

Introduction to Software Testing & Quality Assurance

15 Key Questions that will be Answered by Taking this Course

1. How does testing fit into the overall software development life cycle?
2. What is the role of a test group?
3. What is the role of the QA organization?
4. Should we try to test everything?
5. How do we manage the risks that threaten our project's success?
6. How can I test when the tests aren't written down and I can't get the product documentation that I need?
7. What techniques can I use to develop test cases?
8. How should I do system-level testing?
9. What kind of test documentation should I produce?
10. How can I report bugs in a way that will be useful to the people who need to know about them?
11. How can I determine the quality of the software without running it?
12. What kinds of metrics are useful?
13. How can approach process improvement in a systematic way?
14. How can I take advantage of test automation tools?
15. Where can I learn more?

Comprehensive Course Outline

This course provides a highly practical bottom-up introduction to software testing and quality assurance. Every organization delegates testing and high-level quality assurance activities in a different way. This course provides a broad view of both quality assurance and testing so that participants will have a broad awareness of many of the activities that contribute to managing the quality of a software product.

1. The Software Life Cycle

An introduction to the fundamental elements of the software life cycle, and several life cycle models that are commonly in use. Also, how testing is integrated into the life cycle.

- a. Life cycle models
- b. Types of testing and where they fit in
 - i. Unit Testing
 - ii. Regression Testing
 - iii. Integration Testing
 - iv. System Testing

2. The Role of Testing and QA

An overview of the typical responsibilities of test groups and quality assurance groups, along with common variations. The difference between testing and quality assurance, and the ways that these terms are often abused. The anatomy of a bug.

- *Exercise: Write the objectives for your test team. Compare notes.*

- a. Typical Test Group Objectives
- b. Typical Quality Assurance Objectives
- c. Variations
- d. What is a bug?

3. You Can't Test Everything

Why you cannot completely test any non-trivial application. A demonstration of the magnitude of the testing challenge due to the number of possible inputs, combinations of features and configurations, and paths through the code.

- *Exercise: Define a small part of an application that is familiar to the class and calculate the number of tests required to cover all feature combinations.*

- a. Inputs
- b. Combinations
 - i. Feature Interactions
 - ii. Configurations
- c. Code Paths

4. Risk Management

An introduction to the risk management process – identifying risks, analyzing them, mitigating them, contingency planning, and monitoring. How to use a risk-based approach to prioritize

testing tasks and plan the appropriate level of resources to apply to testing, given that you can't test everything.

- *Exercise: Participate in a facilitated risk brainstorming session.*
- *Exercise: Split into two groups. For two different sample projects with the same software but different risks, plan the testing effort. Compare the results.*

a. Risk Management Process

- i. Identify
- ii. Analyze
- iii. Plan
- iv. Mitigate
- v. Track

b. Risk-Based Testing

- i. Using a Risk Analysis to Plan Testing
- ii. Quality Criteria
- iii. Risk Catalogs
- iv. Risk-Based Release Decisions
- v. When to Use Alternate Methods to Mitigate Risk

5. Exploratory Testing

An introduction to the concept of “Testing in the Dark.” You find an unlabeled CD on your desk with a note that says “Please test.” How to use exploratory testing to find bugs even in the most adverse conditions. How to make exploratory testing a disciplined process, distinct from a chaotic ad-hoc testing process. Also, how to take advantage of exploratory testing even in the most organized development processes.

- *Exercise: Test a virtual software system you've never seen before using exploratory testing techniques.*

a. Testing in the Dark

b. Anatomy of Exploratory Testing

c. Differences Between Exploratory and Ad-Hoc Testing

d. Knowing When Exploratory Testing is the Best Approach

f. Reporting

e. Supplementing a Documented Test Suite With Exploratory Testing

6. Test Design Techniques

A tour of several low-level test design techniques. What situations would lead testers in an independent test group to use these techniques.

- *Exercise: Discuss the appropriate combination of techniques to use for a few different testing projects.*
- *Exercise: Use the all-pairs technique to reduce the number of tests required to cover feature interactions for a particular program.*

a. Functional Analysis

b. Requirements Analysis

c. Partitioning

d. Domain Analysis

- e. User Scenarios
- f. All-pairs
- g. Using Combinations of These Techniques

7. System Testing

The high-level approaches that testers need to apply when conducting system testing, plus some examples of how they might be applied.

