CSE 503

Software Engineering

Software Testing

Automated Test Generation

Challenges for Automated Testing

```
1 Packet filter(Server s) {
2
3   Packet p = s.readUDP();
4
5   p = ... // filter packet
6
7   return p;
8 }
```

How to write tests for `filter`?

Challenges for Automated Testing

```
1 Random r = new Random();
2
3 int rnd(int a, int b) {
4
5   int n = r.nextInt(a, b);
6
7   return n;
8 }
```

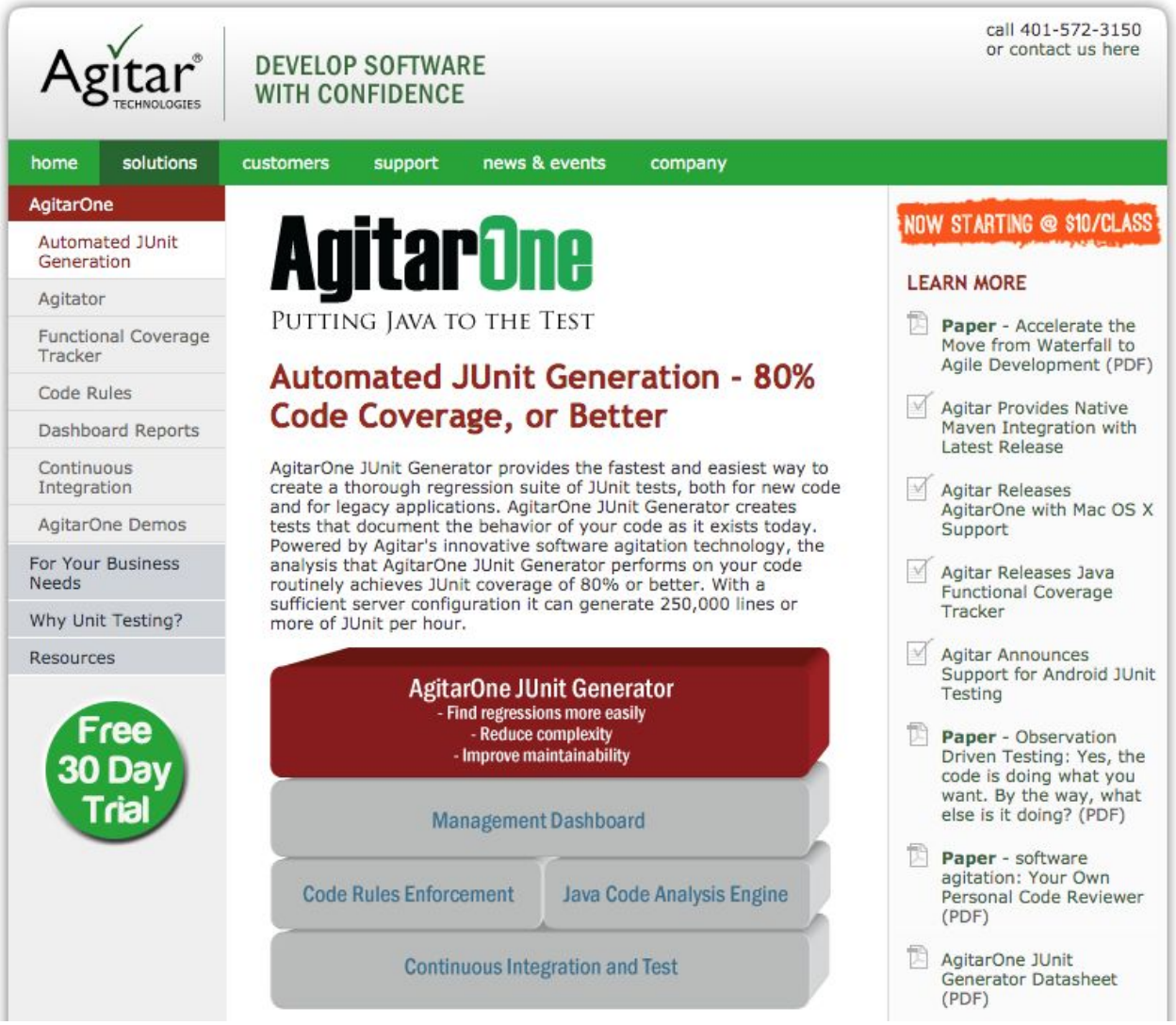
How to write tests for rnd?

