Why Testing?

• Goal: Increase software reliability
  – Software bugs cost US economy $60B/year [NIST’02]

• Approach: Find bugs using testing
  – Estimated savings from better testing $22B/year

• Challenge: Manual testing is problematic
  – Time-consuming, error-prone, expensive

• Research: Automate testing
  – Reduce cost, increase benefit
Some Costly “Bugs”

• NASA Mars space missions  
  – Different metric systems (1999)
• BMW airbag problems (1999)  
  – Recall of >15000 cars
• Ariane 5 crash (1996)  
  – Uncaught exception of numerical overflow  
  – Sample Video
• Your own favorite example?
Testing Effort

• Reported to be >50% of development cost [e.g., Beizer 1990]

• Microsoft: 75% time spent testing
  – 50% testers who spend all time testing
  – 50% developers who spend half time testing
Introduction

• Why look for bugs?
• What are bugs?
• Where they come from?
• How to find them?
Terminology

• Anomaly
• Bug
• Crash
• Defect
• Error
• Failure, fault
• Glitch
• Hangup
• Incorrectness
• J…
“Bugs” in IEEE 610.12-1990

• **Fault**
  – Incorrect lines of code

• **Error**
  – Faults cause incorrect (unobserved) state

• **Failure**
  – Errors cause incorrect (observed) behavior

• Not used consistently in literature!
Correctness

• Common (partial) properties
  – Segfaults, uncaught exceptions
  – Resource leaks
  – Data races, deadlocks
  – Statistics based

• Specific properties
  – Requirements
  – Specification
When to Test

• The later a bug is found, the higher the cost
  – Orders of magnitude increase in later phases
  – Also the smaller chance of a proper fix

• Old saying: test often, test early

• New methodology: test-driven development
  (write tests before code)
Software is Complex

- Malleable
- Intangible
- Abstract
- Solves complex problems
- Interacts with other software and hardware
- Not continuous
Software Still Buggy

• Folklore: 1-10 (residual) bugs per 1000 nbnc lines of code (after testing)

• Consensus: total correctness impossible to achieve for (complex) software
  – Risk-driven finding/elimination of bugs
  – Focus on specific correctness properties
Approaches for Finding Bugs

• Software testing
• Model checking
• (Static) program analysis
Software Testing

• Dynamic approach
• Run code for some inputs, check outputs
• Checks correctness for some executions

• Main questions
  – Test-input generation
  – Test-suite adequacy
  – Test oracles
Other Testing Questions

- Maintenance
- Selection
- Minimization
- Prioritization
- Augmentation
- Evaluation
- Fault Characterization
- ...

Model Checking

• Typically hybrid dynamic/static approach
• Checks correctness for “all” executions

• Some techniques
  – Explicit-state model checking
  – Symbolic model checking
  – Abstraction-based model checking
Static Analysis

- Static approach
- Checks correctness for “all” executions

- Some techniques
  - Abstract interpretation
  - Dataflow analysis
  - Verification-condition generation
Comparison

• Level of automation
  – Push-button vs. manual

• Type of bugs found
  – Hard vs. easy to reproduce
  – High vs. low probability
  – Common vs. specific properties

• Type of bugs (not) found
Soundness and Completeness

• Do we find all bugs?
  – Impossible for dynamic analysis

• Are reported bugs real bugs?
  – Easy for dynamic analysis

• Most practical techniques and tools are both unsound and incomplete!
  – False positives
  – False negatives
Analysis for Performance

• Static compiler analysis, profiling
• Must be sound
  – Correctness of transformation: equivalence
• Improves execution time
• Programmer time is more important

• Programmer productivity
  – Not only finding bugs
Combining Dynamic and Static

• Dynamic and static analyses equal in limit
  – Dynamic: try exhaustively all possible inputs
  – Static: model precisely every possible state

• Synergistic opportunities
  – Static analysis can optimize dynamic analysis
  – Dynamic analysis can focus static analysis
  – More discussions than results
Current Status

- Testing remains the most widely used approach for finding bugs.
- A lot of recent progress (within last decade) on model checking and static analysis:
  - Model checking: from hardware to software
  - Static analysis: from sound to practical
- Vibrant research in the area
- Gap between research and practice
Topics Related to Finding Bugs

• How to eliminate bugs?
  – Debugging

• How to prevent bugs?
  – Programming language design
  – Software development processes

• How to show absence of bugs?
  – Theorem proving
  – Model checking, program analysis