Основы разработки тестовых сценариев

Боциев А.Я., Виценко А.Ю., Крюков А.К., Моренов О.А.,
Пряхин И.В., Семенов Д.С., Чиликин Е.В. Intel

Составляющие теста

Цель теста

Метод тестирования

Окружение, данные, подготовка системы

Сценарий: Шаги, оформленные таблицей или списком

- Действия
- Ожидаемые результаты

Опциональная дополнительная информация:

- Снимки экрана
- Логи
- Файлы, сгенерированные в процессе теста
- ….

Требования к хорошему тестовому сценарию

- Существует обоснованная вероятность выявления тестом дефекта.
- Определены входные данные.
- Определен ожидаемый результат, считаемый «хорошим».
- Воспроизводимость.
- Независимость: может исполняться независимо. Оставляет систему в том же состоянии.
- Тест должен быть наилучшим в своей категории.
- Экономичный. Нет избыточных шагов.

Основные ошибки при составлении тестовых сценариев

- Слишком длинный сценарий.
- Неполное, неправильное или непоследовательное описание условий тестирования или подготовки тестового окружения.
- Пропущенные «очевидные» шаги.
- Использование устаревшей информации о тестируемой системе.
- Неочевидно, кто должен выполнить действие: пользователь или система.
- Неясно, что является успешным результатом.
- Отсутствие очистки системы.

Методы выбора входных значений

Бессистемный выбор входных значений не позволит найти большое количество дефектов. Необходимо использование методов для выбора набора входных значений.

Основные методы выбора входных значений:

- Перебор всех возможных значений,
- Случайные входные данные,
- Предугадывание ошибки,
- Построение графов «причина-следствие»,
- Использование классов эквивалентности,
- Исследование граничных значений.

Метод перебора

Перебираем все возможные значения входных параметров.

Последовательный перебор всех возможных комбинаций входных значений.

Попарный перебор. Перебираем комбинации пары 2х входных параметров. Работаем в предположении, что параметры попарно зависимы. На практике находит ~80% функциональных дефектов низкого уровня.

Случайные входные данные

Генерируются случайные входные данные. Либо данные случайным образом выбираются из большого тестового набора, который не успеваем проверить целиком.

- Часто используется в нагрузочном тестировании.
- Необходимо иметь метод определения корректности выхода.

Пример: программа подсчета числа вхождений символа в строку.

Предугадывание ошибки

Составление тестовых сценариев на основании опыта предыдущего тестирования.

Используйте знания об известных проблемных местах вашего продукта.

Знайте распространенные ошибки программирования и пишите тесты для их поиска.

- Некорректная работа с памятью: переполнение, чтение за пределами, утечки памяти.
- Отсутствие обработки некорректных входных данных.
- Ошибки работы с типами данных: переполнение, приведение, приближение.
- Ошибки многопоточности: deadlock, data race.
- Отсутствие инициализации/сброса переменных.
- Недостаток привилегий, недоступность ресурсов.
- ….

Графы причина-следствие

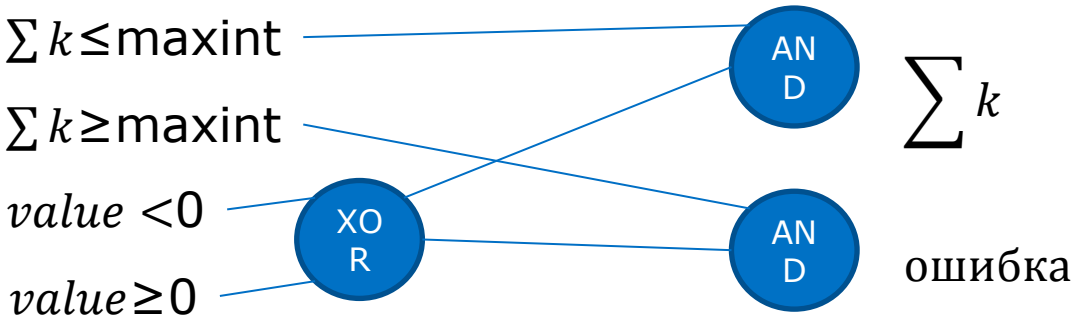
- Выделяем причины и следствия в спецификациях.
- Строим граф, связывающий причины и следствия.
- Выписываем невозможные сочетания причин и следствий.
- Разрабатываем «таблицу решений», где в каждом столбце конкретная комбинация входов и выходов.
- Превращаем каждый столбец в тестовый сценарий.