Challenges for Automated Testing

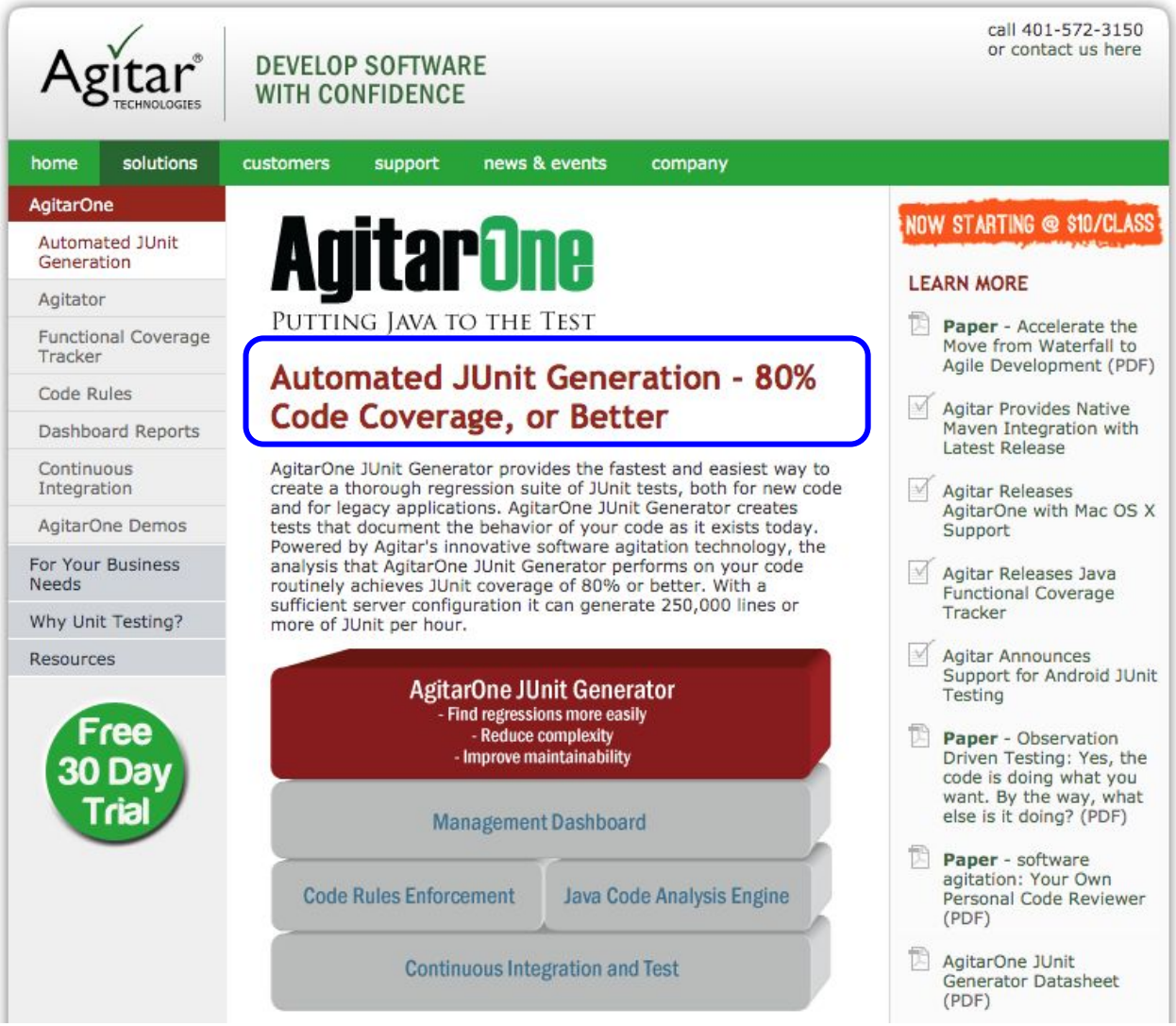
```
1  public List sort(List l) {
2     List s = shuf(l);
3     … // sort the list
4     return s;
5  }
6
7  private List shuf(List l) {
8     List shuf = …;
9     … // shuffle the list
10    return shuf;
11 }
```

How to thoroughly test shuf?

Effectiveness guarantees



Effectiveness guarantees



AgitarOne JUnit Generator provides the fastest and easiest way to create a thorough regression suite of JUnit tests, both for new code and for legacy applications. AgitarOne JUnit Generator creates tests that document the behavior of your code as it exists today. Powered by Agitar's innovative software agitation technology, the analysis that AgitarOne JUnit Generator performs on your code routinely achieves JUnit coverage of 80% or better. With a sufficient server configuration it can generate 250,000 lines or more of JUnit per hour.

