Software Quality Assurance

Presented By
Trina Saha
Lecturer, City University

What is Software Quality? (The Model)

The Ideal Picture:

- Determining & Achieving requirements
- Adequately staffed & enough time to do the work
- Quality Assurance presence in every phase of the development process, from requirements definition to final testing.
- Management's Commitment to quality *on the unquestioned faith that it is always worth whatever it will cost*.

The Real Picture:

Requirements Shift and Waver
Perpetually Understaffed and Behind Schedule
Software Quality Assurance is often a Fancy word for Ad-hoc testing
First one to market wins the most market share
Management more interested in making more money than in the niceties and the necessities of the Software Engineering

What is Software Quality?

(Various Views)

Manufacturing View

Quality is Correctness, Conformance to Specifications

Problem:
Perfect Products that satisfy no one

Aesthetic View

Quality is Elegance, an ineffable experience of Goodness

Problem:
Perfectionists & Underachievers

Customer View

Quality is Fitness of use, whatever satisfies me

Problem:
Chasing will-o'-the-wisp

How much Software Quality?

(Measures)

- Some Measurable Factors

  Functionality, Reliability, Usability, Efficiency, Maintainability, Portability

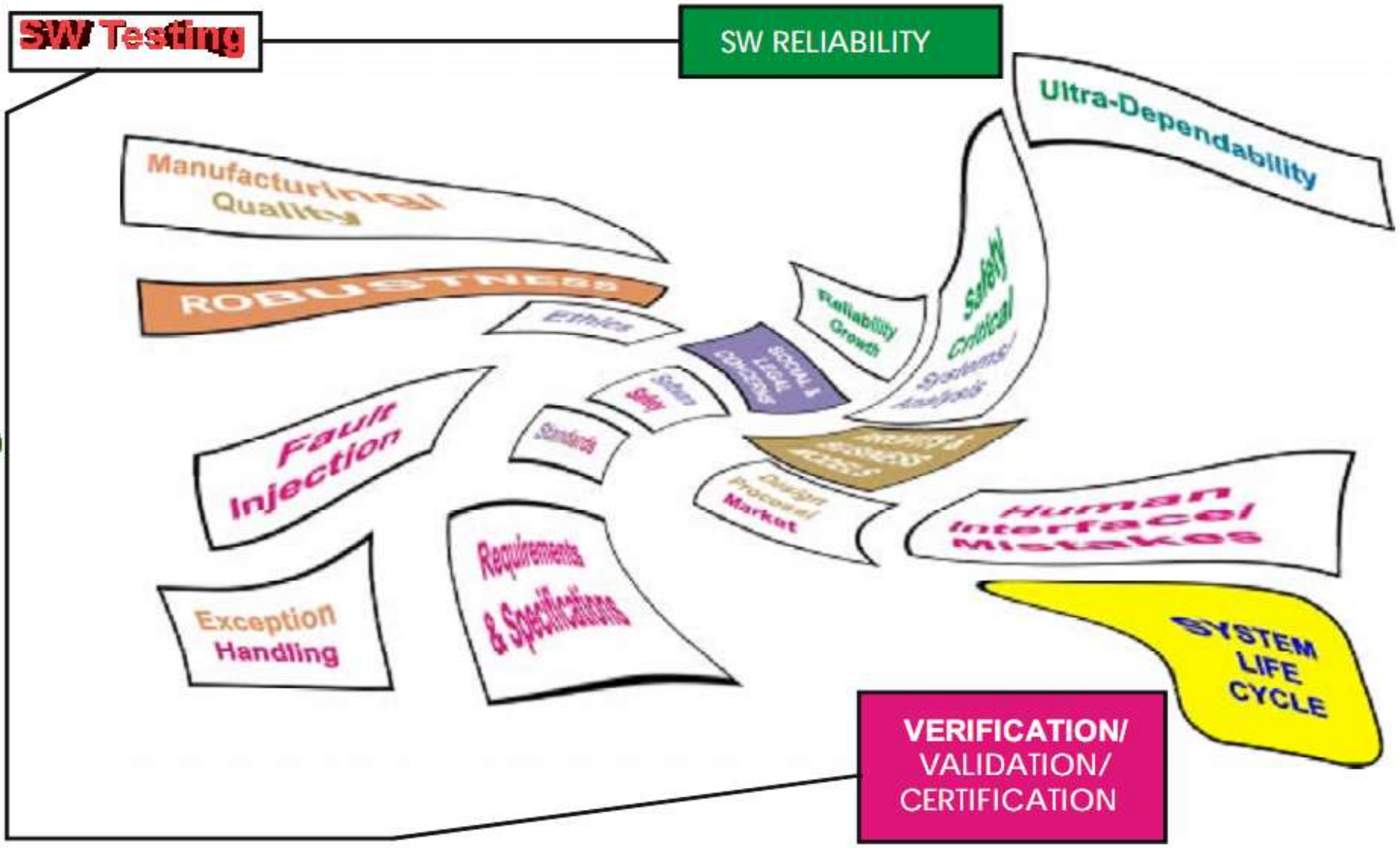
- BUT !!
  - Some factors are more important or detract from others
  - There is no straight forward approach to measure these
  - No matter how each of these factors is taken into account a single BUG may negate everything else that is working right
  - FACT: "The Client" can never know the Quality of the Project…
  - They make Perceptions about Quality based on skill level, past experience and profile of use and we can not control any of these basis

