

Software Testing

Software Testing in a Quality Context

This two-day course is essential for those who want to understand how software testing can be used as a means for improving software quality. The course commences with a discussion of some key quality concepts and how they relate to software quality.

A number of alternative techniques for improving software quality are briefly discussed, together with how these techniques complement the strengths and weaknesses of software testing.

The course then moves on to the core topics of black-box and glass-box test case design. Participants will learn a number of simple but effective test case design techniques that can be applied to all types of software.

The theory presented during the course is supported by a number of practical exercises and demonstrations of software tools that support the software testing process.

Course Objectives

- A clear understanding of the role of testing and how this relates to the goal of improving software quality.
- Understanding of the limitations of testing and alternative quality improvement techniques.
- Ability to apply a number of black-box and glass-box test case design techniques.
- Awareness of testing tools with emphasis on open source tools.

Who Should Attend

- Testers, Programmers, Business Analysts, Systems Analysts, Architects and Designers
- Methodologists, Quality Assurance and Process Improvement Staff

Course Duration

2 days

Contact

Phil Robinson
Email: lonsdale@iinet.net.au
Phone: 0411 261 505

Course Agenda

Introduction

Quality Concepts

- Some popular assumptions about quality
- Different views of quality
- Quality and processes

Software Quality and Risk

- What testing can and can't achieve
- Introduction to risk management
- Identifying risks
- Analysing risks

Techniques For Improving Quality

- Reducing software quality risk
- Early requirements validation
- Inspections
- Configuration management
- Testing
 - Level of testing
 - Types of test
 - Who tests?
 - Quality and independence
 - Quality planning framework

Designing Test Cases

- Test case coverage
- Test case dependencies
- Test case specification
- Test case design techniques
 - Functional test cases (black-box)
 - Structural test cases (glass-box)
 - Error guessing
- Demonstration of management software

Functional Testing Techniques

- State transition testing
- Equivalence partitioning
- Boundary value analysis

Structural Testing Techniques

- Cyclomatic complexity and control flow graphs
- Complexity and modular programming
- Complexity and defect prevention



Lonsdale Systems

www.iinet.net.au/~lonsdale/

Unit Testing

- Test coverage matrix
- Source code instrumentation
- Module testing strategies
- Introduction to “JUnit”
- Demonstration of “JUnit”

Integration Testing

- Integration strategies
- Design reduction and control flow graphs
- Design complexity
- Integration test cases

System and Acceptance Testing

- Goals of system testing
- System test and effect repair process
- Acceptance testing
- End-to-end and scenario testing
- Beta-testing
- Demonstration of “Abbot” automated testing tool

www.opensourcetesting.org

- Overview of open source testing tools
- Browser-based test case management and bug tracking tools
- Web test tools

Review and Conclusion

