# Интеграция тестирования в жизненный цикл разработки ПО

## Software Testing 102

Марат Ахин

Санкт-Петербургский государственный политехнический университет

2011

# Quiz

# 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 РТ?

# Регрессионное тестирование

## Проблема №1

Как выбрать набор тестов $T'$ после изменения в программе?

- Консервативный подход
  - Выбирать все имеющиеся тесты
  - Полное регрессионное тестирование
- Экономичный подход
  - Выбирать 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

# Модульное BPT

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

Преимущества и недостатки?

# Модульное BPT

- 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?

# Модульное BPT

- 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?

# Модульное BPT

- 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?

# Интеграционное BPT

- Подход на основе концепции файервола
  - Анализ изменений на уровне целых модулей
  - Связь элементов программы с тестами не учитывается

Преимущества и недостатки?

# Интеграционное BPT

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

Преимущества и недостатки?

# Системное ВРТ

Какие варианты Вы можете предложить?

Вспомните, что системное тестирование – это модульное тестирование «очень большого и сложного ящика»

Какие варианты Вы можете предложить теперь?

# Системное BPT

Какие варианты Вы можете предложить?

Вспомните, что системное тестирование – это модульное тестирование «очень большого и сложного ящика»

Какие варианты Вы можете предложить теперь?

# Системное ВРТ

Какие варианты Вы можете предложить?

Вспомните, что системное тестирование – это модульное тестирование «очень большого и сложного ящика»

Какие варианты Вы можете предложить теперь?

# 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
- Входные тестовые данные больше не являются корректными
  - Например, была изменена спецификация модуля
  - Необходимо либо обновить тест, либо удалить его
- Ожидаемые выходные данные изменились
  - Опять же, была изменена спецификация модуля
  - Необходимо обновить тест

# Анализ результатов регрессионного тестирования

- Часто довольно сложно понять, о какой именно ситуации идет речь в каждом конкретном случае
  - Особенно если мы тестируем модули со сложной функциональностью
  - В этом нам нисколько не помогает проблема рассинхронизации спецификации и тестов

Как обстоит дело с PT/BPT на практике?

# PT на практике

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

Почему?

# РТ на практике

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

# PT на практике

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

Какие сложности могут при этом быть?

# Содержание

# Уровни тестирования

## Уровень 0

Тестирование ничем не отличается от отладки

## Уровень 1

Цель тестирования – доказать корректность программы

## Уровень 2

Цель тестирования – доказать, что в программе есть ошибки

## Уровень 3

Цель тестирования – увеличить качество ПО

# Уровни тестирования

## Уровень 0

Тестирование ничем не отличается от отладки

## Уровень 1

Цель тестирования – доказать корректность программы

## Уровень 2

Цель тестирования – доказать, что в программе есть ошибки

## Уровень 3

Цель тестирования – увеличить качество ПО

# Уровни тестирования

**Уровень 0**

Тестирование ничем не отличается от отладки

**Уровень 1**

Цель тестирования – доказать корректность программы

**Уровень 2**

Цель тестирования – доказать, что в программе есть ошибки

**Уровень 3**

Цель тестирования – увеличить качество ПО

# Уровни тестирования

**Уровень 0**

Тестирование ничем не отличается от отладки

**Уровень 1**

Цель тестирования – доказать корректность программы

**Уровень 2**

Цель тестирования – доказать, что в программе есть ошибки

**Уровень 3**

Цель тестирования – увеличить качество ПО

# Уровни тестирования

## Уровень 4

Тестирование – это одна из частей процесса разработки ПО

- Подавляющее большинство компаний находятся на уровнях 0 и 1
  - «Monkey testing»
  - Цена/качество
- Некоторые компании «доросли» до уровней 2 и 3
  - Тестирование не является часть процесса разработки
  - Во главу угла ставится непосредственно разработка ПО

# Открытые проблемы в тестировании ПО

- Как тестировать «сверх-высоконадежное» ПО?
- Можно ли автоматизировать генерацию тестов?
- Как перейти к активному использованию ВРТ?
- Как оценивать полноту тестирования в промышленных проектах?
- Как использовать результаты научных исследований на практике?

# Революция в тестировании

Тестирование должно стать неотъемлимой частью разработки ПО