How much Software Quality?

- **SINCE**
  - Creating products of the best possible quality is very expensive
  - The Client may not even notice the difference between the best possible quality and pretty good quality
- **3 Critical Questions**
  - **How much of which quality factor be adequate?**
  - **How do we measure it adequately?**
  - **How do we control it adequately?**
- **The Solution**
  - **Instead of creating a universal metric of quality and then optimizing it, rather consider the problems directly what quality is supposed to solve**

You Are Here

- A lot of subtle relations to other topics



Introduction

- **Definitions of Software Testing**
  - [1]: Testing is the process of executing a program or system with the intent of finding errors.
  - [3]: Testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.
- **Vocabulary & Concepts**
  - Defects, bugs, faults[ANSI], errata[Intel]
  - Testing is more than debugging[BEIZER90]
- **Software testing is an ……** ART
  - because we still can not make it a science
- **Software testing is everywhere**
  - in every phase of software life cycle, whenever software changes
  - 50%+ time in debugging/testing
- **Software testing is not mature**

Who Tests the software?

developer

understands the system but, will test "gently" and, is driven by "delivery"

independent tester

must learn about the system, but, will attempt to break it and, is driven by equality

Why testing?

- **For Quality**
  - **bugs kill**
    - in a computerized embedded world
  - Defect detection (find problems and get them fixed [KANER93])
    - Better early than late
      - » Difficult to upgrade field software in embedded systems
  - To make quality visible [HETZEL88]
- **For Verification & Validation(V&V):**
  - show it works:
    - clean test/positive test
  - or it can handle exceptional situations:
    - dirty test/negative test
- **For Reliability Estimation [KANER93]**
  - E.g. reliability growth testing

Why software testing is difficult -- principles

- **Software fails in different ways with physical systems**
- **Imperfection of human nature(to handle complexity)**
- **Cannot exterminate bugs**
  - We cannot test a typical program completely
  - The Pesticide Paradox[BEIZER90]
    - Every method you use to prevent or find bugs leaves a residue of subtler bugs against which those methods are ineffectual.
    - Fixing the previous(easy) bugs will tend to increase software complexity --> introducing new subtler bugs
  - The Complexity Barrier[BEIZER90]
    - Software complexity(and therefore that of bugs) grows to the limits of our ability to manage that complexity.