Преимущества и недостатки:

- Комбинаторный взрыв числа вариантов.
- Позволяет систематизировать процесс построения сценариев.
- Используются эвристики для уменьшения числа комбинаций.

Пример графа

$result = \sum_{k=0}^{|value|} k$, если <=maxint или ошибка



Вход	$\sum k \leq$maxint	1	1	0	0
	$\sum k \geq$maxint	0	0	1	1
	$value <0$	1	0	1	0
	$value \geq 0$	0	1	0	1
Выход	$\sum k$	1	1	0	0
	ошибка	0	0	1	1

Классы эквивалентности

Если от двух тестов ожидается одинаковый результат, – они эквивалентны.

Группа тестов представляет **класс эквивалентности,** если:

- Все тесты предназначены для выявления одной и той же ошибки.
- Если один тест выявит ошибку, то и остальные это сделают.
- Если один из тестов не выявит ошибку, то и остальные этого не сделают.

Дополнительные практические критерии:

- Тесты включают значения одних и тех же входных данных.
- Для проведения теста выполняются одни и те же операции программы.
- В результате тестов формируются значения одних и тех же выходных данных.
- Ни один из тестов не вызывает выполнения конкретного блока обработки ошибок либо выполнение этого блока вызывается всеми тестами.

Классы эквивалентности - примеры

Программа классификации треугольников.

Классы эквивалентности по корректным входным данным:

- Равнобедренные треугольники.
- Равносторонние треугольники.
- Прямоугольные треугольники.
- Просто треугольники.

Классы эквивалентности по некорректным входным данным:

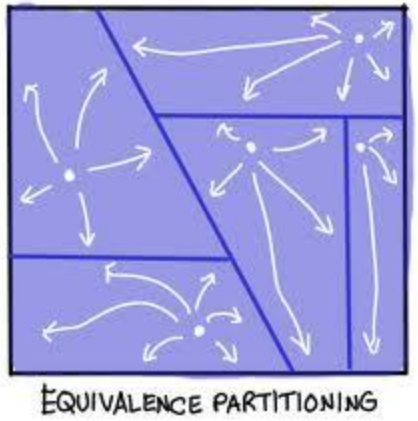
- Отрезки не образуют треугольник.
- Числа больше sizeof(int).
- Строка, содержащая буквы.

Классы эквивалентности - Примеры

Программа, говорящая дату следующего дня.

Классы эквивалентности по корректным входным данным:

- День от 1 до 27;
- Последний день месяца;
- Последний день года;
- 28 февраля високосного года.



Классы эквивалентности по некорректным входным данным:

- Месяц > 12;
- День > 31;
- Неверная строка.

Поиск классов эквивалентности

Построение классов эквивалентности – субъективный процесс.

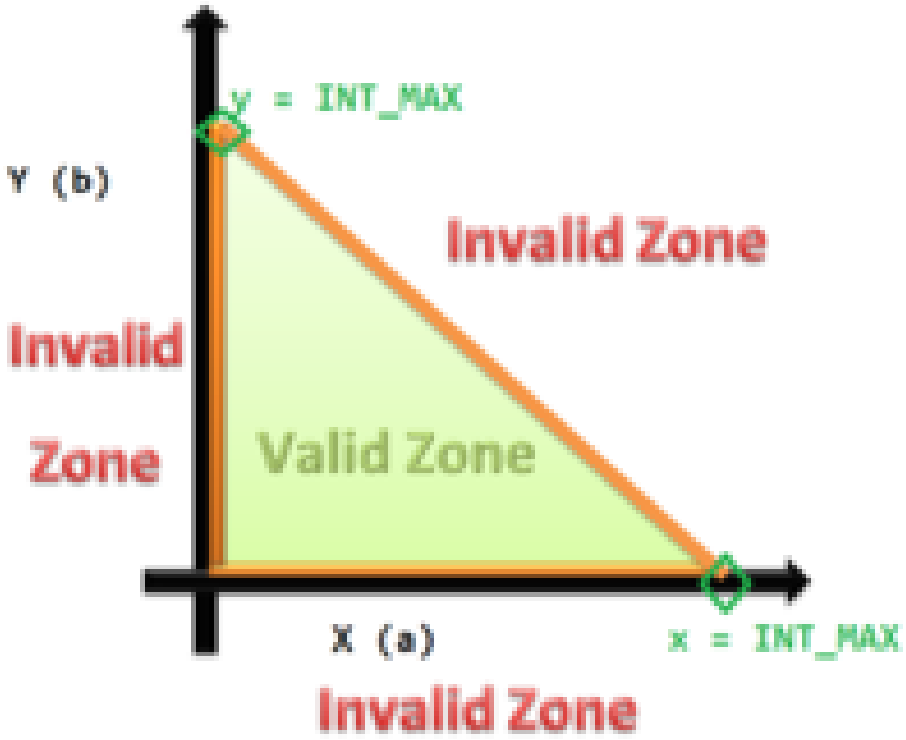
Общие рекомендации:

