skillsoft global knowledge™

FUNDAMENTALS OF SOFTWARE TESTING

Course Code: 2515

Learn the fundamental techniques and approaches to software testing and better understand what to test, how to test it, and in what contexts certain practices make sense.

Testing is a critical role in software development that requires special skills and knowledge that are not commonly taught to software developers, business analysts and project managers. This often results in insufficient time and resources being allocated for this important function, and quality suffers—as do the users of the software. Equally important is the need to measure quality quickly and efficiently because limitations in resources and schedules are realities that aren't going away. Enhancing the professionalism of everyone involved in software testing will make them effective contributors to teams that deliver high, proven-quality software.

Fundamentals of Software Testing provides an eye-opening view into this challenging task based on several sources of industry best practice. It provides a complete picture of the testing process, how it fits into the development life cycle, how to properly scope and prioritize testing activities, and what techniques to use for optimal results. Students come away with many ideas that they can apply in their own projects to improve the effectiveness and efficiency of testing efforts.

What You'll Learn

- Develop a model of the application
- Use their model to determine test coverage
- Identify test oracles for the application
- Create test cases based on the oracles
- Run their tests against the live application
- A deep-dive into the Universal Testing Method
- Look at testing phases, testing approaches, non-functional testing, and testing for different platforms
- An introduction to automation testing and behavior-driven development

Who Needs to Attend

- Testers of all types and levels
- Other disciplines who perform their own testing or are involved in testing

- Quality Assurance Professionals
- Test Management
- QA Managers
- QA Directors
- Software Engineers
- Business Analysts
- Project Managers
- IT Specialists (Security, Capacity Management, Networking...)
- Business Stakeholders
- Outsourcer Staff (Buyers and Suppliers)
- Application Development Managers

FUNDAMENTALS OF SOFTWARE TESTING

Course Code: 2515

CLASSROOM LIVE	$1,760 CAD	3 Day

Classroom Live Outline

Part 1: Introduction and Overview
Establishes a foundation for the course, provides a workable definition of software quality and shows how testing fits into the overall quality process.

Part 2: What to Test and How to Test it — The Universal Testing Method
Testers follow the same basic process that scientists use; we follow the principles of experimentation and measurement. In this course, we map your testing method back to those principles and show how, at each step in your testing, you're making complex decisions about what to test and how to test it. Utilizing a combination of skills, tactics, practices, and tools - this section helps build a base that testers in any context (of any skill level) can apply to solve testing problems.

1. **Model the Testing Space**. Compose, describe and work with visual models of the software to identify relevant dimensions, variables, and dynamics so you can develop an effective test strategy.
2. **Determine Test Coverage**. Understand a variety of common measures of test coverage and choose the most appropriate ones for each project; determine the scope of testing; establish a structure to track test coverage
3. **Determine Test Oracles.** Identify the sources of truth to determine whether the application has passed or failed tests; review common formal and heuristic oracles
4. **Determine Test Procedures**. Define what test procedures and test cases are; identify common test procedure types; learn how to document test procedures in a practical, efficient manner
5. **Configure the Test System**. See how to ensure you have everything needed to support testing; discuss common challenges to test configuration; consider test lab requirements and realities
6. **Operate the Test System**. Learn how to manage tester contact with the

application under test (AUT); discuss different methods of interaction with the system to address different testing objectives; identify common artifacts and practices related to test operation

7. **Observe the Test System.** Learn what empirical data to capture about the application under test and how to preserve testing interactions for review and reproducibility; consider common tools used to assist with test observation; identify common problems and human tendencies related to observation
8. **Evaluate Test Results**. Discuss possibilities and probabilities related to test results (not every test failure is a bug!); identify typical test result evaluation tasks; consider performance test results interpretation; learn key factors related to defect communications
9. **Report Test Results.** Learn how to make credible, professional reports of testing results and testing progress that address the varied needs of your target audiences; identify guiding principles for report design; review best practices and examples for defect reporting, progress status reporting, and quality statistics reporting

Part 3: Test Case Strategies

The heart of good testing is coming up with good test cases. In this section, we will define what makes test cases "good", and discuss these strategies for identifying test cases in specific contexts:

1. White Box strategies
2. Black Box strategies
3. Input and data-based strategies
4. User interface oriented strategies
5. Business Process flow strategies
6. Strategies based on your personal and organizational experiences

Part 4: Common Phases of Testing

Different testing activities take place as the software progresses through its life cycle. (Agile testers perform these same testing activities, even though they are not project phases.) This section explains the common phases of software testing, including the purpose of each, who normally performs it, and the typical types of tests that are done.

