Test Generation Strategies

Dr. Mark C. Paulk

SE 4367 – Software Testing, Verification, Validation, and Quality Assurance

Jonsson School of Engineering and Computer Science

Topics: Software Testing

Part I: Preliminaries

1. Software Testing

- Humans, Errors, and Testing
- Software Quality
- Requirements, Behavior, and Correctness
- Correctness vs Reliability
- Testing and Debugging
- Test Metrics
- Software and Hardware Testing

- Testing and Verification
- Defect Management
- Test Generation Strategies
- Static Testing
- Model-Based Testing and Model Checking
- Types of Testing
- Saturation Effect
- Principles of Testing

Test Generation

Execute the program against the test cases to determine whether it conforms to the requirements… need a source document.

- **white-box, functional testing**

Informal techniques applied directly to the requirements

Formal model based test generation – need requirements modeled using a formal notation

Generating tests directly from code

- **black-box, structural testing**
- **selecting tests to minimize cost**

Static Code Analysis Tools

Control-flow information in a control flow graph (CFG)

- **annotate with data-flow information to make a data flow graph (DFG)**
  - **append list of variables defined and used**

Prioritize more complex modules for peer review

- **cyclomatic complex <10 good, >20 bad**

Model Checking

Extract a (finite-state) model from some source (e.g., requirements, code, …)

Code desired properties in a formal specification language

Input model and desired properties to model checker

- **property is satisfied**
- **property is not satisfied**
- **unable to determine**

Model-based testing → Generation of tests from a formal model of program behavior

Five Classifiers for Dynamic Testing

C1 – source of test generation

C2 – life cycle phase in which testing takes place

C3 – goal of a specific testing activity

C4 – characteristics of the artifact under test

C5 – test process

C1: Source of Test Generation

Black-box testing

White-box testing

Model-based or specification-based testing

Interface testing

C2: Life Cycle Phase

Phase	Technique
Coding	Unit testing
Integration	Integration testing
System integration	System testing
Maintenance	Regression testing
Postsystem, prerelease	Beta testing

C3: Goal-Directed Testing

Robustness testing

Stress testing

Performance testing

Load testing

Functional testing

Security testing
- **vulnerability testing**

GUI testing

Operational testing

Reliability testing

Acceptance testing

Compatibility testing

Configuration testing

…

C4: Artifact Under Test

Application component	**Component testing**
Batch processing	**Equivalence partitioning, FSM testing, …**
Client and server	**Client-server testing**
Compiler	**Compiler testing**
Design	**Design testing**
Database system	**Transaction-flow testing**
OO software	**OO-testing**
…	**…**

C5: Test Process Models
(Method, Methodology)

Waterfall

V-model

Spiral

Agile

Test driven development (TDD)

An Economic Activity

Testing is an economic activity

- **for a given effort testing, there is a return in terms of defects found**
- **we can never guarantee zero defects (even if we warranty the software)**
- **people make mistakes…**
- **simple errors can lead to large failures**

Test Assessment

Measuring the goodness of a test set T, a collection of test inputs, based on one or more test adequacy criteria.

- **adequacy score or coverage, usually between 0 and 1**
- **list of weaknesses in T, which when removed, will raise the score to 1**
- **the weaknesses depend on the criteria used for assessment**
- **improving testing → finding more defects**

The Coverage Principle

Measuring test adequacy and improving a test set against a sequence of well defined, increasingly strong, coverage domains leads to improved reliability of the system under test.

Coverage domains include items such as

- **requirements**
- **classes**
- **mutations**
- **exceptions**
- **data flows**
- **…**

A Hierarchy of Coverage Criteria

Low to high coverage (partial set)

- **requirements**
- **function/method**
- **statement**
- **decision**
- **conditions**
- **modified paths**
- **data flow**
- **mutation**

Saturation Region

Reliability increases as defects are removed…

True reliability saturates, i.e., stops increasing, after a certain amount of test effort has been spent for a given criterion

No new defects are likely to be found and fixed in the saturation region

- **there may still be defects in the software that our tests cannot find**
- **every test generation method has its limitations**

Reliability vs Confidence

Finding and fixing previously undiscovered defects leads to greater confidence in the program

There is a gap between true reliability and our confidence in the program

Test generation methods are practically limited in what they can find

- **use multiple test generation methods**
- **any method of constructing functional tests is inadequate in comparison to code-based coverage criteria**

Myers' Principles of Testing (2004)

A necessary part of a test case is a definition of the expected output or result.

A programmer should avoid attempting to test his or her own program.

A programming organization should not test its own programs.

Thoroughly inspect the results of each test.

Test cases must be written for input conditions that are invalid and unexpected, as well as for those that are valid and expected.

Examining a program to see if it does not do what it is supposed to do is only half the battle; the half is seeing whether the program does what it is not supposed to do.

Avoid throwaway test cases unless the program is truly a throwaway program.

Do not plan a testing effort under the tacit assumption that no errors will be found.

The probability of the existence of more errors in a section of a program is proportional to the number of errors already found in that section.

Testing is an extremely creative and intellectually challenging task.

ISTQB Seven Principles of Testing

http://istqbexamcertification.com/what-are-the-principles-of-testing/

Testing shows presence of defects

- **Even after testing the application or product thoroughly we cannot say that the product is 100% defect free.**

Exhaustive testing is impossible

Early testing

- **In the software development life cycle testing activities should start as early as possible and should be focused on defined objectives.**

Defect clustering

- **A small number of modules contains most of the defects discovered during pre-release testing or shows the most operational failures.**

Pesticide paradox (aka saturation region)

- If the same kinds of tests are repeated again and again, eventually the same set of test cases will no longer be able to find any new bugs.

Testing is context dependent

- Different kinds of programs are tested differently, e.g., life-critical.
- Programmers will unit test code based on their knowledge of the structure of the code (white box testing).
- Product owners and other business stakeholders will test based on their understanding of how the overall product will behave (black box testing).

Absence-of-errors fallacy

- If the system built is unusable and does not fulfill the user's needs and expectations then finding and fixing defects does not help.
- Unless you build the right thing, building it right isn't even a useful conversation.

Summary – Things to Remember

Early testing removes defects cheaper

Defect clustering aka Pareto principle – most defects are in a small part of the code

Saturation region aka pesticide paradox – using the same testing criterion reaches a point of diminishing returns

Absence of errors fallacy – if you aren't building the right thing, building it right is a waste

Questions and Answers