут быть пропущены или дополнены новыми границами. Анализ и тестирование граничных значений может обнаружить большинство подобных дефектов, вынуждая программное обеспечение демонстрировать поведение, относящееся к области эквивалентности, отличной от той, к которой должна относиться граничная точка. Анализ граничных значений может использоваться на любом уровне тестирования. Данный метод применяется при тестировании требований, в которых присутствуют диапазоны значений (включая даты и время). Покрытие вычисляется как отношение числа тестируемых граничных значений к общему числу граничных значений и чаще всего выражается в процентах.

Тестирование с помощью таблицы альтернатив

Комбинаторные методы полезны при тестировании требований, содержащих условия, которые дают разные результаты в зависимости от комбинаций. Одним из таких методов является тестирование с помощью

таблицы альтернатив. Таблицы альтернатив – хороший способ записи сложных бизнес-правил, которые должны быть реализованы в системе. В процессе создания таблицы, тестировщик определяет условия (входы) и результирующие действия системы (выходы). Пары условий и действий образуют строки таблицы, при этом условия указываются сверху, а действия – снизу. Каждый столбец представляет собой бизнес-правило с уникальной комбинацией условий и действий, связанных с этим правилом. Значения условий часто отображаются в виде логических (истина или ложь) или дискретных (красный, синий, зеленый) значений, но могут быть также в виде чисел или числовых диапазонов. В одной таблице могут сочетаться значения разных типов. Обозначения, используемые для таблиц альтернатив: Условия:

● Y означает, что условие истинно (используется также обозначение T или 1) ● N означает, что условие ложно (используется также обозначение F или 0)

● – означает, что значение условия может быть любым (используется также обозначение N/A) Действия:

● X означает, что действие должно быть выполнено (используется также обозначение Y, T или 1)

● Пустое поле означает, что действие не должно выполняться (используется также обозначение N, F или 0) Полная таблица альтернатив содержит по столбцу на каждую комбинацию условий. Таблицу можно сократить, убрав столбцы, которые содержат несуществующие комбинации или комбинации, не влияющие на результат. Более подробная информация размещена в программе подготовки продвинутого уровня (ISTQB-ATA Продвинутый уровень, Тест-аналитик). Стандарт покрытия для таблиц альтернатив подразумевает наличие хотя бы одного теста для каждого столбца таблицы. Обычно это подразумевает покрытие всех комбинаций условий. Покрытие измеряется как отношение количества правил, проверенных хотя бы одним тестом, к общему числу правил, выраженное в процентах. Преимущество метода заключается в том, что он выявляет комбинации условий, которые могли быть не проверены при тестировании. Метод

помогает определить несоответствия в требованиях, может быть применен во всех ситуациях и на любом уровне, где поведение программного обеспечения зависит от комбинации условий.

Тестирование с помощью таблицы переходов

В зависимости от прошлых условий и состояния система может вести себя по-разному. Описать прошлое системы можно с помощью концепции состояний. Диаграмма состояний и переходов показывает начальное и конечное состояния системы, а также описывает переходы между состояниями. Каждый переход вызывается событием (например, вводом данных пользователем). Если одно и то же событие может привести к разным переходам, выбор перехода может задаваться контрольным условием. Смена состояния может завершаться выполнением какого-либо действия (вывод результатов или сообщения об ошибке и т.д.). Таблица переходов представляет собой все возможные комбинации начальных и конечных состояний, включая действительные и недействительные переходы, инициирующие события, защитные условия и результирующие действия. Диаграммы состояний и переходов обычно, показывают только действительные переходы и исключают недействительные переходы. Тесты создаются для покрытия типичной последовательности состояний, покрытия каждого возможного состояния, покрытия каждого возможного перехода, проверки специфических последовательностей переходов, или для проверки недействительных переходов. Тестирование с помощью таблицы переходов наиболее распространено в сфере встроенного ПО и при тестировании программных меню. Метод также подходит для моделирования бизнессценариев, имеющих конкретные состояния, или для тестирования переходов по экранным формам. Концепция состояний абстрактна, она может отражать несколько строк кода или целый бизнес-процесс. Покрытие измеряется как отношение числа протестированных состояний или переходов к общему числу состояний или переходов в объекте тестирования, выраженное в процентах. Более подробная информация размещена в

программе подготовки продвинутого уровня (ISTQB-ATA Продвинутый уровень, Тест-аналитик).

Тестирование с помощью сценариев использования

Тесты можно разработать на основе сценариев использования, которые представляют собой способ описания взаимодействий с программными объектами. В сценариях использования всегда присутствуют участники (пользователи, внешние устройства и компоненты) и субъекты (компоненты или системы, к которым применяется сценарий использования). Каждый сценарий использования описывает поведение субъекта при взаимодействии с участником (UML 2.5.1 2017). Сценарий использования может быть описан взаимодействиями и активностями, предусловиями и постусловиями или естественным языком, если это возможно. Взаимодействия между участниками и субъектом могут приводить к изменению состояния субъекта; они могут изображаться графически, с помощью диаграмм или моделей бизнес-процессов. Сценарий использования может отражать различные варианты поведения, включая исключительные ситуации и обработку ошибок (реакцию системы и восстановление из-за ошибок программирования, приложений и связи, например, в результате чего появляется сообщение об ошибке). Тесты разрабатываются с целью проверки различных вариантов поведения (базового, исключительного, альтернативного, обработки ошибок). Покрытие может выражаться как процент протестированных вариантов поведения к общему числу вариантов. Более подробная информация о покрытии размещена в программе подготовки продвинутого уровня (ISTQB-ATA Продвинутый уровень, Тест-аналитик).

