CSCB07 - Software Design

Unit Testing

Fall 2023

What is Software Testing?

- Running a program in order to find faults
  - ➢ Examining the code without execution is not testing
- The main practical approach to validate/verify software
  - ➢ Formal methods that aim at proving the correctness of a program are not scalable
- “Program testing can be used to show the presence of bugs, but never to show their absence!” — Edsger W. Dijkstra

Why Do We Test Software?

- Software is everywhere
  - Communication, transportation, healthcare, finance, education, etc.
- Software failures could have severe consequences
  - A 2002 NIST report estimated that defective software costs the U.S. economy $59.5 billion per year and that improvements in testing could reduce this cost by about a third
  - In certain areas such as healthcare and transportation, software failures could cost lives

Infamous Software Failures

- Northeast blackout of 2003
  - Caused by a failure of the alarm system
  - Affected 40 million people in USA and 10 million people in Canada
  - Contributed to at least 11 deaths
  - Cost around $6 billion
- Ariane 5 explosion (1996)
  - Unhandled floating point conversion exception
  - Estimated loss: $370 million
- NASA's Mars lander (1999)
  - Crashed due to an integration fault
  - Estimated loss: $165 million
- Therac-25 radiation therapy machine
  - Three patients were killed

Testing Levels

- Acceptance testing
  - Test whether the software is acceptable to the user
- System testing
  - Test the overall functionality of the system
- Integration testing
  - Test how modules interact with each other
- Module testing
  - A module is a collection of related units that are assembled in a file, package, or class
  - Test modules in isolation including how the components interact with each other
  - Responsibility of the programmer
- Unit testing
  - Test units (methods) individually
  - Responsibility of the programmer

Unit Testing

- Test a single unit in isolation
- Unit testing process:
  1) Write one or more tests to verify the behavior of the unit
     i. These are called unit tests
     ii. Each unit test is written by specifying an input along with the expected output/state corresponding to that input
  2) Run the tests and check if the actual outputs match the expected ones
- Automation is key

↳ JUnit → run all the tests automatically.

check & fast

JUnit

- A unit testing framework for Java that supports automated test execution and verification
- Tests are organized within JUnit as **test methods** contained within **test classes**
- Test results are validated using assertions
  - E.g. assertEquals, assertTrue, assertNotNull
- Junit makes use of annotations to define and manage tests
  - E.g. @Test, @BeforeEach

assertEquals(a,b);
check if a.equals(b) is true

@BeforeEach
void setup() {
}
call before any
any unit test.

How to build a test set?

test a particular behaviour.

- Black-box approach
  - ➢ Derive tests from external descriptions of the software (i.e. treat the component being tested as a black-box)
- White-box approach
  - ➢ Derive tests from the source code internals of the software

Ammann and Offutt, Introduction to Software Testing, Second Edition, 2017

Criteria-based Test Design

- ***Coverage Criterion:*** A rule or collection of rules that impose test requirements on a test set
  - ➢ E.g. For each statement in the code, there should be at least one test that covers it
- Satisfying a coverage criterion gives a tester some amount of confidence in two crucial goals
  1. We have looked in many corners of the input space, and
  2. Our tests have a fairly low amount of overlap
- Code coverage criteria can be used to quantify the thoroughness of white-box testing

Limitations of Criteria-based Test Design

- In some cases, achieving full coverage might be impossible or unrealistic. For example:
  - Dead code
  - Complex criteria
- For the other cases, full coverage might not necessarily mean that all the bugs will be detected