- Не забывайте о классах некорректных данных.
- Формируйте классы в виде таблицы или плана.
- Определите диапазоны числовых значений входных данных.
- Проанализируйте варианты выбора из списков и меню.
- Поищите переменные, значения которых должны быть равными.
- Поищите классы значений, зависящих от времени.
- Выявите группы переменных, совместно участвующих в конкретных вычислениях.
- Посмотрите, на какие действия программа отвечает эквивалентными событиями.
- Продумайте варианты среды тестирования.

Граничное тестирование

Тестирование значений, лежащих на границе классов эквивалентности, т.к. там выше вероятность возникновения ошибки

```
int safe_add( int a, int b )
{
    int c = a + b ;
    if ( a >= 0 && b >= 0 && c
< 0 )
    {
       fprintf ( stderr,
"Overflow!\n");
    }
    if ( a < 0 && b < 0 && c >=
0 )
    {
       fprintf ( stderr,
"Underflow!\n");
    }
    return c;
}
```



Граничное тестирование - применение

- Определяем границу класса эквивалентности.
- Проверяем значения, лежащие ровно на границе.
- Проверяем значения, лежащие максимально близко к границе с обоих сторон.

Пример:
При покупке более 100 единиц товара дается скидка 5%. Нужно проверить:

- 100
- 99
- 101

Оракулы

Оракул – эвристический принцип или механизм идентификации потенциальной проблемы

- Оракулы – это эвристики. Они несовершенны и подвержены ошибкам
- Оракул лишь может указать нам на проблемное место, но не дать ответ, корректно ли поведение программы.

	Интерпретация теста		
Реальное состояние программы		**Дефект**	**Функционал**
	Дефект	Hit	Miss
	Функционал	False Alarm	Correct acceptance

Оракулы. Примеры-1

	Описание	Преимущества	Недостатки
Нет оракула (некомпетентный человек)	Не проверяем результаты. Просто «исполняем пока не упадет»	• Можем использовать любые объемы данных • Полезно на начальных стадиях тестирования	• Находим только самые серьезные и видимые проблемы • Сложность воспроизведения
Нет оракула (компетентный человек)	Человек выполняет тест, не зная правильного результата. Используется «здравый смысл» для определения правильности результата	• Возможность нахождения широкого круга проблем	• Субъективность результата • Плохо работает с неопытными тестерами
Идеальный оракул	Механизм всегда дающий ответ о прохождении/провале любого теста	• Находит все типы проблем • Можем автоматизировать все	• Мифический объект. Не существует в реальности

Оракулы. Примеры-2

	Описание	Преимущества	Недостатки
Оракулы согласованности	Согласованно: • Внутри продукта • Сопоставимыми продуктами • Образом • Заявлениями • Историей • Ожиданиями пользователя • целью	• Большинство оракулов попадает в эту структуру • Дает идеи для дизайна тестов и убедительное обоснование результатов	• Слишком общая структура
Частичный	• Проверяет только некоторые аспекты тестового вывода • Все оракулы - частичны	• Более высока вероятность существования • Более низкая стоимость создания и использования	• Может пропустить систематические ошибки • Может пропустить очевидные ошибки

Оракулы. Примеры-3

	Описание	Преимущества	Недостатки
Ограничения	Проверка на: • Недопустимые значения (пример – почтовый код) • Недопустимые взаимоотношения (пример - размер страницы печати должен соответствовать возможностям принтера)	• Находит самые очевидные ошибки программы • Полезно, на любых стадиях разработки	• Недостаточно для проверки всей программы • Пропускает ошибки по корректности значений
Паттерн знакомой проблемы	• Программы ведет себя похоже на другую программу с известной проблемой • Недостаточно для заведения дефекта, но мотивирует «копать глубже»	• Диагностирует потенциальные проблемы (но не указывает на них)	• Может отвлекать значительные ресурсы на исследование

