
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

**Факультет компьютерных наук**
**Департамент программной инженерии**

# ОБЕСПЕЧЕНИЕ КАЧЕСТВА И ТЕСТИРОВАНИЕ

## Семинар 7: Мутации исходного кода

Москва, 2020

# ПОКРЫТИЕ КОДА

- Покрытие кода — мера, используемая при тестировании программного обеспечения. Она показывает процент исходного кода программы, который был выполнен в процессе тестирования.

# ПОКРЫТИЕ КОДА

- На основе структурных элементов тестируемой системы, которые выполняются или задействуются в ходе тестирования.
- На основе структуры входных данных, используемых во время тестирования.
- На основе элементов требований, проверяемых при выполнении тестов.
- На основе явно сформулированных предположений об ошибках, выявление которых должны обеспечить тесты.
- На основе произвольных моделей устройства или функционирования тестируемой системы

# МУТАЦИОННОЕ ТЕСТИРОВАНИЕ

- Мутационное тестирование - это метод тестирования ПО, основанный на всевозможных изменениях исходного кода и проверке реакции на эти изменения набора автоматических тестов. Если тесты после изменения кода успешно выполняются, значит либо код не покрыт тестами, либо написанные тесты неэффективны.
- Критерий, определяющий эффективность набора автоматических тестов, называется Mutation Score Indicator (MSI).

# МУТАЦИОННОЕ ТЕСТИРОВАНИЕ

- Мутация (Mutation) - одно изменение исходного кода
- Мутант (Mutant) - результат мутации, новый мутированный исходный код.
- Убитый мутант (Killed mutant) - тесты отреагировали на изменение в коде и поймали ошибку(тесты упали).
- Выживший мутант (Survived, Escaped Mutant) - после мутирования тесты успешно выполнились
- Идентичные/эквивалентные мутанты - мутации, которые приводят к идентичному коду с точки зрения логики

# МУТАЦИОННЫЕ ОПЕРАТОРЫ

- Мутационные операторы (Mutation Operator, Mutator):
  - Удалить оператор программы.
  - Заменить каждое логическое выражение на логическую константу «истина» или «ложь».
  - Заменить каждую арифметическую операцию на другую. Например, + на *, - или /.
  - Заменить каждую логическую операцию на другую. Например, > на >=, == или <=.
  - Заменить каждую переменную на другую (из той же области видимости). Переменные должны иметь одинаковые типы.

# МУТАЦИОННОЕ ТЕСТИРОВАНИЕ

- RIP модель

| | |
|---|---|
| REACH | Тест должен достигнуть мутированного оператора. |
| INFECT | Входные данные теста должны привести к разным состояниям программы-мутанта и исходной программы. |
| PROPAGATE | Значение переменной должно повлиять на вывод программы и быть проверено тестом. |

# ПРИМЕР

Исходный код

```
if (a && b) {
    c = 1;}
else {
    c = 0;
}
```

Мутант

```
if (a || b) {
    c = 1;}
else {
    c = 0;
}
```

# МУТАЦИОННОЕ ТЕСТИРОВАНИЕ

- Сильное мутационное тестирование требует выполнение всех трех условий и гарантирует что набор тестов в действительности может обнаружить изменение.
- Слабое мутационное тестирование (или слабое мутационное покрытие) требует выполнение только первых двух условий. Слабое мутационное тестирование тесно связано с методами покрытия кода.

# МУТАЦИОННОЕ ТЕСТИРОВАНИЕ

- Инструменты:
  - PIT mutation
  - muJava
  - Mutator

# HOMEWORK 6

- Account.java
- Формулировка
  - Построить покрытие тестами класса Account(*)
  - Провести мутационное тестирование класса Account:
    - Привести пример убитых мутантов (для каждого тестового метода)
    - Привести пример выжившего мутанта (если будет обнаружен) и изменение в тестах его убивающее.
    - Привести пример эквивалентного мутанта
- Принимается
  - Файл с примерами мутантов
  - Отчет о покрытии(*)

# ЛИТЕРАТУРА


1. Про мутационное тестирование https://habr.com/ru/post/334394/
2. https://pitest.org
3. Пример работы с PIT mutator - https://habr.com/ru/post/139337/
4. Еще про мутационное тестирование http://getbug.ru/mutatsionnoe-testirovanie-na-prostom-primere/
5. JS - https://habr.com/ru/post/341094/
6. Python - https://habr.com/ru/company/vdsina/blog/512630/

# СПАСИБО! ВОПРОСЫ?


НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