Software Testing: Taxonomy

- **By purposes**
  - Correctness testing
    - Black-box
    - White-box
  - Performance testing
  - Reliability testing
    - Robustness testing
      - » Exception handling testing
      - » Stress/load testing
  - Security testing
- **By life cycle phase[PERRY95]**
  - Requirements phase testing
  - Design phase testing
  - Program phase testing
  - Evaluating test results
  - Installation phase testing
  - Acceptance testing
  - Testing changes: maintenance
- **By scope**
  - implied in [BEIZER95]
    - Unit testing
    - Component testing
    - Integration testing
    - System testing
  - or in [PERRY90]
    - Unit testing
    - String testing
    - System testing (α test)
    - Acceptance testing (β test)

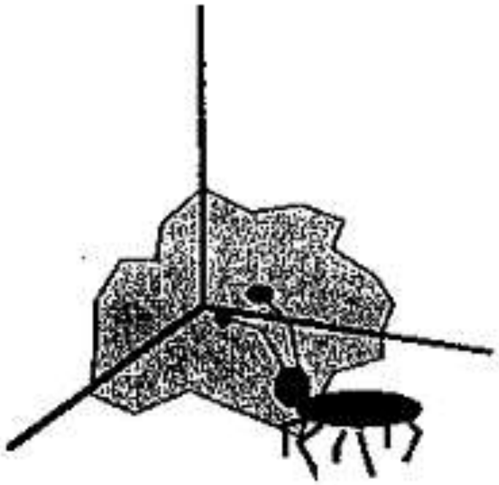
A good test

- Has a high probability of finding an error
- is not redundant – testing time and resources are limited
- Should be a “best of breed” – the test that has the highest likelihood of uncoveringa whole class of errors should be used
- Should be neither too simple nor too complex

Test Case Design

"Bugs lurk in corners and congregate at boundaries ..."

Boris Beizer

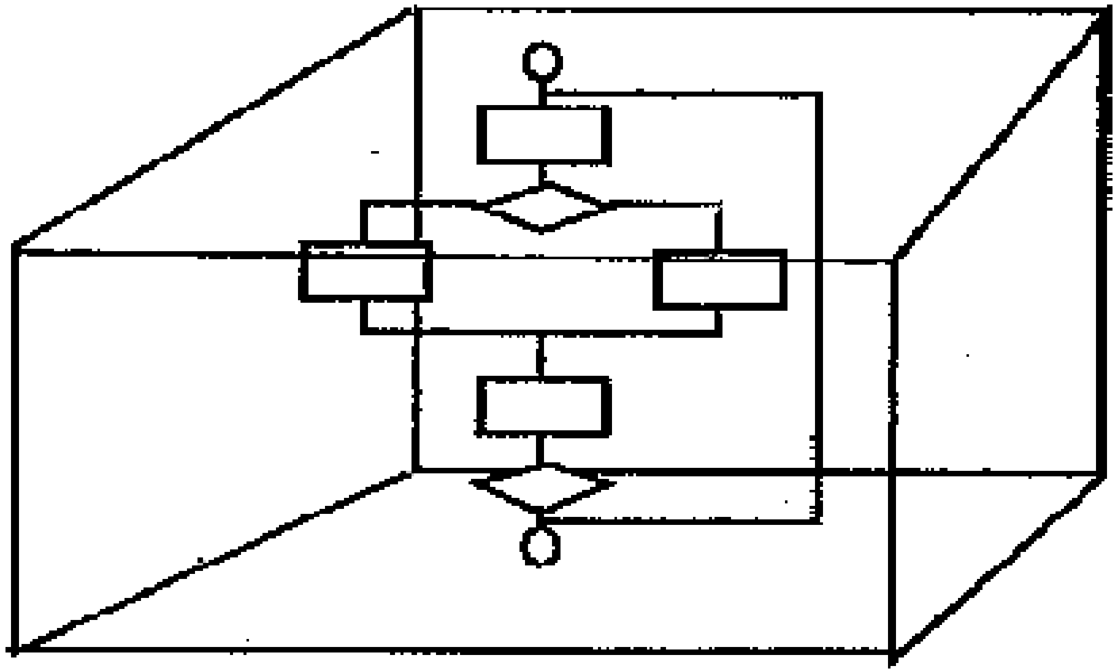


OBJECTIVE	to uncover errors
CRITERIA	in a complete manner
CONSTRAINT	with a minimum of effort and time

