Software Testing
Introduction

CS 4501 / 6501
Software Testing

[Ammann and Offutt, “Introduction to Software Testing”]
Software is Everywhere
“ ‘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
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
Software Failures

- 2017: 606 recorded software failures, impacting 3.7 billion people, 314 companies, $1.7 trillion in financial losses

- 2016: Nissan recalled 4 millions cars from the market due to software failure in the airbag sensory detectors.

- 2016: Info lost due to the browser back button while using TurboTax web software

- 2015: Bloomberg’s trading terminal failures forced the British government to postpone $4.4 billion debt sale

- 2014: Dropbox’s outage was due to a fault in a maintenance script

- 2012: Faults in a new Knight Capital’s trading software causes $440 millions

- 2003: Northeast blackout due to the alarm system in the energy management system failure, affecting 40 million people in 8 US states, 10 million people in Ontario, Canada

- 1999: NASA’s Mars lander crashed due to a unit integration fault

- 1997: Ariane 5 explosion: Exception-handling bug forced self-destruct on maiden flight (64-bit to 16-bit conversion), causing $370 millions
How Important is Testing?

History of Software Testing

What? I've done the coding and now you want to test it. Why? We haven't got time anyway.

OK, maybe you were right about testing. It looks like a nasty bug made its way into the Live environment and now costumers are complaining.

Testers! You must work harder! Longer! Faster!

1960s - 1980s
Constraint

1990s
Need

2000+
Asset
Testing in the 21st Century

- Safety critical, real-time software
- Embedded software
- Enterprise applications
- Security
- Web
- Mobile

Software testing is getting more important

What are we trying to do when we test? What are our goals?
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
Static and Dynamic Testing

• **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
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

Test → Design → Information

Acceptance Test → System Test → Integration Test → Module Test → Unit Test

[AO, p.23]

Validation → Verification
Testing Goals Based on 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 – Debug Only?

- **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
Levels 1, 2 – Developer vs Tester?

- **Level 1**: Purpose is to show correctness (developer-biased view)
  - Correctness is impossible to establish or demonstrate

- **Level 2**: Purpose is to show failure (tester-biased view)
  - A negative view puts testers and developers into an adversarial relationship
Levels 3, 4 – “Mature” Testing

- **Level 3**: Purpose is to show the presence of failures
  - There are risks when using software
  - Testers and developers cooperate to reduce risk

- **Level 4**: Purpose is to increase quality of the software
  - Testing should be an integral part of the development process
  - Testers and developers cooperate to improve the software quality

How mature is your testing?
Are you at level 0, 1, 2, or 3?
We hopes to teach you to become “change agents” (level 4)
Summary

• Testing is the most time consuming and expensive part of software development

• Not testing is even more expensive

• Having too little testing effort early increases the testing cost

• Planning for testing after develop is prohibitively expensive

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

• Testing improves software quality, reduce cost, and preserve customer satisfaction
What’s Next?

• Fault, error, failure

• Reachability, Infection, Propagation, and Revealability (RIPR) model