Why do we care?

- Therac-25 (1985)
- Patriot missile (1991)
- Ariane V (1996)
- Millenium bug (2000)
- Microsoft attacks (2003)

- NIST: cost to US, $59 billion
Quality and testing

- “Errors should be found and fixed as close to their place of origin as possible.” Fagan

- “Trying to improve quality by increasing testing is like trying to lose weight by weighing yourself more often.” tMcConnell
Life Testing

- Used regularly in hardware
  - Addresses “normal use”

- n specimens put to test
- Test until r failures have been observed
- Choose n and r to obtain the desired statistical errors
- As r and n increase, statistical errors decrease
- Expected time in test = \( \mu_0 \frac{r}{n} \)
  - Where \( \mu_0 = \) mean failure time
Butler and Finelli

“The Infeasibility of Experimental Quantification of Life-Critical Software Reliability”

In order to establish that the probability of failure of software is less than $10^{-9}$ in 10 hours, testing required with one computer is greater than 1 million years.
Tools for Improving Quality

- Formal specification
- Self-checking (paranoid) code
- Program verification and validation
- Testing
- Deploy with capabilities to repair
Types of Testing: Purpose

- Conformance testing
- Usability testing
- Performance testing
- Acceptance testing
- Reliability testing
- ...

...
Other classifications

- Scope
  - Unit, component, system, regression, ...

- Access to code
  - Black box vs. white box
  - (Note that black box testing still assumes knowledge of coding and development in general)
What are you trying to test?

- Most common actions?
- Most likely problem areas?
- Risk-based testing
Risks

- Identify criteria of concern: availability, quality, performance, ...
- Risk of it not being met
  - likelihood
  - consequences
- If I’m testing code for a grocery store, what is the impact of the code not being highly available?
Risk Heuristics (just a few)

- New features
- New technology
- Overworked developers
- Regression
- Dependencies
- Complexity
- Bug history
- Language specific bugs
- Environment changes
- Late changes
- Slipped in “pet” features
- Ambiguity
- Changing requirements
- Bad publicity
- Liability
- Learning curve
- Criticality
- Popularity
Who should test code?

- Unit test is always done by developer
- Different views on system test
  - Build to test
    - Test-driven development
  - Completely independent
    - Acceptance testing
  - Middle-of-the-road
Four Parts of Testing

- Model
- Select test cases
- Execute test cases
- Measure
Basic Software Model

- User Interfaces
- APIs
- Operating system
- Files
- Input
- Output
- Storage
- Processing

---

- Environment
- Capabilities

---
Test Case Selection
From the User Interface: Inputs

- Error messages
- Default values
- Character sets and data types
- Overflow input buffers
- Input interactions
- Repeated inputs
From the User Interface: Outputs

- Concept: What inputs create interesting outputs?
  - REQUIRES DOMAIN EXPERTISE

- State-based outputs
- Invalid outputs

- Changeable outputs
- Screen refreshes
Bubble diagrams:
Reverse state exploration

- Define a failure state
- What would have happened to get you there?
- Repeat

- Find a way to force it down that path

- Let’s try it: BMW key fobs
Capabilities – Storage and Processing

- Same input, different initial conditions
- Too many or too few values in a data structure
- Alternative ways to modify constraints
- Invalid operator/operand combinations
- Recursive functions
- Overflow/underflow computations
- Feature interaction
How to select specific cases

- Data based
  - Boundary conditions
  - Equivalence classes
- Control based
  - State transitions
Execution
Execution tools

- Automation – often primarily scripts
- Critical for regression testing
- GUI tools are abundant, but marginal
Measurement
Test Coverage Metrics

- Statement coverage
  - basic block coverage
- Decision coverage
  - Each Boolean expression
- Condition coverage
  - Each entity in Boolean expressions
- Path coverage
  - loops

- Advantages of different models?
Estimating how many bugs are left

- Historical data
- Capture-recapture model from biology
A variant: exploratory testing
Testing Approaches

- Analytical
- Information-driven
- Intuitive
- Exploratory
  - Design the tests and test concurrently
  - Learn the system and test it as you go
  - Structure creative testing
  - Think while testing
  - Risk-based
  - Analogous to Extreme Programming
Exploratory Testing Tasks

- Explore
  - Elements of the product
  - How the product should work
- Design Tests
  - Which elements
  - Speculate on possible quality problems
- Execute Tests
  - Observe behavior
  - Evaluate against expectations

All with test design techniques best suited for the product
Exploratory Testing Practice

- Used to probe for weak areas
- Especially useful when
  - Weak specifications and requirements
  - Little domain knowledge
  - Time pressures
- Less appropriate when
  - Well-defined test requirements
  - Strong need for regression testing
  - Repeatable over releases
    - Cost of maintenance
    - Few new test cases
Planning

- Decompose the product into elements
  - Areas of function that you can test in 1-2 days
  - Define charters
    - Decomposition into units that can be tested in 1-2 hours
- Quality criteria
  - Capability, reliability, usability, performance, installability, compatibility, ...
- Select test techniques
Charter

- Provides clear mission of why this test
- Suggests what and how it should be tested, as well as problems to look for
- Not a detailed plan, but should be as specific as possible
- Might include risks, documents and desired output
References

- Whittaker, *How to Break Software*
- Kaner, *The Impossibility of Complete Testing* at [www.kaner.com](http://www.kaner.com) (Articles)
References

- Patriot missile: http://www.fas.org/spp/starwars/gao/im92026.htm
- Ariane 5: http://www.esa.int/export/esaCP/Pr_33_1996_p_EN.html