Test Case Design

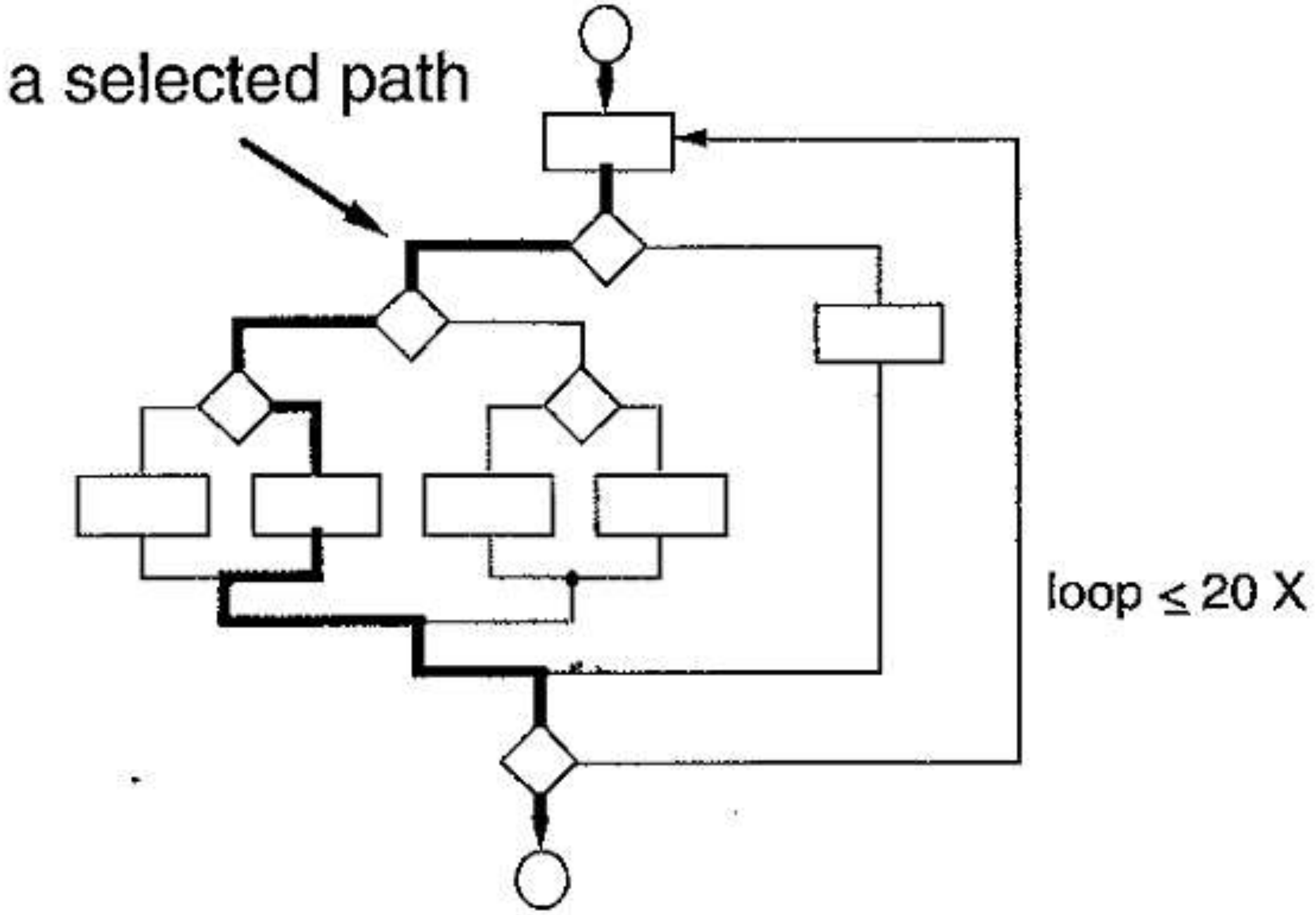
- Black-box testing
  - Conducted at the SW interface
  - Used to demonstrate that Sw functions are operational; that input is properly accepted and output is correctly produced, and thar the integrity of the SW is maintained
- White-box testing
  - close examination of procedural details.
  - Logical paths through the SW is tested by providing test cases that exercise sprecific sets of conditions andor loops.