Оракулы. Примеры-4

	Описание	Преимущества	Недостатки
Регрессия	Сравниваем текущие результаты с предыдущими. Считаем предыдущие результаты правильными	• Простота реализации • Можем работать с большими объемами данных • Множество инструментов уже существуют для помощи в таких проверках	• Не подходит если поменялся дизайн • Пропустит проблемы не найденные в предыдущем запуске
Само-проверяющие данные (self-verifying data)	• Встроить правильный ответ в тестовые данные • CRC, digital signature, hashes, checksums	• Дает возможность анализа результатов • Малая зависимость от пользовательского интерфейса • Можем работать с большими объемами данных • Не требует внешних оракулов	• Должны определить ответ до запуска теста • Может потребовать переработки всех тестов при изменении логики или архитектуры • Пропустит дефекты не относящиеся к проверяемым полям

Оракулы. Примеры-5

	Описание	Преимущества	Недостатки
Модель – конечный автомат	Представляем программу в виде конечного автомата	• Хороший выбор вспомогательных инструментов • Хорошо автоматизируется	• Поддержка модели может быть очень дорогой • Не выходит за рамки заданных переходов • Часть ошибок остается невидимой
Модель взаимодей-ствия	• Мы знаем, что если выполнить действие X, то другая часть системы должна сделать Y	• Хорошо поддается автоматизации	• Лишь срез поведения программы. Возможность ложной тревоги и пропуска ошибок • Построение модели может потребовать много времени

Оракулы. Примеры-6

	Описание	Преимущества	Недостатки
Бизнес-модель	• Мы понимаем бизнес-процессы программы. (например, алгоритм вычисления налога, логику работы склада и т.д.)	• Как правило может быть выражено уравнениями или неравенствами • Ошибки такого рода важны для программы	• Не всегда дают однозначный ответ о корректности • Эксперт может ошибаться в рамках всех сценариев работы программы
Теоретическая модель (например физическая, математическая)	• Мы имеем теоретическую модель, описывающую поведение программы	• Хорошо подходит для мат моделей, функций • Ошибки в этом аспекте, скорее всего, важны	• Теоретические модели не всегда применимы к реальным ситуациям • Иногда сложность получения корректного результата по сложности сопоставима с исходной программой

Оракулы. Примеры-7

	Описание	Преимущества	Недостатки
Статистическая модель	Проверка относительно статистических предсказаний. Например: • X обычно больше Y • 80% покупателей имеют почтовый код из следующего набора	• Позволяет проверять большие объёмы данных • Позволяет обрабатывать данные других тестов	• Высокая вероятность misses & false alarms • Может пропускать очевидные ошибки
Данные подготов-ленные вручную	• Результаты тщательно выбираются автором тестов	• Полезно для больших и сложных систем • Ожидаемый результат может быть понят другими	• Медленно • Дорого • Высокая стоимость поддержки/ разработки
Человек	• Человек решает корректно ли поведение программы	• Иногда – это единственный способ	• Медленно • Субъективно • Зависит от доверия к принимающему

Материалы и источники

1. The Art of Software Testing. Glenford J. Myers
2. Сэм Канер, Джек Фолк, Енг Кек Нгуен. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений.
3. Материалы http://wikipedia.org
4. Материалы http://www.testingeducation.org
5. Материалы http://www.oneclicktesting.com
6. Материалы http://www.stagsoftware.com
7. Материалы http://www.softwaretestingmentor.com
8. Материалы http://www.makinggoodsoftware.com
9. Материалы http://www.cavdar.net
10. Материалы http://users.csc.calpoly.edu
11. Материалы http://www.protesting.ru
12. Материалы http://www.cs.swan.ac.uk
13. Материалы http://testingforall.com
14. Материалы http://people.cs.aau.dk
15. http://www.testingeducation.org/BBST/foundations/BBSTFoundationsNov2010.pdf

Домашнее задание - задача

Разработайте набор тестовых сценариев (как позитивных, так и негативных) для следующей программы:

Имеется консольное приложение. Ему на вход подается 2 строки. На выходе приложение выдает число вхождений второй строки в первую.

Например,

Строка 1	Строка 2	Вывод
абвгабвг	аб	2
стстсап	стс	2

Набор тестовых сценариев запишите в виде таблицы, приведенной выше.