Регрессионное тестирование
Случайное тестирование
Software Testing 102

Марат Ахин

Санкт-Петербургский государственный политехнический университет

2013

Quiz

Recap

Содержание

Тестирование ПО в процессе разработки

Как ПО изменяется в процессе разработки?

- Инкрементально, небольшими независимыми шагами
  - Изменение уже существующего кода
  - Исправление ошибок
  - Добавление новой функциональности
  - Адаптация имеющихся компонентов к новым задачам

Любые (даже самые незначительные) изменения могут серьезно повлиять на качество ПО

Тестирование ПО в процессе разработки

Как ПО изменяется в процессе разработки?

- Инкрементально, небольшими независимыми шагами
  - Изменение уже существующего кода
  - Исправление ошибок
  - Добавление новой функциональности
  - Адаптация имеющихся компонентов к новым задачам

Любые (даже самые незначительные) изменения могут серьезно повлиять на качество ПО

Тестирование ПО в процессе разработки

Как ПО изменяется в процессе разработки?

- Инкрементально, небольшими независимыми шагами
  - Изменение уже существующего кода
  - Исправление ошибок
  - Добавление новой функциональности
  - Адаптация имеющихся компонентов к новым задачам

Любые (даже самые незначительные) изменения могут серьезно повлиять на качество ПО

Тестирование ПО в процессе разработки

- После любого изменения требуется проверить, что в программе не появилось новых ошибок
- Для этого мы выполняем все имеющиеся тесты и проверяем, что все они успешно завершаются

Основной вид тестирования в процессе разработки ПО – это **регрессионное тестирование**

Регрессионное тестирование

Как выглядит одна итерация регрессионного тестирования?

1. Мы модифицируем программу P и получаем программу P'
2. Из всего множества тестов T мы выбираем набор тестов T', который необходимо выполнить на P'
3. Для новой функциональности мы разрабатываем новые тесты T''
4. Полученный набор тестов $T' + T''$ запускается на P'
5. Результаты выполнения анализируются с последующей возможной модификацией как программы, так и набора тестов

Какие проблемы связаны с RT?

Регрессионное тестирование

Проблема №1

Как выбрать набор тестов T' после изменения в программе?

- Консервативный подход
  - Выбираем все имеющиеся тесты
  - Полное регрессионное тестирование
- Экономичный подход
  - Выбираем все тесты, каким-либо образом связанные по требованиям/спецификации/интуиции с изменениями в коде
  - «Выборочное» регрессионное тестирование

Регрессионное тестирование

Каким свойствам должно удовлетворять выборочное регрессионное тестирование?

- **Полнота** – способность выбирать те тесты, которые могут обнаружить ошибки, связанные с изменениями в коде
- **Точность** – способность пропускать такие тесты, которые не изменяют своего поведения на модифицированной программе
- **Эффективность** – способность выполняться быстрее, чем полное регрессионное тестирование
- **Универсальность** – применимость в большинстве практических ситуаций

«Качественно. Быстро. Дешево. Выберите любые два.»

Регрессионное тестирование

- Умный подход
  - Выбирать тесты, которые «затрагивают» при выполнении измененные части программы
  - Выборочное регрессионное тестирование
- Существует множество подходов к RPT на различных уровнях
  - Модульное RPT
  - Интеграционное RPT
  - Системное RPT
- Все подходы различаются по двум основным критериям
  - Способ идентификации измененных программных компонентов
  - Метод получения информации о покрытии элементов программы тестами

Подход МакКарти

- Анализ изменений на уровне целого модуля
- Связь элементов программы с тестами задается вручную разработчиком

Преимущества и недостатки?

Подход на основе концепции файервола

- Анализ изменений на уровне целых модулей
- Связь элементов программы с тестами задается вручную разработчиком

Преимущества и недостатки?

Поход Ротермела и Харролд

- Анализ изменений на уровне узлов CFG программы
- Связь элементов программы с тестами задается на уровне CFG на основе динамической информации о выполнении каждого теста

Преимущества и недостатки?

Подход Балла

- Анализ изменений на уровне узлов CFG программы
- Связь элементов программы с тестами задается на уровне CFG на основе динамической информации о выполнении каждого теста

Преимущества и недостатки?

Подход на основе AST

- Анализ изменений на уровне вершин AST программы
- Связь элементов программы с тестами задается на уровне AST на основе динамической информации о выполнении каждого теста

Преимущества и недостатки?