Test phases or types discussed:

1. Unit and Software
2. Integration
3. System and System Integration
4. Product Readiness
5. Regression
6. User Acceptance

Part 5: Approaches to Testing

Different approaches to testing are used to address different testing objectives and different project conditions. Some approaches are more formal, lengthy, traceable, and reproducible. Others are more free-form, quicker, less traceable, and less reproducible. The range of such approaches forms a continuum from which testers select the optimal combination for a given project. The best selection of approaches addresses the needs for both positive and negative testing.

1. The Testing Approach Continuum
2. Scripted Testing
3. Freestyle Testing
4. Middle-Ground (Charters, Checklists, Scenarios)

Part 6: Non-Functional Testing

Without question, functional testing is a must-have for software quality. However, there's more to the picture than that. This section describes several key types of non-functional testing and identifies, what their scope is, and what techniques or best practices apply.

1. Performance
2. Usability
3. Accessibility
4. Security
5. Portability
6. Localization

Part 7: Platform Specialization

Software is not just on the desktop—it runs on numerous platforms, and it all needs to be tested. This section takes multiple platforms into consideration and identifies each platform's unique challenges, and the best testing approaches for each given platform.

1. Web-Based
2. Mobile
3. SOA (Service-Oriented Architecture)
4. Telephony and Voice
5. DW/BI (Data Warehouse and Business Intelligence)
6. COTS/MOTS - Package Implementations (COTS)

Part 8: Test Automation — Bonus Section

There have been many organizations that have set out to implement automation testing in their projects, and many of them have failed. This section identifies the different types of tools and practices that fall into the "automation" category and helps set realistic expectations and goals for automated testing. Learn how to optimize your automation testing investment and plan properly for long-term success. This is a bonus section that is discussed as time permits.

1. Automated Test Tools
2. System Monitor Tools
3. File/Database Comparison Tools
4. Static Analysis Tools

Part 9: Behavior Driven Development (BDD) & Test Driven Development (TDD) — Bonus Section

BDD and TDD are related approaches to software development that came out of the Agile movement and have proven to have a significant positive impact on software quality. This section provides an introduction to the concepts so testers can be prepared to adopt them together with developers and other project members using iterative development methods. This is a bonus section that is discussed as time permits.

1. Test-Driven Development activities
2. Behavior-Driven Development activities
3. Tools for Different Languages

Part 10: Managing Testing Projects

Whether you lead a team of testers or work as the lone tester on a project, effectively managing the testing work is key to your ability to successfully test the software on time with the resources at hand. In this section, we will address the basics of managing your work in a way that is relevant to individual contributors and lead leads alike.

1. Planning for Testing (Universal Testing Method Steps 1-4)
2. Requirements Traceability
3. Test Resource Needs
4. Testing Risks and Issues
5. Testing Entry and Exit Criteria
6. Measuring Testing Progress

FUNDAMENTALS OF SOFTWARE TESTING

Course Code: 2515

VIRTUAL CLASSROOM LIVE	$1,760 CAD	3 Day

Virtual Classroom Live Outline

Part 1: Introduction and Overview
Establishes a foundation for the course, provides a workable definition of software quality and shows how testing fits into the overall quality process.

Part 2: What to Test and How to Test it — The Universal Testing Method
Testers follow the same basic process that scientists use; we follow the principles of experimentation and measurement. In this course, we map your testing method back to those principles and show how, at each step in your testing, you're making complex decisions about what to test and how to test it. Utilizing a combination of skills, tactics, practices, and tools - this section helps build a base that testers in any context (of any skill level) can apply to solve testing problems.

1. **Model the Testing Space**. Compose, describe and work with visual models of the software to identify relevant dimensions, variables, and dynamics so you can develop an effective test strategy.
2. **Determine Test Coverage**. Understand a variety of common measures of test coverage and choose the most appropriate ones for each project; determine the scope of testing; establish a structure to track test coverage
3. **Determine Test Oracles.** Identify the sources of truth to determine whether the application has passed or failed tests; review common formal and heuristic oracles
4. **Determine Test Procedures**. Define what test procedures and test cases are; identify common test procedure types; learn how to document test procedures in a practical, efficient manner
5. **Configure the Test System**. See how to ensure you have everything needed to support testing; discuss common challenges to test configuration; consider test lab requirements and realities
6. **Operate the Test System**. Learn how to manage tester contact with the

application under test (AUT); discuss different methods of interaction with the system to address different testing objectives; identify common artifacts and practices related to test operation

