CS 520
Theory and Practice of Software Engineering
Spring 2022

In-class exercise 2: Software testing

March 3, 2022
Roadmap

• Unit testing
• Test effectiveness
  – Code coverage
  – Mutation analysis
Unit testing

- **A unit is the smallest testable part** of the software system.

- **Goal:** Verify that each software unit performs as specified.

- **Focus:**
  - Individual units (not the interactions between units).
  - Usually input/output relationships.
JUnit 4: Test – Normative behavior

A single unit test [@Test]

```java
@Test
gpublic void testNewRectangleSatisfiesPrecondition() {
    // Given known inputs
    Assert.assertTrue(WIDTH > 0);
    Assert.assertTrue(HEIGHT > 0);
    // Test on those inputs
    Rectangle rectangle = new Rectangle(WIDTH, HEIGHT);
    // Check for expected output
    Assert.assertNotNull(rectangle);
    Assert.assertEquals(WIDTH, rectangle.getWidth());
    Assert.assertEquals(HEIGHT, rectangle.getHeight());
}
```
JUnit 4: Test – Exceptional behavior

```java
// Check expected output
@Test(expected=IllegalArgumentException.class)
public void testNewRectangleViolatesPrecondition() {
    // Given known inputs
    int negativeWidth = - WIDTH;
    Assert.assertFalse(negativeWidth > 0);
    Assert.assertTrue(HEIGHT > 0);
    // Test on those inputs
    Rectangle newRectangle = new Rectangle(negativeWidth, HEIGHT);
}
```
Test effectiveness

Ratio of detected defects is the best effectiveness metric!

Problem

• The set of defects is unknowable

Solution

• Use a proxy metric, for example code coverage or mutation detection rate
Statement coverage (a.k.a. line coverage)

- Every **statement** in the program must be executed at least once
- Given the control-flow graph (CFG), this is equivalent to node coverage
Decision coverage (a.k.a. branch coverage)

- **Every decision** in the program must take on all possible outcomes (true/false) **at least once**
- Given the CFG, this is equivalent to edge coverage
- **Example:** if \((a>0 \&\& b>0)\)
  - \(a=1, b=1\)
  - \(a=0, b=0\)
Assumption: Mutant detection rate is a good proxy for fault detection rate.
Mutant detection rate

Search for a test case that passes on the original program but fails on the mutant

- If found, the mutant is not equivalent to the original program. This is called a detectable mutant.
- If not found, the mutant is equivalent to the original program. This is called an undetectable mutant.

\[
\text{mutant detection rate} = \frac{\# \text{ detectable mutants}}{\# \text{ all mutants}}
\]
Group selection

- Form 2-, 3-, or 4-person teams
  - If you need more members in your team, raise your hand and ask the instructor
- Use Moodle to self-select a team
- Select a team member responsible for submitting the completed exercise by next Thursday
Set up

1. Make sure that you have Git (v2.7.4 or later) and Java (v8 or later) installed.
   - Git: https://git-scm.com/

   **USEFUL TIP:** Make sure that the JDK comes early in the PATH environment variable. This can fix a lot of issues with modern and multiple JDK installations. For more information, see https://www.java.com/en/download/help/path.xml

2. Clone the cs520 git repository:
   git clone https://github.com/LASER-UMASS/cs520 inclass2