Программная инженерия
Лекция 6. Тестирование

Составитель: Эверстов В.В.

Дата составления: 03/04/2014

Дата модификации: 03/04/2014

Тестирование

- Тестирование представляет собой процесс, демонстрирующий отсутствие ошибок в программе,
- Цель тестирования – показать, что программа корректно исполняет предусмотренные функции,
- Тестирование – это процесс, позволяющий убедиться в том, что программа выполняет свое назначение.

Определения

Тестирование представляет собой процесс, демонстрирующий отсутствие ошибок в программе,

Цель тестирования – показать, что программа корректно исполняет предусмотренные функции,

Тестирование – это процесс, позволяющий убедиться в том, что программа выполняет свое назначение.

Тестирование

Тестирование – это процесс исполнения программы с целью обнаружения ошибок.

Следствия

- **тестирование – процесс деструктивный** (т. е. обратный созидательному, конструктивному).
- Из определения следует так же, **как нужно строить набор тестовых данных и кто должен (а кто не должен) тестировать данную программу**.

Удачные тесты

- Для усиления определения тестирования проанализируем два понятия «удачный» и «неудачный».
- Тестовый прогон, приведший к обнаружению ошибки, нельзя назвать неудачным хотя бы потому, что, это целесообразное вложение капитала. Отсюда следует, что в слова «удачный» и «неудачный» необходимо вкладывать смысл, обратный общепринятому.
- Поэтому будем называть тестовый прогон удачным, если в процессе его выполнения обнаружена ошибка, и неудачным, если получен корректный результат.

Стратегия черного ящика

- При таком подходе обнаружение всех ошибок в программе является критерием исчерпывающего входного тестирования. Последнее может быть достигнуто, если в качестве тестовых наборов использовать все возможные наборы входных данных.
- Если в задаче о треугольниках один треугольник корректно признан равносторонним, нет никакой гарантии того, что все остальные равносторонние треугольники так же будут корректно идентифицированы. Поскольку программа представляет собой черный ящик, единственный способ удовлетворения приведенному выше критерию – перебор всех возможных входных значений.

Стратегия черного ящика

- Вполне вероятно, что останутся некоторые ошибки, например, метод может представить треугольник со сторонами 3, 4, 5 неразносторонним, а со сторонами 2, A, 2 – равносторонним. Для того, чтобы обнаружить подобные ошибки, нужно перебрать не только все разумные, но и все вообще возможные входные наборы. Следовательно, мы приходим к выводу, что для исчерпывающего тестирования задачи о треугольниках требуется бесконечное число тестов.

Следствия

- **нельзя создать тест, гарантирующий отсутствие ошибок**;
- **разработка таких тестов противоречит экономическим требованиям.**

Следствия

- Поскольку исчерпывающее тестирование исключается, нашей целью должна стать максимизация результативности капиталовложений в тестирование (иными словами, максимизация числа ошибок, обнаруживаемых одним тестом).
- Для этого мы можем рассматривать внутреннюю структуру программы и делать некоторые разумные, но, конечно, не обладающие полной гарантией достоверности предположения.

