Introduction to Software Testing

CS 3250
Software Testing

[Ammann and Offutt, “Introduction to Software Testing,” Ch. 1, Ch. 2.1]
Bug?

“‘Bug’ – as such little faults and difficulties are called – show themselves, and months of anxious watching, study, and labor are requisite before commercial success – or failure – is certainly reached.” [Thomas Edison, 1878]


- Bug is used informally.
- Fault? Error? Or failure?
- This course will try to use words that have precise, defined, and unambiguous meaning – and avoid using the term “bug”
Fault, Error, and Failure

• **Fault**: a static defect in the software’s source code
  - Cause of a problem

• **Error**: An incorrect internal state that is the manifestation of some fault
  - Erroneous program state caused by execution of the defect

• **Failure**: External, incorrect behavior with respect to the requirements or other descriptions of the expected behavior
  - Propagation of erroneous state to the program outputs
What is Software Testing?

- **Testing** = process of finding input values to check against a software *(focus of this course)*

  Test case consists of test values and expected results

- **Debugging** = process of finding a fault given a failure

1. Testing is fundamentally about choosing finite sets of values from the input domain of the software being tested

2. Given the test inputs, compare the actual results with the expected results
Testing Categories

Static testing

- Testing without executing the program
  - Software inspection and some forms of analysis
  - Effective at finding certain kinds of problems such as problems that can lead to faults when the program is modified

Dynamic testing

- Testing by executing the program with real inputs
Testing Categories (2)

**White-box testing**
Test with complete information of the software (architecture, how components interact, functions and operations, code, rationale, ...)

**Gray-box testing**
Test with incomplete information of the software or limited knowledge of internal details (may know how components interact but not have detailed knowledge about internal program functions, source code, rationale, ...)

**Black-box testing**
Test from the outside (functionality and behavior). Not require source code, architecture, detailed knowledge about internal program functions, rationale, ...
### Testing Categories (3)

#### Functional testing
- Unit testing
- Integration testing
- System testing
- Smoke testing
- Interface testing
- Beta/acceptance testing
- Agile testing
- Regressing testing
- Continuous integration testing

... and many more ...

#### Non-functional testing
- Performance testing
- Load testing
- Stress testing
- Security testing
- Compatibility testing
- Reliability testing
- Usability testing
- Compliance testing

... and many more ...
Validation and Verification (IEEE)

• **Validation**: The process of evaluating software at the end of software development to ensure compliance with intended usage
  - Not done by developers, but by experts in the intended usage of the software

• **Verification**: The process of determining whether the products of a given phase of the software development process fulfill the requirements established during the previous phase
  - Requires technical background on the software, normally done by developers at the various stages of development

IV&V stands for “independent verification and validation”
Testing and SW Development Lifecycle

Requirements Analysis

Architectural Design

Subsystem Design

Detailed Design

Implementation

System Test

Integration Test

Module Test

Unit Test

Acceptance Test

- Check if software does what the user needs
- Check overall behavior w.r.t. specs
- Check interface between modules in the same subsystem
- Check interactions of units and associated data structures
- Check each unit (method) individually

[based in part on AO, p.23]
Test Process Maturity

Beizer’s scale for test process maturity

- **Level 0**: There is no difference between testing and debugging
- **Level 1**: The purpose of testing is to show correctness
- **Level 2**: The purpose of testing is to show that the software does not work
- **Level 3**: The purpose of testing is not to prove anything specific, but to reduce the risk of using the software
- **Level 4**: Testing is a mental discipline that helps all IT professionals develop higher quality software

[AO, p.9]
Level 0 – Testing is Debugging

- Level 0: Testing is the same as debugging

- Not distinguish between incorrect behavior and mistakes in the program

- Not help develop software that is reliable

- Adopted by most CS students 😞
  - Get programs to compile
  - Debug with few arbitrarily chosen inputs or those provided by instructors

Levels 1 – Software Works

- **Level 1**: Purpose is to show correctness (developer-biased view)

- Correctness is impossible to establish or demonstrate

- What do we know if “no failures”?
  - Good software?
  - Bad tests?

- No strict goal, no stopping rule or formal test technique

- No quantitatively way to evaluate; test managers are powerless

Levels 2 – Software Doesn’t Work

- **Level 2**: Purpose is to show failure (tester-biased view)
  - A negative view puts testers and developers into an adversarial relationship – bad team morale
  - What do we know if “no failures”? Good software? Bad tests?

“Mature” Testing

Correctness cannot generally be achieved or demonstrated through testing.

Testing can only show the presence of failure, not the absence.

Developers and testers should be on the same boat.
Levels 3 – Risk Reduction

- **Level 3**: Purpose is to **reduce the risk** of using the software

  - There are risks when using software
    - Some may be small with unimportant consequences
    - Some may be big with important consequences, or even catastrophic

- Testers and developers cooperate to reduce risk
Levels 4 – Quality Improvement

- **Level 4**: Purpose is to **increase quality** of the software
  - Testing should be an integral part of the development process
  - Testers become technical leaders, measuring and improving software quality
    - Help developers improve the ability to produce quality software
    - Train developers
  - Testers and developers cooperate to improve the quality

- Example: Spell checker
  - Purpose: to improve our ability to spell
  - Change of mindset:
    - “find misspelled words” → “improve ability to spell”
How “Mature” is Your Testing?

Are you at level 0, 1, 2, 3, or 4?

We hopes to teach you to become “change agents” (level 4)
Tactical Goals: Each Test

“If you don’t know why you’re conducting each test, it won’t be very helpful” – Jeff Offutt

- What is objective and requirement of each test?
- What fact does each test try to verify?
- What are the threshold reliability requirements?
- What are the planned coverage levels?
- How many tests are needed?
Wrap-up

- A tester’s goal is to eliminate faults as early as possible

- Testing
  - Improves software quality
  - Reduce cost
  - Preserve customer satisfaction

- What’s next?
  - Fault, error, failure
  - Reachability, Infection, Propagation, and Revealability (RIPR) model