ISSTA 2015 Demo

Feedback-controlled Random Test Generation

Kohsuke Yatoh\textsuperscript{1*}, Kazunori Sakamoto\textsuperscript{2}, Fuyuki Ishikawa\textsuperscript{2}, Shinichi Honiden\textsuperscript{1,2}

\textsuperscript{1}University of Tokyo, \textsuperscript{2}National Institute of Informatics

\textsuperscript{*} He is currently affiliated with Google Inc., Japan. All work is done in Univ. Tokyo and nothing to do with Google.
Introduction

Today's demonstrations:

- Randoop (with multi-pool generator)
  - To generate random tests

- OpenCodeCoverageFramework
  - To instrument Java sources
  - To observe coverage on the fly

All artifacts are available on http://www.klazz.net/pub/issta2015/
Randoop

Random tester for object-oriented programs
[ICSE 07, http://mernst.github.io/randoop/]

Example:

```
class List;
class Int;
```

Example:

```
List l1 = new List();
Int i1 = new Int(0);
l1.add(i1);
l1.addAll(l1);
List l2 = new List();
l2.addAll(l1);
```
Randoop Multi-pool Generator

We wrote 'multi-pool' generator for Randoop.

- Improves coverage of generated tests
  - No change on Input / Output formats

- Tracks coverage on the fly by OpenCodeCoverageFramework

- Detailed technical talk on Friday!
Workflow

1. OpenCodeCoverage Framework

Input
- Class List

Class List (Instrumented)

Output
- Random method sequences

2. Randoop with multi-pool generator
OpenCodeCoverageFramework


- AST(CST) manipulation library for 8+ languages
- Can insert coverage handler function calls
- Not-so-optimized, but very flexible

```c
int abs(int a) {
    if (a < 0) {
        a = -a;
    }
    return a;
}
```

```c
int abs(int a) {
    Hstmt(0);
    if (H.branch(0, a < 0)) {
        Hstmt(1);
        a = -a;
    }
    Hstmt(2);
    return a;
}
```
Demo 1: OpenCodeCoverageFramework

On typical project with ant build file:

# 1. Instrument source files
> $ SimpleOccf.exe src/

# 2. Add soccf-runtime.jar to classpath
> $ vi build.xml

# 3. Compile
> $ ant
**Randoop (multi-pool)**

Creates random method sequences from given class list

- We added 'multi-pool' generator
- We wrapped Randoop by Docker to isolate test environment for safe execution
Demo 2: Randoop (multi-pool) on Docker

# 1. Run randoop (multi-pool) on Docker

> $ python ./run.py \ 
  /path/to/randoop-control.jar \ 
  /path/to/target.jar

# 2. Pull generated regression tests

> $ sudo docker cp \ 
  `cat data/0/cid.txt`:/work .
Summary

Today's demonstrations:

- Randoop (with multi-pool generator)
  - To generate random tests

- OpenCodeCoverageFramework
  - To instrument Java sources
  - To observe coverage on the fly

All artifacts are available on http://www.klazz.net/pub/issta2015/