7. **Observe the Test System.** Learn what empirical data to capture about the application under test and how to preserve testing interactions for review and reproducibility; consider common tools used to assist with test observation; identify common problems and human tendencies related to observation
8. **Evaluate Test Results**. Discuss possibilities and probabilities related to test results (not every test failure is a bug!); identify typical test result evaluation tasks; consider performance test results interpretation; learn key factors related to defect communications
9. **Report Test Results.** Learn how to make credible, professional reports of testing results and testing progress that address the varied needs of your target audiences; identify guiding principles for report design; review best practices and examples for defect reporting, progress status reporting, and quality statistics reporting

Part 3: Test Case Strategies

The heart of good testing is coming up with good test cases. In this section, we will define what makes test cases "good", and discuss these strategies for identifying test cases in specific contexts:

1. White Box strategies
2. Black Box strategies
3. Input and data-based strategies
4. User interface oriented strategies
5. Business Process flow strategies
6. Strategies based on your personal and organizational experiences

Part 4: Common Phases of Testing

Different testing activities take place as the software progresses through its life cycle. (Agile testers perform these same testing activities, even though they are not project phases.) This section explains the common phases of software testing, including the purpose of each, who normally performs it, and the typical types of tests that are done.

Test phases or types discussed:

1. Unit and Software
2. Integration
3. System and System Integration
4. Product Readiness
5. Regression
6. User Acceptance

Part 5: Approaches to Testing

Different approaches to testing are used to address different testing objectives and different project conditions. Some approaches are more formal, lengthy, traceable, and reproducible. Others are more free-form, quicker, less traceable, and less reproducible. The range of such approaches forms a continuum from which testers select the optimal combination for a given project. The best selection of approaches addresses the needs for both positive and negative testing.

1. The Testing Approach Continuum
2. Scripted Testing
3. Freestyle Testing
4. Middle-Ground (Charters, Checklists, Scenarios)

Part 6: Non-Functional Testing

Without question, functional testing is a must-have for software quality. However, there's more to the picture than that. This section describes several key types of non-functional testing and identifies, what their scope is, and what techniques or best practices apply.

1. Performance
2. Usability
3. Accessibility
4. Security
5. Portability
6. Localization

Part 7: Platform Specialization

Software is not just on the desktop—it runs on numerous platforms, and it all needs to be tested. This section takes multiple platforms into consideration and identifies each platform's unique challenges, and the best testing approaches for each given platform.

1. Web-Based
2. Mobile
3. SOA (Service-Oriented Architecture)
4. Telephony and Voice
5. DW/BI (Data Warehouse and Business Intelligence)
6. COTS/MOTS - Package Implementations (COTS)

Part 8: Test Automation — Bonus Section

There have been many organizations that have set out to implement automation testing in their projects, and many of them have failed. This section identifies the different types of tools and practices that fall into the "automation" category and helps set realistic expectations and goals for automated testing. Learn how to optimize your automation testing investment and plan properly for long-term success. This is a bonus section that is discussed as time permits.

1. Automated Test Tools
2. System Monitor Tools
3. File/Database Comparison Tools
4. Static Analysis Tools

Part 9: Behavior Driven Development (BDD) & Test Driven Development (TDD) — Bonus Section

BDD and TDD are related approaches to software development that came out of the Agile movement and have proven to have a significant positive impact on software quality. This section provides an introduction to the concepts so testers can be prepared to adopt them together with developers and other project members using iterative development methods. This is a bonus section that is discussed as time permits.

1. Test-Driven Development activities
2. Behavior-Driven Development activities
3. Tools for Different Languages

Part 10: Managing Testing Projects
Whether you lead a team of testers or work as the lone tester on a project, effectively managing the testing work is key to your ability to successfully test the software on time with the resources at hand. In this section, we will address the basics of managing your work in a way that is relevant to individual contributors and lead leads alike.

1. Planning for Testing (Universal Testing Method Steps 1-4)
2. Requirements Traceability
3. Test Resource Needs
4. Testing Risks and Issues
5. Testing Entry and Exit Criteria
6. Measuring Testing Progress

Sep 15 - 17, 2025 | 12:00 - 4:30 PM EDT

Oct 20 - 22, 2025 | 12:00 - 4:30 PM EDT

Nov 17 - 19, 2025 | 12:00 - 4:30 PM EST

Dec 17 - 19, 2025 | 12:00 - 4:30 PM EST

FUNDAMENTALS OF SOFTWARE TESTING

Course Code: 2515

PRIVATE GROUP TRAINING	2 Day

FUNDAMENTALS OF SOFTWARE TESTING

Course Code: 2515

PRIVATE GROUP TRAINING	3 Day

Visit us at www.globalknowledge.com or call us at 1-866-716-6688.