Управление регрессионными тестами

Проблема №2

Как управлять набором регрессионных тестов?

- Когда и как добавлять в набор новые тесты?
- Когда можно удалять старые тесты?

Добавление новых тестов

Когда надо добавлять новый регрессионный тест?

- Когда в ПО появилась новая функциональность
- Когда в ПО была исправлена ошибка
- Когда мы хотим улучшить тестовое покрытие ПО
- Когда мы можем себе позволить добавить новый неповторяющийся тест

Добавление новых тестов

Когда надо добавлять новый регрессионный тест?

- Когда в ПО появилась новая функциональность
- Когда в ПО была исправлена ошибка
- Когда мы хотим улучшить тестовое покрытие ПО
- Когда мы можем себе позволить добавить новый неповторяющийся тест

Добавление новых тестов

Когда надо добавлять новый регрессионный тест?

- Когда в ПО появилась новая функциональность
- Когда в ПО была исправлена ошибка
- Когда мы хотим улучшить тестовое покрытие ПО
- Когда мы можем себе позволить добавить новый неповторяющийся тест

Добавление новых тестов

Когда надо добавлять новый регрессионный тест?

- Когда в ПО появилась новая функциональность
- Когда в ПО была исправлена ошибка
- Когда мы хотим улучшить тестовое покрытие ПО
- Когда мы можем себе позволить добавить новый неповторяющийся тест

Добавление новых тестов

- С течением времени число тестов увеличивается
- Чем больше тестов, тем лучше тестовое покрытие
- Проблемы начинаются, когда тестов становится слишком много

Что такое «слишком много»?

Удаление старых тестов

Когда можно удалять старый тест?

- Никогда
- Когда тест дублирует другие тесты
- Когда тест не улучшает тестовое покрытие
- Когда тест ни разу не обнаружил ошибки за все время тестирования
- Когда тест обнаруживает такие же ошибки, как и другие тесты

Удаление старых тестов

Когда можно удалять старый тест?

- Никогда
- Когда тест дублирует другие тесты
- Когда тест не улучшает тестовое покрытие
- Когда тест ни разу не обнаружил ошибки за все время тестирования
- Когда тест обнаруживает такие же ошибки, как и другие тесты

Удаление старых тестов

Когда можно удалять старый тест?

- Никогда
- Когда тест дублирует другие тесты
- Когда тест не улучшает тестовое покрытие
- Когда тест ни разу не обнаружил ошибки за все время тестирования
- Когда тест обнаруживает такие же ошибки, как и другие тесты

Удаление старых тестов

Когда можно удалять старый тест?

- Никогда
- Когда тест дублирует другие тесты
- Когда тест не улучшает тестовое покрытие
- Когда тест ни разу не обнаружил ошибки за все время тестирования
- Когда тест обнаруживает такие же ошибки, как и другие тесты

Удаление старых тестов

Когда можно удалять старый тест?

- Никогда

- Когда тест дублирует другие тесты
- Когда тест не улучшает тестовое покрытие
- Когда тест ни разу не обнаружил ошибки за все время тестирования
- Когда тест обнаруживает такие же ошибки, как и другие тесты

Приоритизация регрессионных тестов

Проблема №3

Как запускать регрессионные тесты?

- Взяли имеющийся набор тестов и запустили их
- Такой подход может не всегда нас устраивать
- Что мы можем изменить?

Приоритизация регрессионных тестов

- Мы можем изменить **порядок**, в котором мы запускаем регрессионные тесты
- Зачем?
  - Чем раньше мы узнаем о том, что в ПО появилась регрессионная ошибка, тем скорее мы сможем приступить к ее исправлению
  - Часто причиной непрохождения различных (напрямую не связанных друг с другом) тестов является одна и та же ошибка в ПО
  - Иногда время на тестирование является ограниченным, и необходимо найти наибольшее число ошибок с учетом всех ограничений

Как мы можем приоритизировать регрессионные тесты?

Приоритизация регрессионных тестов

- При помощи интуиции
  - Подход работает, если у Вас хорошая интуиция
  - Кроме интуиции можно использовать имеющийся опыт разработки ПО
- На основе знаний о тестовом покрытии ПО
  - Сперва выполняются тесты, которые имеют наибольшее покрытие ПО
  - Сперва выполняются тесты, которые покрывают более важные компоненты ПО
- На основе истории разработки
  - Приоритет отдается тестам, которые чаще других обнаруживали регрессионные ошибки
  - Первыми выполняются тесты, проверяющие корректность работы наиболее «проблемных» компонентов ПО

Приоритизация регрессионных тестов

- Случайным образом
  - Подход перекликается со случайным BPT
  - Если мы можем случайным образом поменять порядок выполнения тестов, то почему бы это не сделать?
- На основе характеристик тестов
  - Первыми выполняются тесты с наименьшим временем выполнения
  - Приоритет отдается тестам, которые наиболее активно работают с окружением программной системы

Анализ результатов регрессионного тестирования

Проблема №4

Что делать с результатами регрессионного тестирования?