White-Box Testing



... Our goal is to ensure that all statements and conditions have been executed at least once ...

Selective Testing



White-Box Testing methods

1. Guarantee that all *independent paths* within a module have been exercise at least once
2. Exercise all logical decisions on their *true* and *false* sides
3. Execute all loops at their boundaries and within their operational bounds
4. Exercise internal data structures to assure their validity

Why Cover?

- logic errors and incorrect assumptions are inversely proportional to a path's execution probability
- we often believe that a path is not likely to be executed; in fact, reality is often counter intuitive
- typographical errors are random; it's likely that untested paths will contain some

Verification and Validation

- Verification - Are we building the product right?
- Validation – Are we building the right product ?

Testing Strategy

- System Engineering
- Requirements
- Design
- Code →

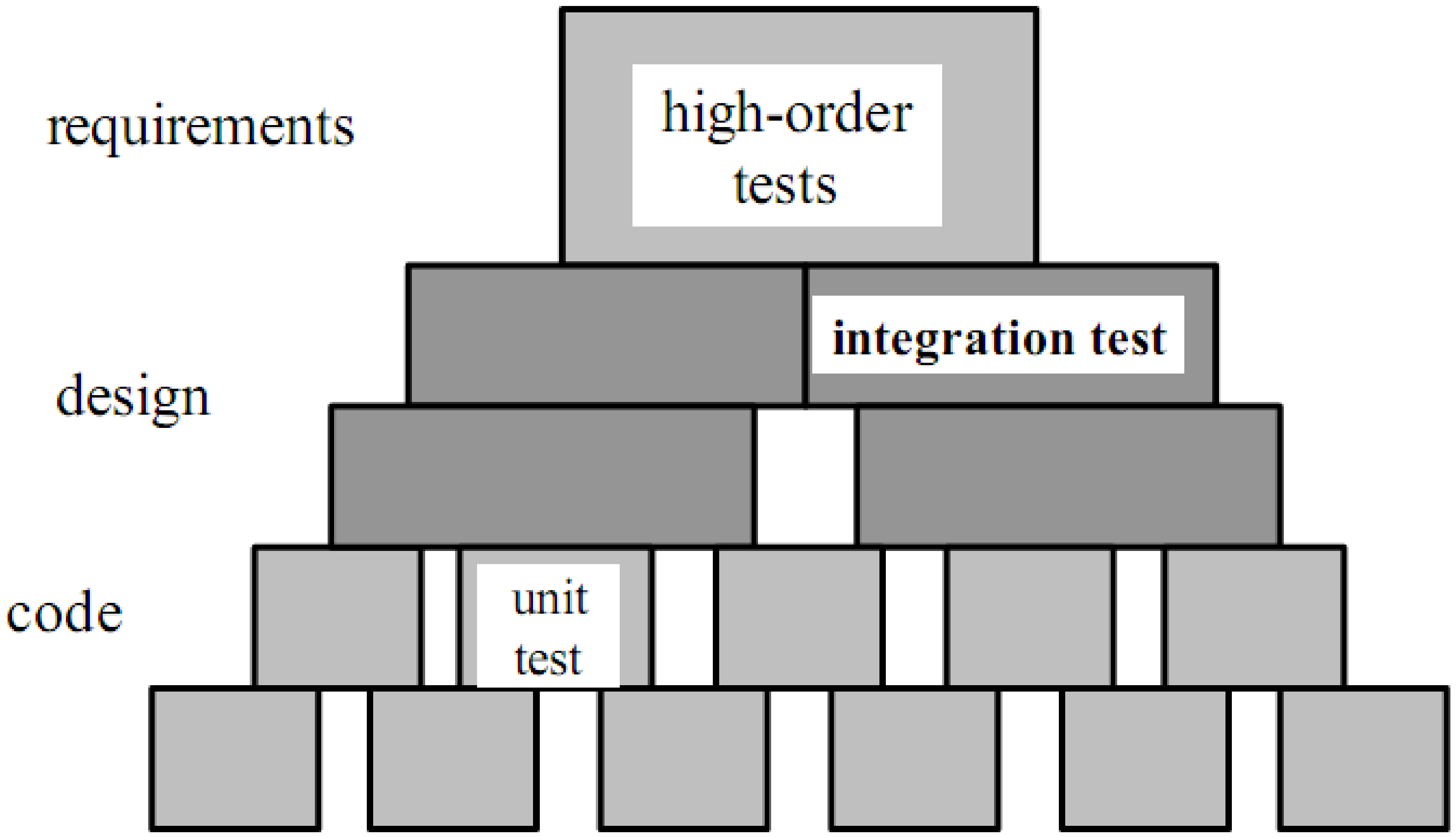
System test

Validation Test

Integration test

Unit Test

Testing Strategy



Correctness Testing

- **Needs some type of oracles**
- **Black-box testing/behavioral testing**
  - also: data-driven; input/output driven[1]; requirements-based[3]
  - Test data are derived solely from the program structure[9]
  - "Exhaustive input testing"[1]
  - But, what about omissions/extras in spec?
- **White-box testing/structural testing**
  - also: logic-driven[1]; design-based[3]
  - Application of test data derived from the specified functional requirements without regard to the final program structure[9]
  - "Exhaustive path testing"[1]
  - But, what about omissions/extras in code?
- **Other than bugs, we may find:**
  - Features
  - Specification problems
  - Design philosophy (e.g. core dumps v.s. error return code)

Correctness Testing Methods/Tools

- **Control-flow testing**
  - Trace control-flow using control-flow graph; coverage
- **Loop testing**
  - A heuristic technique; should be combined with other methods
  - Applied when there is a loop in graph
- **Data-flow testing**
  - Trace data-flow using data-flow graph; coverage
- **Transaction-flow testing**
  - Testing of on-line applications and batch-processing software