Стратегия белого ящика

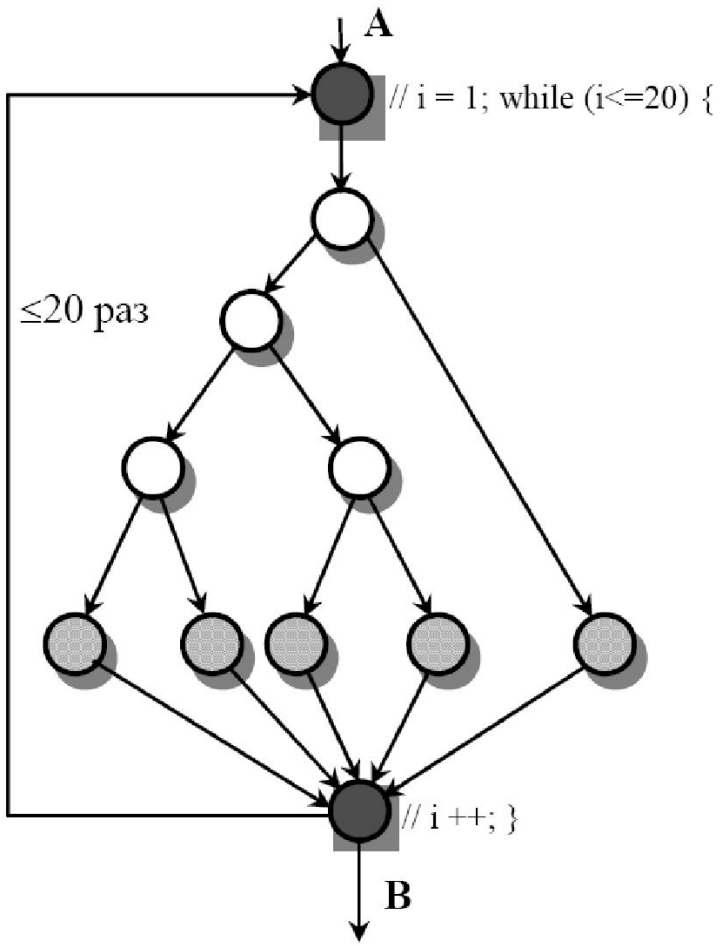
- Стратегия белого ящика, или стратегия тестирования, управляемого логикой программы, позволяет исследовать внутреннюю структуру программы.
- В этом случае тестирующий получает тестовые данные путем анализа логики программы

Стратегия белого ящика

- Исчерпывающему входному тестированию может быть поставлено в соответствие исчерпывающее тестирование маршрутов. Подразумевается, что программа проверена полностью, если с помощью тестов удается осуществить выполнение этой программы по всем возможным маршрутам ее потока (графа) передач управления.

Недостатки

- Число не повторяющих друг друга маршрутов в программе – астрономическое,
- $5^{20}+5^{19}+\ldots+5^{1}=10^{14}$



Недостатки

- хотя исчерпывающее тестирование маршрутов является полным тестом и хотя каждый маршрут программы может быть проверен, сама программа будет содержать ошибки.
  - Во-первых, исчерпывающее тестирование маршрутов не может дать гарантии того, что программа соответствует описанию.
  - Во-вторых, программа может быть неверной в силу того, что пропущены некоторые маршруты. Исчерпывающее тестирование маршрутов не обнаружит их отсутствия.
  - В-третьих, исчерпывающее тестирование маршрутов не может обнаружить ошибок, появление которых зависит от обрабатываемых данных. Существует множество примеров таких ошибок.

Принципы тестирования

Описание предполагаемых значений выходных данных или результатов должно быть необходимой частью тестового набора.

Следует избегать тестирования программы ее автором.

Программирующая организация не должна сама тестировать разработанные ею программы.

Необходимо досконально изучать результаты применения каждого теста.

Тесты для неправильных и непредусмотренных входных данных следует разрабатывать так же тщательно, как для правильных и предусмотренных.

Принципы тестирования

Необходимо проверять не только, делает ли программа то, для чего она предназначена, но и не делает ли она то, что не должна делать.

Не следует выбрасывать тесты, даже если программа уже не нужна.

Нельзя планировать тестирование в предположении, что ошибки не будут обнаружены.

Вероятность наличия необнаруженных ошибок в части программы пропорциональна числу ошибок, уже обнаруженных в этой части.

Тестирование — процесс творческий.

Классификация ошибок

- По времени появления ошибки можно разделить на три вида:
  - **Структурные ошибки.** К данному типу ошибок относятся такие как: несоответствие числа открывающих скобок числу закрывающих, отсутствие парного оператора (например, **try** без **catch**), неправильное употребление синтаксических знаков и т. п.
  - **Ошибки компиляции.** Возникают из-за ошибок в тексте кода. Они включают ошибки в синтаксисе, неверное использование конструкций языка.
  - **Ошибки периода выполнения.** Возникают, когда программа выполняется и обнаруживается, что оператор делает попытку выполнить недопустимое или невозможное действие. Например, деление на ноль.