- Если все тесты проходят – все хорошо
- Если тест не проходит – то все зависит от того, по какой причине он не проходит

Варианты?

Анализ результатов регрессионного тестирования

- В тестируемом модуле есть ошибка
  - Ошибку необходимо локализовать и исправить
  - После исправления требуется выполнить повторное РТ
- Тест больше не является корректным
  - Например, была изменена спецификация модуля
  - Необходимо либо обновить тест, либо удалить его

Регрессионное тестирование на практике

Как обстоит дело с PT/BPT на практике?

Регрессионное тестирование на практике

- PT используется очень часто
  - TDD
  - Agile Development
  - RUP
- BPT практически не используется

Почему?

Регрессионное тестирование на практике

- Крайняя сложность выбора регрессионных тестов
- Опасность пропустить регрессионную ошибку при использовании небезопасного BPT
- Страх перед использованием «непонятной» технологии
- Простота экстенсивного пути решения проблем PT
- Отсутствие инструментальной поддержки
  - До недавнего времени

Регрессионное тестирование на практике

- Сложность получения динамической информации о покрытии тестами компонентов программы
  - Требуется инструментирование ПО дополнительными трассирующими вызовами
  - Журналирование особого вида

Какие сложности могут при этом быть?

Содержание

Генерация тестов

Основная идея

Заставить компьютер работать вместо нас

- Дешевле
- Быстрее
- Нет человеческого фактора

Генерация тестов

- kd-tree
- stoi
- md5sum
- PDF reader

Fuzzing

- Прародитель случайного тестирования

- Полностью случайные данные
- Вариант smoke testing

```
P8FT8PjBG7s71Bw1a8EP4svDPL5g4E791
TJcs5t9ZbxQAsLZx436PdJcxk3vq61192
```

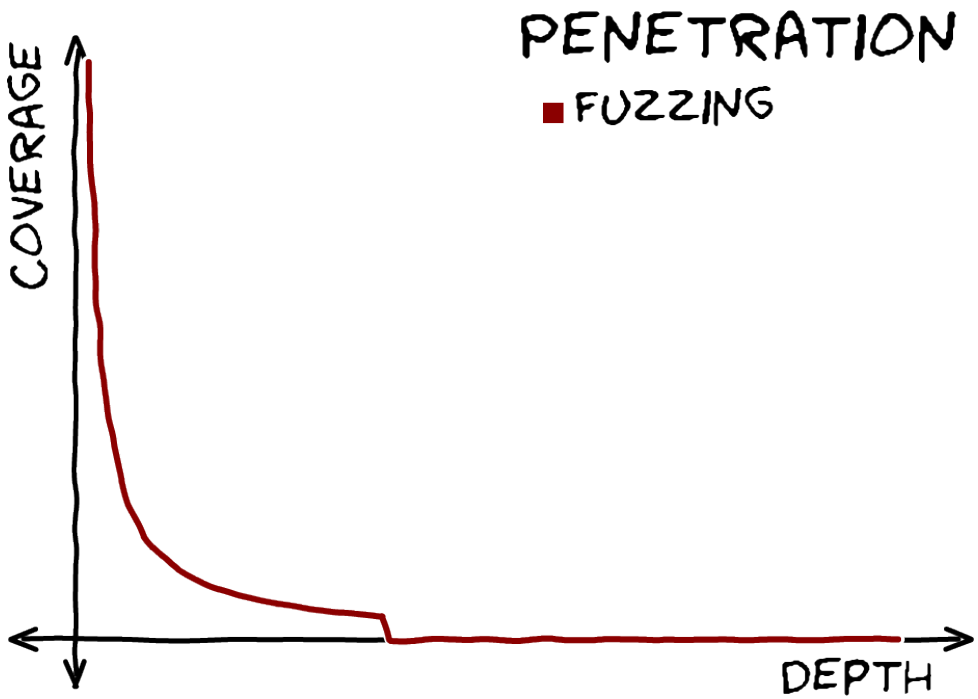
Fuzzing

- md5sum
- kd-tree
- stoi
- PDF reader

- Работает для всех программ
- Но есть одно «но»... ©

Проблема валидности данных

- Полностью случайные данные являются невалидными входными данными для большинства программ



