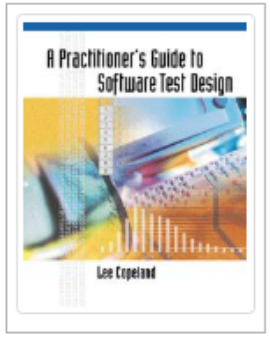
Textbook Chapters on Software Testing



Practitioner's Guide to Software Test Design

by Copeland, Lee

AVAILABILITY

Your institution has unlimited access to this book.

Available for Online Reading

58 pages remaining for copy (out of 58)

116 pages remaining for print or chapter download (out of 116)

Available for Full Download

Check out this book for up to 21 days.

Read Online

Full Download

Chapter Download

Saved to Bookshelf

Share Link to Book

Cite Book

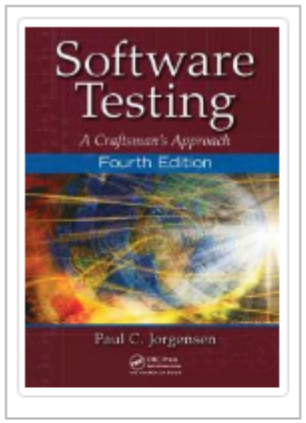
Table of Contents | Description

TABLE OF CONTENTS

Lesson 4.2.1 – Introduction to Testing (Chapter 1, 12, 13)
Lesson 4.2.4 – White box testing (Chapter 10)

http://ebookcentral.proquest.com.virtual.anu.edu.au/lib/anu/detail.action?docID=227688

Textbook Chapters on Software Testing



Read Online

Full Download

Chapter Download

Add to Bookshelf

Share Link to Book

Cite Book

Software Testing

by Jorgensen, Paul C.

AVAILABILITY

Your institution has access to multiple copies of this book.

Available for Online Reading

71 pages remaining for copy (out of 71)

71 pages remaining for print or chapter download (out of 71)

Available for Full Download

Check out this book for up to 7 days.

Table of Contents | Description

TABLE OF CONTENTS

Lesson 4.2.3 - Black box testing (Chapter 5, 6, 7)

http://ebookcentral.proquest.com/lib/anu/detail.action?docID=1463516&token=304736b8-97e5-4363-8b61-03e024cc8468

Module 4 – Topic 4.2 – Lesson 4.2.1

Introduction to Testing

Lesson Outline

- What is testing?
  - What is versus what ought to be
- The challenges of software testing
  - You cannot test everything!
- Test case design
  - Input, output, order of execution
- Test planning
  - Scripted testing versus exploratory testing
- Types and levels of testing
  - Different types and levels of testing complement one another

What is Testing?

The process of comparing "what is" with "what ought to be"

When I launch my app after several hours of development

When the code that I have not tested works perfectly in production

IEEE Standard Glossary of Software Engineering Terminology

The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component.

This definition focuses on test execution.

The "specified conditions" are embodied in test cases, the focus of this course.

An Expanded Definition

Testing is a concurrent lifecycle process of engineering, using and maintaining testware in order to measure and improve the quality of the software being tested.

A definition by Rick Craig and Stefan Jaskiel in their book "Systematic Software Testing"

This view includes the planning, analysis, and design that leads to the creation of test cases.

The Challenges in Software Testing

Typical Testing Challenges

- Management that either does not understand testing or (apparently) does not care about quality
- Not enough time to test properly or thoroughly
- Too many combinations of inputs to test
- Difficulty in determining the expected results of each test
- No training in testing processes
- No tool support

Our focus: make you more efficient and effective in your testing by helping you choose and construct test

The Impossibility of Test Everything

Specification: Take an integer input value, add 1 to input, divide it by 30000 and return the value

You do not see implementation

short int – 2^{16} values
int – 2^{32} values
long int – 2^{64} values

Assume input value is a short int, which value(s) do you choose to test the implementation?

Input	Expected Output	Actual Output
0	0	
1	0	
-42	0	
32000	1	

Black box testing

- Boundary Value Testing
- Equivalence Class Testing
- Decision table testing

The Impossibility of Test Everything

```
func() {
    while (a) {
        if(b) {
            …
        } else {
            if(c) {
                …
            }
        }
        if(d) {
            …
        }
    }
    …
}
```

If the loop has 10 repetitions, how many distinct program execution paths are there?

- Can you execute all execution paths at lease once?

Executing all execution paths is in general infeasible!

- Every decision point doubles the number of paths (i.e., $2^{|decision|}$)
- Every loop powers the paths by the number of iterations

White box testing

- Control flow graph
- Control flow testing

Five Levels of Testing Maturity

- Level 0 – "Testing == Debugging"
- Level 1 – "To show that software works"
- Level 2 – "To show the software doesn't work"
- Level 3 – "Not to prove anything, but to reduce the perceived risk of not working to an acceptable value"
- Level 4 – "Testing is not an act. It is a mental discipline that results in low-risk software without much testing effort"

Test Case Design

To be most effective and efficient, test cases must be designed

What "Design" Means

- To formulate a strategy for
- To plan out in systematic, usually documented form
- To create or contrive for a particular purpose
- To create or execute in an artistic or highly skilled manner

Software engineers often treat testing as an afterthought, developing test cases that 'feel right' but have little assurance of being complete.

We must design tests that have the highest likelihood of finding the most errors with a minimum amount of time and effort.