Методы белого ящика

Тестирование белого ящика основывается на внутренней структуре объекта тестирования. Методы могут применяться на всех этапах, однако, методы, рассмотренные ниже, чаще всего используются в модульном тестировании. Существуют и другие методы, используемые в тестировании критичных систем и обеспечивающие более сильное покрытие, но здесь они

не рассматриваются. Более подробная информация об этих методах размещена в программе подготовки продвинутого уровня (ISTQB Продвинутый уровень, Технический тест-аналитик).

Тестирование и покрытие операторов

Тестирование операторов направлено на проверку исполняемых операторов в коде. Покрытие вычисляется как отношение количества операторов, выполненных тестом, к общему числу операторов в тестируемом коде.

Тестирование и покрытие условий

Тестирование условий направлено на проверку логических условий в коде, а также кода, выполняемого в зависимости от исхода условия. Для этого тесты следуют потокам управления, которые выходят из условия (путь для выхода «истина» и для выхода «ложь»; для оператора выбора (CASE) тесты потребуются для всех возможных результатов, включая выход по умолчанию). Покрытие вычисляется как отношение числа исходов условий, проверенных тестом, к общему числу исходов тестируемых условий.

Ценность тестирования операторов и условий

Стопроцентное покрытие операторов означает проверку всех исполняемых инструкций в коде хотя бы один раз, но это не дает уверенности в проверке логики условий. Из двух методов, рассматриваемых здесь, тестирование операторов обеспечивает более слабое покрытие, чем тестирование условий. Стопроцентное покрытие условий проверяет все ветки потока управления, что включает проверку выходов «истина» и «ложь» для условных операторов. Покрытие условий позволяет находить ситуации, в которых исполняемый код может быть пропущен в зависимости от результата условия. Стопроцентное покрытие условий гарантирует стопроцентное покрытие операторов, но не наоборот.

Методы, основанные на опыте

При использовании методов тестирования, основанных на опыте, тесты создаются на основе умения, интуиции и опыта тестировщика. Данные методы могут пригодиться при создании тестов, которые невозможно получить, применяя другие, более системные методы. В зависимости от выбранного подхода и прошлого опыта можно получить очень разные степени покрытия и эффективности тестирования. Измерение покрытия для данных методов может быть затруднительным или вообще невозможным. Наиболее популярные методы рассматриваются в последующих разделах.

Предположение об ошибках

Предположение об ошибках – это способ предотвращения ошибок, дефектов и отказов, основанный на знаниях тестировщика, включающих:

- Историю работы приложения в прошлом
- Наиболее вероятные типы дефектов, допускаемых при разработке
- Типы дефектов, которые были обнаружены в схожих приложениях, Структурированный подход к предположению об ошибках предполагает создание списка всех возможных ошибок, дефектов и отказов с последующей разработкой тестов, направленных на поиск дефектов из этого списка. Списки отказов и дефектов могут быть построены на основе опыта, исторических данных об отказах и ошибках, а также на общих знаниях о причинах отказа программ.

Исследовательское тестирование

Во время исследовательского тестирования неформальные (т.е. не созданные заранее) тестовые сценарии разрабатываются, выполняются, анализируются и оцениваются динамически во время выполнения тестов. Результаты тестирования используются для изучения компонента или системы и последующей разработки тестовых сценариев для непокрытых областей. Исследовательское тестирование может проводиться сессиями, что позволяет структурировать активность. При использовании сессионного подхода, исследовательское тестирование выполняется в определенном

промежутке времени, при этом тестировщик использует концепцию тестирования, содержащую цели, и отмечает выполненные действия и обнаруженные факты. Исследовательское тестирование лучше всего подходит в ситуациях, когда документация недостаточная, либо вовсе отсутствует, в условиях очень сжатых сроков и как дополнение к другим, более формальным, методам тестирования. Исследовательское тестирование относится к реактивным стратегиям тестирования и может включать использование различных методов черного и белого ящиков или методов, основанных на опыте.

Тестирование на основе чек-листов. При тестировании по чек-листам тестировщик проектирует, реализует и выполняет тесты, покрывающие тестовые условия, указанные в чек-листе. В качестве составной части анализа тестировщики могут создавать новые или расширять чек-листы, либо использовать готовые чек-листы, не меняя их. Такие списки могут быть построены на опыте, на исторических данных об ошибках, на информации о приоритетах для пользователей и понимании, как и почему происходят отказы в программе. Чек-листы могут создаваться для поддержки разных видов тестирования, включая функциональное и нефункциональное тестирование. При отсутствии детальных тестовых сценариев, чек-листы помогают определить направления тестирования и увеличивают согласованность тестирования. Поскольку чек-листы содержат общее описание, это снижает повторяемость результатов.