Проблема валидности данных

- Большинство программ ожидают структурированные входные данные

```
P8FT8PjBG7s71Bw1a8EP4svDPL5g4E791
TJcs5t9ZbxQAsLZx436PdJcxk3vq61192
```

vs

```
3.14159265358979323846264338327 97
[(1,0,0),(0,1,0),(0,0,1),(1,1,1)]
```

Generative random testing

- Если мы знаем структуру, то мы можем ей воспользоваться

- Генерируем отдельные элементы
- Комбинируем их в соответствии с заданной структурой
- Вносим случайные нарушения структуры

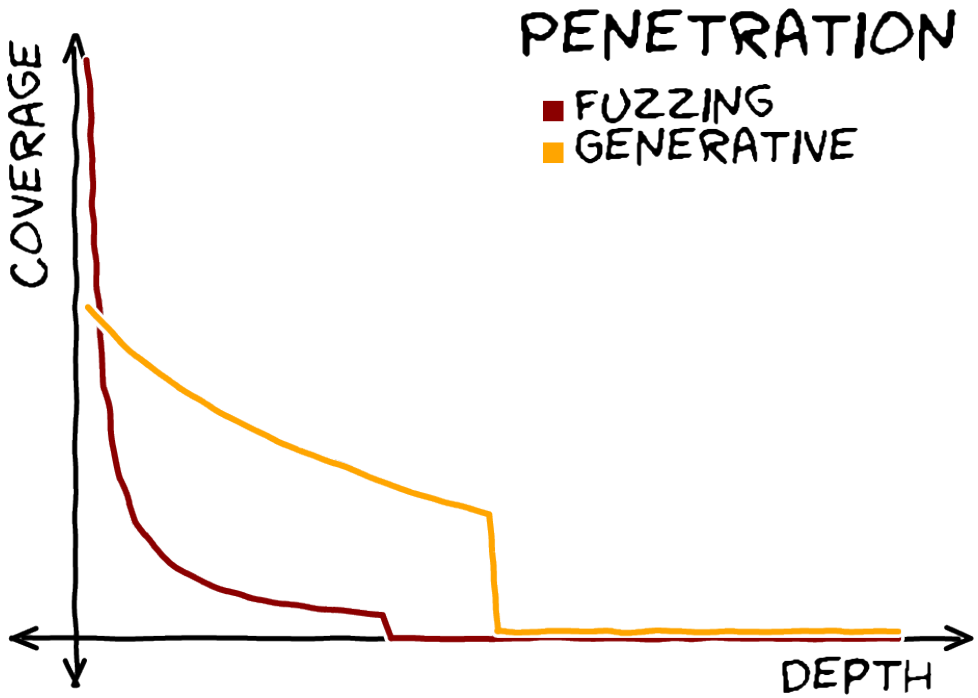
Generative random testing

- kd-tree
- stoi
- PDF reader

- Работает для структурированных входных данных
- Но есть одно «но»... ©

Проблема сложной структуры

- Иногда структура входных данных является слишком сложной



Mutation random testing

- Обычно у нас есть какой-то набор тестовых входных данных

- Подвергаем тестовые данные мутации
- При этом возможно использование знания структуры данных
- Часть данных может генерироваться случайно