Test Case - Input

- Input can come from
  - data entered at a keyboard
  - data from interfacing systems
  - data from interfacing devices
  - data read from files or databases
  - the state the system is in when the data arrives
  - the environment within which the system executes

- What inputs or input combinations to test?
  - Black-box and white-box testing techniques help you decide

Test Case – Output

- Output can be
  - data displayed on a computer screen
  - data can be sent to interfacing systems and to external devices
  - data can be written to files or databases
  - the state or the environment may be modified by the system's execution

Run the program with the input and see if actual output (what-is) matches expected output (what ought to be)

Test Case – Expected Output (Oracle)

- An oracle is any program, process, or data that provides the test designer with the expected result of a test
  - Kiddie oracles – Just run the program and see comes out
  - Regression test suites – Compare the output to the results of the same tests run against a previous version of the program
  - Validated data – Compare the results against a standard such as a table, formula, or other accepted definition of valid output
  - Purchased test suites – Run the program against a standardized test suite that has been previously created and validated

Test Case – Order of Execution

- Cascading test cases, e.g.,
  1. Create an order
  2. Read an order
  3. Update the order
  4. Delete the order
  5. Read the deleted order

👍 Smaller and simpler test cases

👎 One fails, the subsequent test may be invalid

- Independent test cases, e.g.,
  - Each test case is entirely self contained

👍 Any number of tests can be executed in any order

👎 Larger and more complex test cases

Test Planning

Describes systematically the approach to testing software

Test Strategy & Test Plan

- Test strategy – a high-level document, mostly static

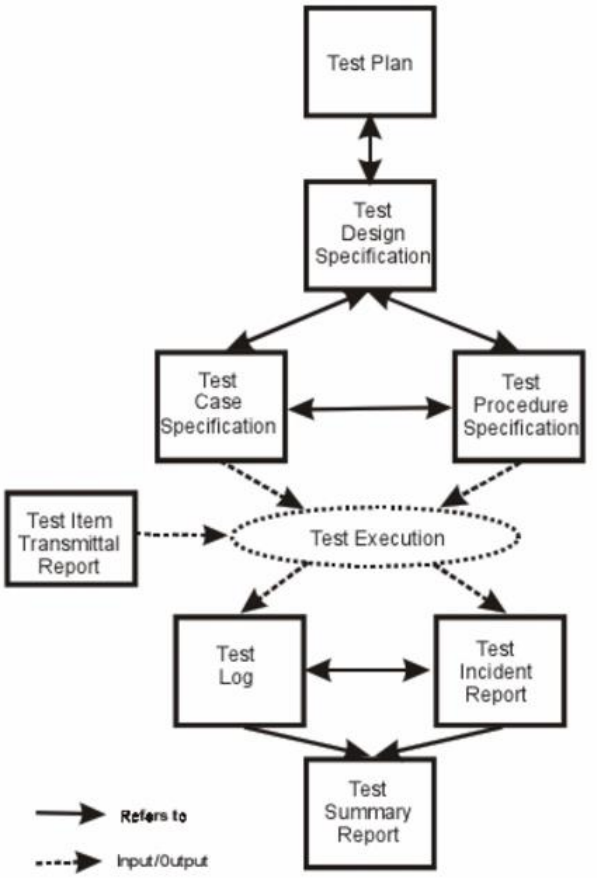
  Scope & objectives
  Roles & responsibilities
  Testing tools and automation
  Testing metrics
  Defect tracking mechanisms
  Testing standards
  Risk & change management

- Test plan – can be derived from the strategy, more fluid
  - describes what, how, when to test and who will test

Scripted Testing

- IEEE Standard for Software Test Documentation
  - Test plan
  - Test design specification
  - Test case specification
  - Test procedure specification
  - Test log
  - Test incident report
  - Test summary report

Plan your work, work you plan



Have a look at "Sample Master Test Plan"

Exploratory Testing

- Exploratory testing is "simultaneous learning, test design, and test execution"
  - It proceeds according to a conscious plan, but not a rigorous plan. It is not scripted in detail.
  - To the extent that the next test we do is influenced by the result of the last test we did

There's so much we don't know about the product and we know our testing can never be fully complete.

We cannot play the game of "Twenty Questions" by writing out all the questions in advance.

Agile Test Planning

- Have a strategy
- Define testing scope
- Be flexible & prepared for change
- Keep communication open
- Don't plan too much!

The use of scripted testing does not preclude the use of exploratory testing.

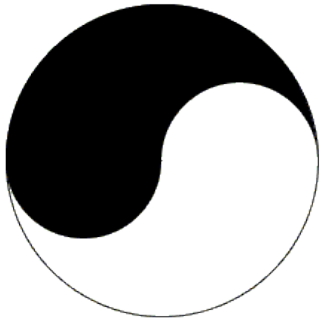
The use of exploratory testing does not preclude the use of scripted testing.

Types and Levels of Testing

Different types and levels of testing complement one another

Types of Testing

- Black box testing
  - What to analyse
    - Based solely on the requirements and specifications
  - Types of black box testing
    - Boundary value testing
    - Equivalence class testing
    - Decision table testing
- White box testing
  - What to analyse
    - based on the internal paths, structure, and implementation of the software under test
  - Types of white box testing
    - Control flow testing (also known as path testing)



Levels of Testing

- Unit testing (our focus in this course)
  - Test an unit individually
- Integration testing
  - Assemble units together into subsystems and finally into systems
- System testing
  - Focuses on defects that arise at this highest level of integration
  - Includes many types of testing: functionality, usability, security, internationalization and localization, reliability and availability, capacity, performance, backup and recovery, portability, and many more
- Acceptance testing
  - When completed successfully, will result in the customer accepting the software and giving us their money