Software Engineering: Software Testing and Verification

Lecture 1
CS295

Course Staff
- Instructor
  - Alex Aiken
- Teaching Assistant
  - Steven Elia

Course Communication
- All class materials will be on the web
  - Lecture notes, handouts, papers to read, etc.
- Read the web site for announcements
- Ask questions in the newsgroup
  - Preferred for most questions over email

Course Structure
- Lectures
  - Course taught from notes
  - Focus on programmer's view of software engineering
    - Technology over business issues
- Homeworks
  - For those taking the class for 3 credits
  - Use a verification/testing tool, report findings
- In-class midterm, take-home final

Why Take This Class?
- Be a better programmer
- See the future
  - This is where the field is headed
- Prepare for research in the area
- For the war stories

This Lecture
The course is about improving software quality.

Claim: This is central to software engineering.
The Ariane Rocket Disaster

Post Mortem
- Failure due to unhandled floating point exception
- Cost
  - $100M's for loss of mission
  - Multi-year setback to the Ariane program

East Coast USA

East Coast USA: 2003 Blackout

Post Mortem
- Local failure rapidly cascaded through grid
- Major contributing cause was unnoticed crash of automated alarm systems
- 10M's of people affected

Mars Polar Lander
### Post Mortem

- A units problem
  - Caller expected values in inches/feet
  - Callee assumed values in meters
  - Essentially, a type error
- Total loss of $100M mission

### Security

- Often exploit bugs in programs
- Widespread problem, getting worse
  - Code Red
  - Titan Rain
  - Moonlight Maze
  - Operation Orchard
  - Stuxnet Worm

### The Big Question

How do we know the code works?

Look first at how software is developed . . .

### Software Development Today

Why do we have this structure?

![Software Development Diagram]

### Typical Scenario (1)

- Decision Maker: "OK, calm down. We'll slip the schedule. Try again."
- Programmer: "I'm done."
- Tester: "It doesn't compile!"

### Typical Scenario (2)

- Decision Maker: "Now remember, we're all in this together. Try again."
- Programmer: "I'm done."
- Tester: "It doesn't install!"
Typical Scenario (3)

Let’s have a meeting to straighten out the spec.

I’m done.

No, half of your tests are wrong!

Typical Scenario (4)

Try again, but please hurry up!

I’m done.

It still fails some tests we agreed on.

Typical Scenario (5)

Oops, the world has changed. Here’s the new spec.

I’m done.

Yes, it’s done!

Software Development Today

Why do we have this structure?

Independent Testing and Development

Testing is basic to every engineering discipline
- Design a drug
- Manufacture an airplane
- Etc.

Why?
- Because our ability to predict how our creations will behave is imperfect
- We need to check our work, because we will make mistakes

Key Assumptions

- Independent development and testing
- Specifications must be explicit
- Specifications evolve
- Resources are finite
- Human organizations need decision makers
- Examine each of these separately
Independent Testing and Development of Software

- In what way is software different?
- Folklore:
  - "Programmers are optimists"
  - The implication is that programmers make poor testers
  - Economics: "Programming costs more than testing"
  - The implication is that programming is a higher-skill profession
- How valid is the folklore, and how much is due to the current state of the art in testing?

Explicit Specifications

- Software involves multiple people
  - At least a programmer and a user
  - But usually multiple programmers, testers, etc.
- Any team effort requires mutual understanding of the goal
  - A specification
  - Otherwise, team members inevitably have different goals in mind

Specifications Change

- Why?
- Many software systems are truly "new"
  - Differ from all that went before in some way
  - Initial specification will change as problems are discovered and solved
- The world is changing
  - What people want
  - The components you build on (e.g., the OS version)

Software Specifications

- Software specifications are usually
  - in prose
  - imprecise
  - out of date
- Current state of specification is not conducive to automation
  - Not consumable by tools
  - Without a spec, there is nothing to check

Finite Resources

- Organizations make trade-offs
  - Not all goals can be achieved
  - Because resources are finite
- $s express relative costs among goals
  - Goals that are hard to quantify pose a problem
  - E.g., correctness, completeness

"We have 2 months, 5 programmers, and 2 testers. Here is a priority list of features. A feature is finished when it passes all of the tests for that feature; a programmer does not move on to a new feature until all higher priority features are finished or assigned to other programmers. We start now and ship whatever features are finished in 60 days."

Reality

- Many proposals for improving software quality
- But the world tests
  - > 50% of the cost of software development
- Conclusion
  - Testing is important
  - Take a closer look at testing practice . . .
The Purpose of Testing

- Two purposes:
  - Find bugs
    - Find important bugs
  - Elucidate the specification

Specifications

- Good testers clarify the specification
  - This is creative, hard work
- There is no hope tools will automate this
  - This part will stay hard work
- An extreme example
  - Warning: cheap laughs
  - "AOL spilled my coffee"

Example

- Test case
  - Add a child to Mary Brown's record
- Version 1
  - Check that Ms. Brown's # of children is one more
- Version 2
  - Also check Mr. Brown's # of children
- Version 3
  - Check that no one else's child counts changed

Manual Testing

- Test cases are lists of instructions
  - "test scripts"
- Someone manually executes the script
  - Do each action, step-by-step
    - Click on "login"
    - Enter username and password
    - Click "OK"
    - ...
    - And manually records results
- Low-tech, simple to implement

Manual Testing

- Those are the best reasons
- There are also not-so-good reasons
  - Not-so-good because innovation could remove them
  - Testers aren't skilled enough to handle automation
  - Automation tools are too hard to use
  - The cost of automating a test is 10X doing a manual test

Manual Testing

- Manual testing is very widespread
  - Maybe not dominant, but very, very common
- Why? Because
  - Some tests can't be automated
    - Usability testing
  - Some tests shouldn't be automated
    - Not worth the cost
Automated Testing

- **Idea:**
  - Record manual test
  - Play back on demand
- **This doesn’t work as well as expected**
  - *E.g.*, Some tests can’t/shouldn’t be automated

Fragility

- Test recording is usually very fragile
  - Breaks if environment changes anything
  - *E.g.*, location, background color of textbox
- **More generally, automation tools cannot generalize**
  - They literally record exactly what happened
  - If anything changes, the test breaks
- **A hidden strength of manual testing**
  - Because people are doing the tests, ability to adapt tests to slightly modified situations is built-in

Breaking Tests

- When code evolves, tests break
  - *E.g.*, change the name of a dialog box
  - Any test that depends on the name of that box breaks
- Maintaining tests is a lot of work
  - Broken tests must be fixed; this is expensive
  - Cost is proportional to the number of tests
  - Implies that more tests is not necessarily better

Improved Automated Testing

- Recorded tests are too low level
  - *E.g.*, every test contains the name of the dialog box
- **Need to abstract tests**
  - Replace dialog box string by variable name *X*
  - *Variable name X is maintained in one place*
    - To ensure that when the dialog box name changes only *X needs to be updated and all the tests work again*
- This is just structured programming
  - *Just as hard as any other system design*
  - Really, a way of making the specification more concise

Discussion

- **Automated testing is the state of the art**
- **Testers have two jobs**
  - Clarify the specification
  - Find (important) bugs
- Only the latter is subject to automation
- Helps explain why there is so much manual testing

Summary

- **How do we know the code works?**
  - Compare the code with a specification
  - We need a specification!
- **Testing is important**
  - But very expensive
  - And clearly not sufficient
- **We can do better!**
  - Many useful techniques
  - The topic of this course