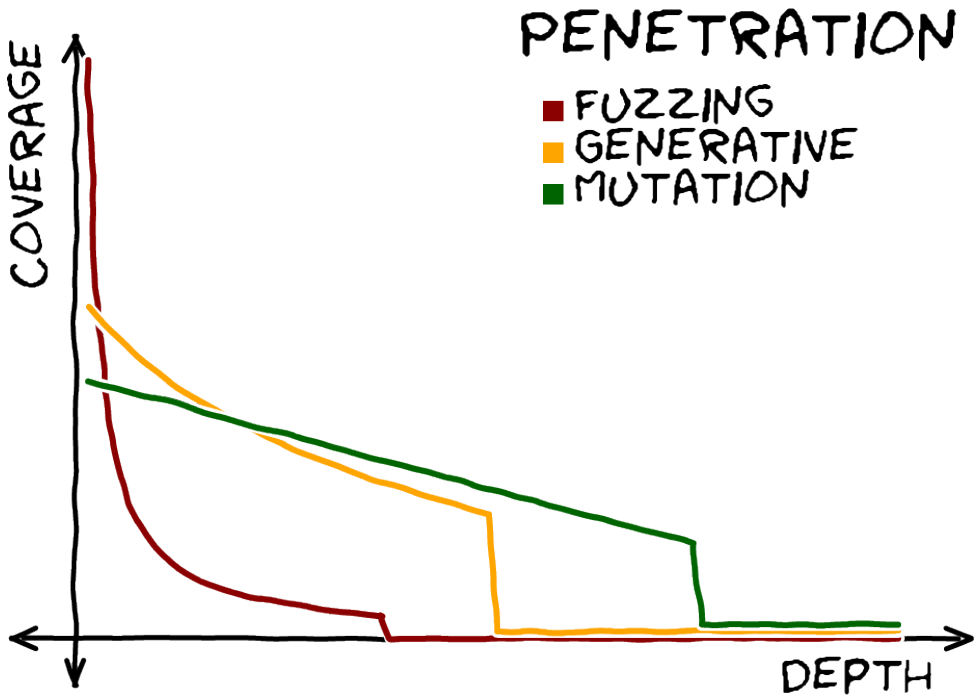
Mutation random testing

- PDF reader
- Web browser

- Работает практически для всего
- Но есть одно «но»... ©

Проблема скелета в шкафу

- Добраться до самых дальних закоулков нельзя



Directed random testing

- Мы можем узнать, где мы были
- Мы можем узнать, как мы туда пришли

А дальше начинается магия...

Directed random testing

```
1  void f(int x) {
2    int y = x^2;
3
4    bool A = x > 0;
5    bool B = x < 10;
6    bool C = y > 20;
7
8    if (B && C) {
9      if (A) {
10       ...
11     }
12   }
13 }
```

SMT FTW!

SMT $\Leftrightarrow$ NP-hard

Never believe NP-hard is always hard!

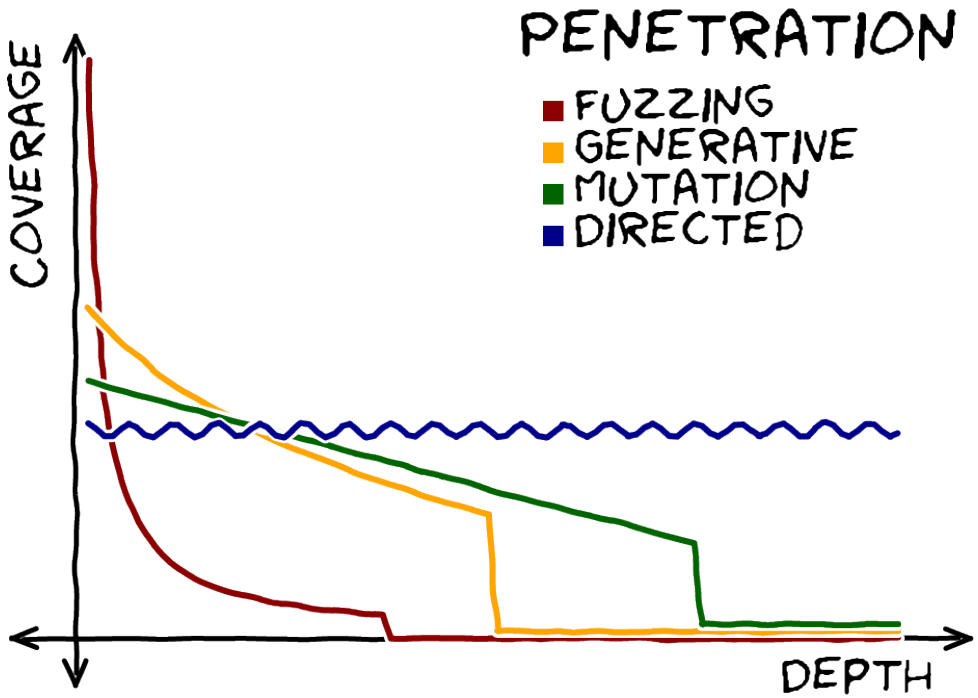
Directed random testing

- Visual Studio
- Microsoft Office

- Работает для всего
- Но есть одно «но»... ©

Проблема мистера Икс

- Некоторые части системы могут быть «черным ящиком»



W.I.L.T.

Содержание

Homework

Try `http://pex4fun.com`