Effectiveness guarantees

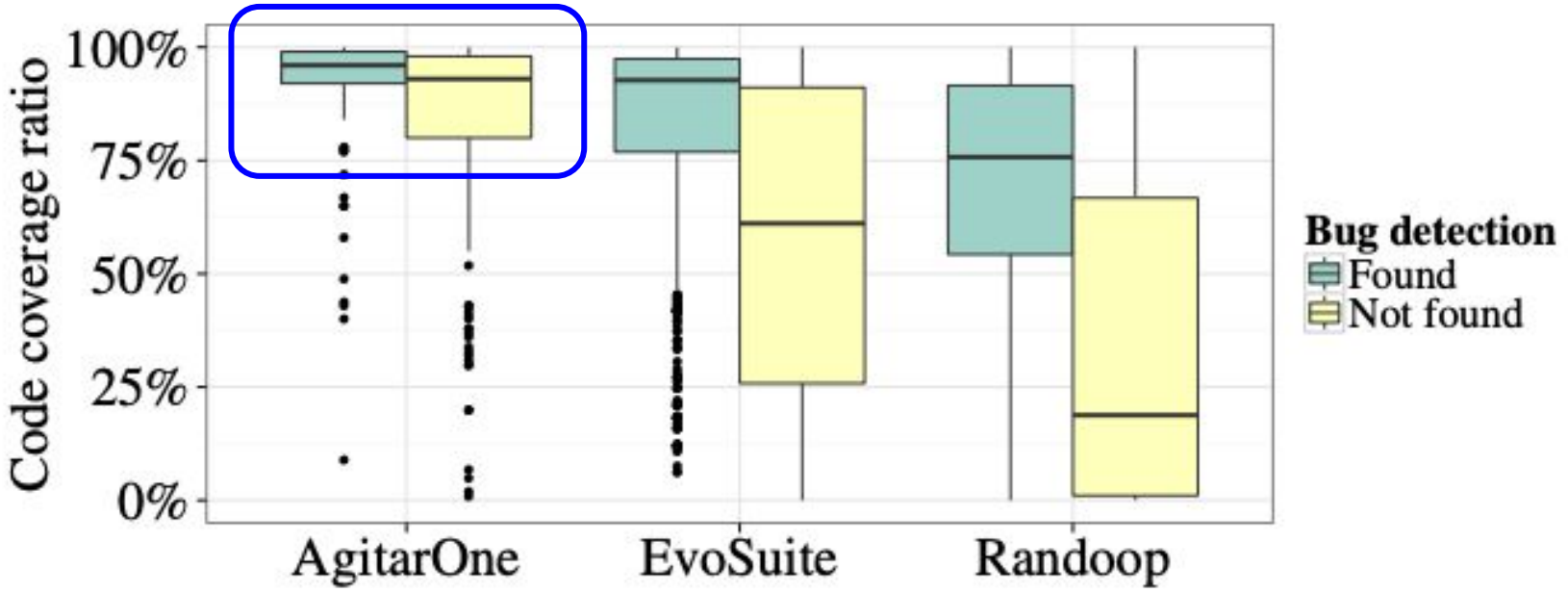


Fig. 3: Code coverage ratios for generated test suites that found a bug and generated test suites that did not. The differences are significant for all tools (Mann-Whitney U test, $p < 0.001$).

Challenges for Automated Test Generation

- Effectiveness (generated tests' ability to detect faults)
- Readability (generated tests' size and clarity)
- Maintainability (co-evolution of code and tests)

Other challenges

“If you ask Randoop to test code that modifies your file system (such as File.delete()), then Randoop will generate tests that modify your file system! Be careful when choosing classes and methods to test.”

Live demo

Instructions

1. git clone https://github.com/rjust/defects4j defects4j
2. cd defects4j
3. ./init.sh
4. ./framework/bin/defects4j checkout -p Lang -v 12f -w ./Lang-12
5. ./framework/bin/gen_tests.pl -g evosuite -pLang -v12f -n1 -o ./gen-tests -b30
6. ./framework/bin/gen_tests.pl -g randoop -pLang -v12f -n1 -o ./gen-tests -b30
7. ./framework/bin/defects4j coverage -w ./Lang-12
   -s./gen-tests/Lang/evosuite/1/Lang-12f-evosuite.1.tar.bz2
8. ./framework/bin/defects4j coverage -w ./Lang-12
   -s./gen-tests/Lang/randoop/1/Lang-12f-randoop.1.tar.bz2
9. (View:
   a. ./Lang-12/src/main/java/org/apache/commons/lang3/RandomStringUtils.java
   b. ./gen-tests/Lang/randoop/1/Lang-12f-randoop.1.tar.bz2
   c. ./gen-tests/Lang/evosuite/1/Lang-12f-evosuite.1.tar.bz2)