- a. Load Testing
- b. Performance Testing
- c. Stress & Hot Spot Testing
- d. Spike & Bounce Testing
- e. Reliability Testing
- f. Configuration Testing
- g. Acceptance Testing
- h. Sample System Test Strategies

8. Test Documentation

How to develop an arsenal of repeatable tests so you don't have to redo the test design process every time you re-test the software. How to carefully calibrate the level of detail in the test documentation so that there is a good return on the investment for the time it takes and the later maintenance that will be required.

- a. Rude Awakening – IEEE 829
- b. What Will Your Test Documents Be Used For?
- c. The Quality Plan
- d. Test Project Planning
- e. Test Case Documentation
 - i. Manual Tests
 - ii. Automated Tests

9. Bug Isolation and Reporting

When testers are successful, they will come across bug symptoms. This section explains the fine art of bug isolation – turning a symptom into a well-defined bug report. Also, the organizational factors that need to be considered in order to maximize the chance of getting the bug fixed.

- *Exercise: Write a bug report based on the observed symptoms.*

- a. "A problem well-stated is half-solved."
- b. The Goal of Bug Reporting
- c. The Importance of Reproducibility
- d. Isolation and Simplification
 - i. Bottom-up
 - ii. Top-down
 - iii. The "binary defect search" technique
- e. Generalization
- f. Bug Reporting
 - i. Important Elements
 - ii. Working with Developers
 - iii. Follow-up

10. Static Testing

How to find bugs early without executing the software. An overview of a handful of review and inspection techniques, and how testers should participate. Also, how to utilize static analysis tools and services.

- a. Reviews and Inspections
 - i. Desk Check
 - ii. Walkthrough
 - iii. Inspection
 - iv. Combining Review Techniques
- b. Static Analysis Tools
 - i. Complexity Analysis
 - ii. Defect Detection
 - iii. Coding Standards Enforcement

11. Metrics

A sampling of metrics that the tester can use to demonstrate progress.

- a. Code Coverage Analysis
- b. Functional and Requirements Coverage
- c. Bug Metrics
 - i. Find Rate vs. Close Rate
 - ii. Severity
 - iii. Bug Reviews

12. Process Improvement

The importance of making continual improvements, and the difficulties of big-bang re-engineering. How to approach process improvement.

- a. Avoiding “Genius Mode” – Don’t Jump to Solutions
- b. Identify the Problems
- c. Prioritize
- d. Identify Solutions
- e. Choose a Solution
- f. Manage the Improvement

13. Overview of Automated Testing

An introduction to the benefits and pitfalls of automated test execution. Situations where test automation is most useful. How to avoid creating unmaintainable tests. How data-driven techniques can allow non-programmers to create automated tests.

- *Exercise: For a variety of sample testing projects, decide where it is appropriate to apply automation.*

- a. Why Automate Testing?
 - i. What Can Automated Testing Achieve?
 - ii. The Limits of Automated Testing
- b. How Test Automation Tasks Are Typically Delegated

- c. Critical Testware Maintenance Issues
- d. Data-Driven Testing

14. Resources

A recap of tools that can assist testers, plus additional tools and where to find them. A list of recommended books, periodicals, conferences, organizations, and web sites that testers can go to for additional information.

- a. Tools Recap
- b. Books
- c. Periodicals
- d. Conferences
- e. Organizations
- f. Web Sites

Course Materials

Each student will receive course notes and information on how to find other resources.

About the Instructor

Danny R. Faught, as proprietor of Tejas Software Consulting, is an independent software quality consultant and trainer. Danny graduated with honors from the University of North Texas with a Bachelor's Degree in Computer Science.

Danny frequently speaks and writes about software quality. He created the comp.software.testing FAQ, co-founded the swtest-discuss mailing list, and maintains a well-known trio of testing FAQs at testingfaqs.org. Danny is a member of the American Society for Quality, the Project Management Institute, and the Forth Worth Chamber of Commerce. Danny also serves on the Practicality Review Board for STQE magazine.

Course Dates

This course is available as a public-enrollment course from time to time. Check <http://tejasconsulting.com/courses> for details on upcoming course dates. It is also available as a private course at your location. Contact Danny Faught at +1 817 294 3998 or faught@tejasconsulting.com for details.