Классификация ошибок

- По степени нарушения логики на:
  - **Синтаксические ошибки** заключаются в нарушении правописания или пунктуации в записи выражений, операторов и т. п., т. е. в нарушении грамматических правил языка.
  - **Семантические ошибки** заключаются в нарушении порядка операторов, параметров функций и употреблении выражений.
  - **Прагматические ошибки** (или логические) заключаются в неправильной логике алгоритма, нарушении смысла вычислений и т. п. Они являются самыми сложными и крайне трудно обнаруживаются.

Ошибки на этапе тестирования

- На этапе тестирования ищутся **прагматические ошибки** периода выполнения, так как остальные выявляются в процессе программирования.

Группы ошибок

- **Ошибка адресации** – ошибка, состоящая в неправильной адресации данных (например, выход за пределы участка памяти).
- **Ошибка ввода-вывода** – ошибка, возникающая в процессе обмена данными между устройствами памяти, внешними устройствами.
- **Ошибка вычисления** – ошибка, возникающая при выполнении арифметических операций (например, разнотипные данные, деление на нуль и др.).
- **Ошибка интерфейса** – программная ошибка, вызванная несовпадением характеристик фактических и формальных параметров (как правило, семантическая ошибка периода компиляции, но может быть и логической ошибкой периода выполнения).
- **Ошибка обращения к данным** – ошибка, возникающая при обращении программы к данным (например, выход индекса за пределы массива, не инициализированные значения переменных и др.).
- **Ошибка описания данных** – ошибка, допущенная в ходе описания данных.

Классификация

- По объекту тестирования
  - Функциональное тестирование (functional testing)
  - Тестирование производительности (performance testing)
  - Юзабилити-тестирование (usability testing)
  - Тестирование интерфейса пользователя (UI testing)
  - Тестирование безопасности (security testing)
  - Тестирование локализации (localization testing)
  - Тестирование совместимости (compatibility testing)

Тестирование производительности

- Нагрузочное тестирование (load testing)
- Стресс-тестирование (stress testing)
- Тестирование стабильности (stability / endurance / soak testing)

По знанию системы

- Тестирование чёрного ящика (black box)
- Тестирование белого ящика (white box)
- Тестирование серого ящика (grey box)

По степени автоматизации

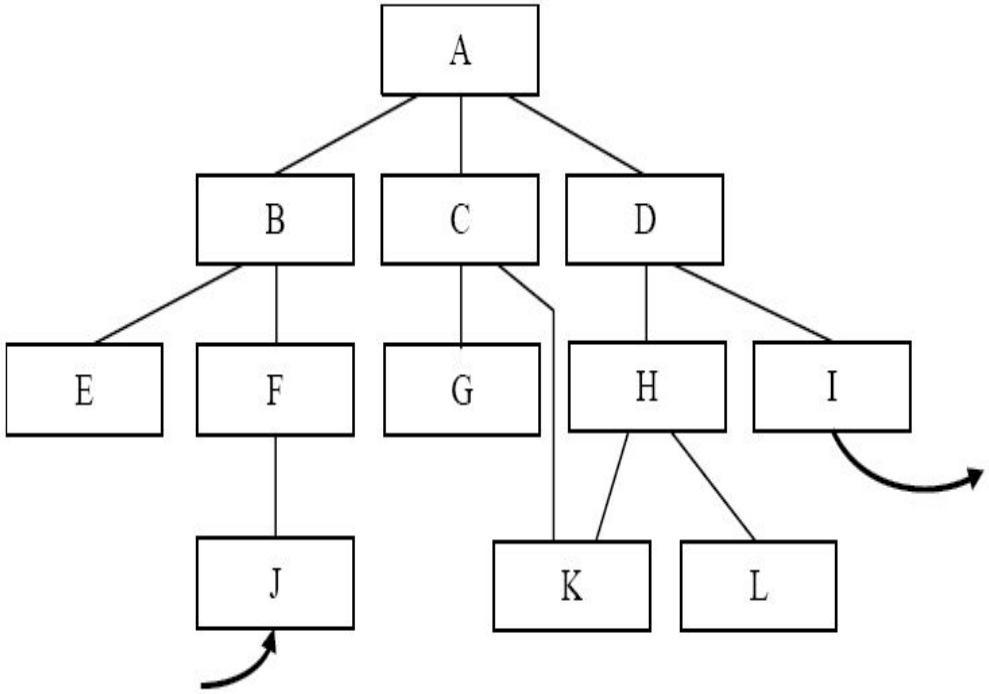
- Ручное тестирование (manual testing)
- Автоматизированное тестирование (automated testing)
- Полуавтоматизированное тестирование (semiautomated testing)

