Quality Assurance – Part II: Software Testing in a DSD

Lian Yu
The School of Software and Microelectronics
Peking University
No.24 Jinyuan RD, Beijing 102600, P. R. China

Topics

• Objectives of software testing
• Principles and testability
• Test case design
• Type of testing
• Testing strategy
• Validation testing and system testing
• Software Testing in a DSD (open questions)
Testing Objectives

- [Myers79]: Software Testing is the process of executing a program or system with the intent of finding errors.
  - A good test case is one that has a high probability of finding an as-yet undiscovered error
  - A successful test is one that uncovers an as-yet undiscovered error

- [Hetzel88]: It involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.

What is Testing and What is Not?

- Testing is a process for finding semantic or logical errors
- Testing is not aimed to find syntactic (symbolic) errors
- Testing can reveal the presence of errors, NOT their absence.
- Can you tell the differences between testing, debugging and compilation debugging?
Why Software Testing?

- No single verification technique is sufficient to find all categories of errors, especially in large, complex systems. For example, review and inspection are not adequate because they do not execute the program. Can you test a car without driving it?
- Why we need testing:
  - Instrumentation systems & appliances
    - coded in assembly/machine language ==> costly to repair
  - Process control systems
    - failure is politically unacceptable & economically undesirable.
  - Military systems
    - involve risk of human life.
    - $10 - 25 per statement, higher for “complete treatment.”
  - Safety - Critical Health Care Systems.

Software Testing in Practice

- Testing amounts to 40% -- 80% of total development costs
  - 40% for information systems
  - 80% for real time embedded systems
- On one occasion the average cost of a field detected bug was US$15,000 (1991).
- Testing receives the least attention and often not enough time and resources.
- The testers are often forced to abandon testing efforts because changes have been made.
- Testing is (usually) at the end of the development cycle and is often left with little time because other activities have been late.
Software Testing in Practice

• Most companies’ new hires are testers.
• Most testing work is manually testing the software system; help from tools is still limited.
• In many cases, testing is not performed by using systematic testing methods or techniques.
• Because no systematic methods or techniques are used, testing is not effective.
• Sometimes there are “conflicts of interest” between testers and developers.

Benefits of Software Testing

• Benefits:
  – Conscious of need for error-free software
  – Ensure the more flagrant kinds of errors will be detected (this is not really true; maybe the more obvious kinds of errors)
  – As a backup to design reviews, structured walk through, etc. (not a backup – studies show that testing and reviews in practice detect different kinds of errors)
  – A framework to apply new SQA techniques
• Much more effective than programmers’ casual testing:
  – Most Programmers: 20-50% c1 coverage
  – >= 85% c1 coverage would be adequate to discover 90% of the errors (U.S. Air Force requirement on JAVS, 1973-74)
C0 and C1 Coverage

C0 coverage:
\[
\frac{\text{# of statement tested}}{\text{# of statement in Program}}
\]

C1 coverage:
\[
\frac{\text{# segment tested}}{\text{# total segments in program}} \quad \text{(Define segment)}
\]

Segment:
e.g., statements, conditions, methods, paths, def-use, states, transitions,

The Psychology of Testing

• Bad feelings about testing:
  – a dirty business
  – dumb work/menial work
  – headache undertaking
  – under-funded & too much work
  – too little time & under pressure
  – no good tools
  – not rigorous, systematic
  – no generally accepted principles
  – requires a critic’s mentality
  – destructive work
Verification, Validation, Testing & Debugging

- Testing: attempts to uncover program errors.
- Debugging: follows successful testing and attempts to remove program errors.
- Verification: “are we building the product right?”
- Validation: “are we building the right product?”
- Testing is a pragmatic approach to Software Quality Assurance (SQA).
- Program correctness proof is a theoretical approach (to SQA).

Verification, Validation, Testing & Debugging

- Testing is aimed at proving that errors exist; it cannot assert that a program is error free.
- Program correctness proofs attempt to prove that a program is free from errors. Once proved, the program is correct forever. (assuming the environment doesn’t change)
- A combination of the above two is symbolic execution. This technique executes programs using symbolic values and the rules developed by program correctness proof.
- What about model checking?
Testing Principles

• All tests should be traceable to customer’s requirements
• Tests should be planned long before testing begins
• 80% of the bugs are likely to be traceable to 20% of program components
• Testing should begin “in the small” and progress toward to testing “in the large”
• Exhaustive testing is impossible
• To be more effective, testing should be conducted by an independent third party

Test Case Design

• White-box testing
  – knowing the internal workings of a product
    • focus on the program’s structure and internal logic
    • test cases are designed according to the structure and internal logic
  – well-known techniques
    • Statement testing: test cases are designed to execute every statement
    • Basic path testing: test cases are designed to exercise the control flow paths of a program
    • Condition testing: test cases are designed to exercise each outcome of a condition
    • Data flow testing: test cases are designed to test data elements’ define and use relationships
Test Case Design

- Black-box testing
  - Knowing the functional specification
    - focus on functional requirements
    - test cases are designed to test the functionality of the program

Test Case Design

- Well-known black-box testing techniques
  - boundary value analysis: test cases are derived according to the boundary values of variables
    - data invariance provides a useful tool here
  - causal-effect analysis: test cases are derived according to the stimuli and responses and input output relationships
  - equivalence partitioning: partition the input and output domains into disjoint areas and test cases are designed according to the partitions
  - What about operational profile testing? Regression testing?
Software Testing Strategies

- integrate software test case design methods into a well-planned sequence of steps
- a road map for testing
  - test planning, test case design,
  - test execution,
  - result evaluation
  - how much effort and time
  - resources required
- testing strategies must be customizable
- testing strategies must support planning and management

Test Categories

- Unit Testing: testing at the individual module level, functions, modules, classes, etc.
- Integration Testing: testing the interfaces among the modules or components.
- Validation Testing: test according to customer’s functional requirements.
- System Testing
  - Recovery Testing
  - Security Testing
  - Stress Testing
  - Performance Testing
  - Regression testing
Software Testing Budget

Size of testing budget:
• expressed in terms of x% of total SW development costs.
• in terms of system size: $x per statement.
• compute in terms of productivity of the test team in terms of time: planned & spent.

Software Testing in DSD

• What are the new issues related to quality in DSD? What kinds of quality assurance techniques can still be used in DSD?
• DSD provides both opportunity and difficulty in testing
  – Testing can be performed by a remote group that closely mimics the real users
  – Test cases must be specified precisely so that there are no ambiguities caused by natural language misunderstanding
  – With widely separated locations, testing can go on at one location after another location has completed coding – 24 hour development
• What existing testing techniques can be used? And what are the new techniques that should be introduced?
  – Process
  – Methodologies and Techniques
  – Tools
References