  - Has both control-flow and data-flow attributes

Flow-coverage testing

- **Domain testing**
  - Software dominated by numerical processing
- **Syntax testing**
  - Command-driven software and similar applications
- **Finite-state testing**
  - Using finite-state machine model
  - motivated from hardware logic design
  - Excellent for testing menu-driven applications

When to stop testing?

- **Trade-off between budget+time and quality**
  - Part of acceptance testing
- **Stopping rules:**
  - When reliability meets requirement
    - Statistical models
      - » E.g. reliability growth models
    - Data gathering --> modeling -->prediction
    - Not possible to calculate for ultra-dependable system
      - » Because failure data is hard to accumulate
  - When out of resources: test case, money and/or time

Testing is Controversial

- **Alternatives to testing**
  - "human testing[MYERS79]"
    - inspections, walkthroughs, reviews
  - Engineering methods
    - Clean-room v.s. testing
  - Formal Verification v.s. Testing

- **Flames**
  - Traditional coverage-based testing is flawed.
  - Testing can only prove the software is flawed.
  - Inspection/review more effective than testing?
  - "If we have good process, good quality, we don't need much testing"

Conclusions

- **Complete testing is infeasible**
  - Complexity problem
  - Equivalent to Turing halting problem
- **Software testing is immature**
  - Crucial to software quality
- **Testing is more than debugging**
  - For quality assurance, validation and reliability measurement
- **Rules of thumb**
  - Efficiency & effectiveness
  - Automation
- **When to stop: need good metrics**
  - Reliability
  - Time & budget

References of Research Paper

- http://www.informatik.uni-trier.de/~ley/db/conf/icse/icse2013.html
- http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6569023
- http://icst2012.soccerlab.polymtl.ca/Download/ICST2012_TOC.pdf
- http://www.jatit.org/volumes/research-papers/Vol18No2/5Vol18No2.pdf
- http://download.springer.com/static/pdf/489/art%253A10.1007%252Fs11219-012-9181-z.pdf?auth66=1411907731_5380cc5cddbe7910e24058ac23512622&ext=.pdf
- http://www.sersc.org/journals/IJSEIA/vol4_no1_2010/2.pdf
- http://www.ijcse.com/docs/INDJCSE11-02-05-096.pdf