По степени изолированности компонентов

- Компонентное (модульное) тестирование (component/unit testing)
- Интеграционное тестирование (integration testing)
- Системное тестирование (system/end-to-end testing)

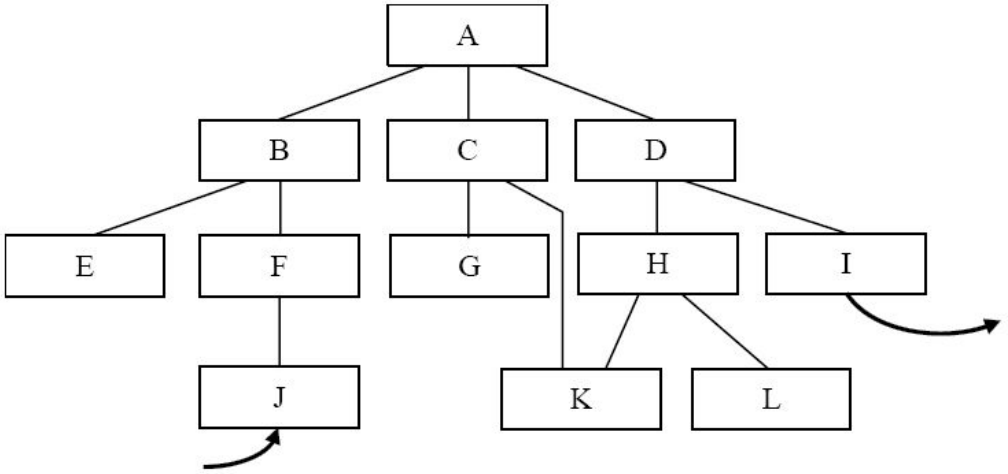
Нисходящее тестирование

- Нисходящее тестирование начинается с верхнего, головного класса (или модуля) программы.



Восходящее тестирование

- Данная стратегия предполагает начало тестирования с терминальных классов (т. е. классов, не использующих методы других классов).



По времени проведения

- Альфа-тестирование (alpha testing)
- Тестирование при приёмке (smoke testing)
- Тестирование новой функциональности (new feature testing)
- Регрессионное тестирование (regression testing)
- Тестирование при сдаче (acceptance testing)
- Бета-тестирование (beta testing)

Стратегия Тестирования

- Если спецификация содержит комбинации входных условий, то начать рекомендуется с применения метода функциональных диаграмм. Однако, данный метод достаточно трудоемок.
- В любом случае необходимо использовать анализ граничных значений. Этот метод включает анализ граничных значений входных и выходных переменных. Анализ граничных значений дает набор дополнительных тестовых условий.

Стратегия Тестирования

- Определить правильные и неправильные классы эквивалентности для входных и выходных данных и дополнить, если это необходимо, тесты, построенные на предыдущих шагах.
- Для получения дополнительных тестов рекомендуется использовать метод предположения об ошибке.
- Проверить логику программы на полученном наборе тестов. Для этого нужно воспользоваться критерием покрытия решений, покрытия условий, покрытия решений/условий либо комбинаторного покрытия условий (последний критерий является более полным).

ИТОГ

- Проектирование теста достаточно трудоемкий процесс. Оно включает в себя следующие этапы:
  - задаться целью теста;
  - написать входные значения;
  - написать предполагаемые выходные значения;
  - выполнить тест и зафиксировать результат;
  - проанализировать результат.