Testing For Developers with xUnit

Abstract

High quality automated unit tests are one of the key development practices that enable incremental development and delivery of software. XUnit is the generic name given to the family of tools/frameworks used by developers when developing automated unit tests. JUnit, NUnit, MsTest and CppUnit are some of the better known members of the family.

This 2 or 3 day course provides the participants with a vocabulary of test smells and patterns with which to elevate their craft when writing test code. These smells give them a way to reason about the quality of their test code as well as a set of reusable test code design patterns that can be used to eliminate the smells. It also describes techniques for choosing the right set of tests and how to organize the test code including naming standards.

While the course is designed to be equally applicable to test-driven and test-after styles of development, the interplay between when the tests are written and how the system is designed for testability (and how this impacts one's test automation strategy) is highlighted to help participants to make an informed choice.

Benefits

The participant should finish with a better understanding of why automated unit tests help us build better quality software with a lower life-cycle cost and how to design and code the tests to achieve this outcome in a consistent and predictable fashion.

Course Outline

Testing for Developers with xUnit introduces basic testing and test automation techniques, including:

- how to think like a tester,
- how to determine what tests to write,
- how to use the XUnit testing framework, and
- techniques for making tests maintainable.
- Techniques for improving test coverage

Prerequisites:

- At least 6 months experience programming in language of training
- Most students will get significantly more out of this course if they already have prior experience in using an xUnit-based open-source testing framework.

The 2-day course consists of 10 modules covering a broad range of topics concluding with testing indirect inputs and outputs. The 3 day course adds modules on Design-for-Testability, Testing Legacy Code and Test-Driven Development While the course is largely process agnostic, that is, it applies equally well to Test-Driven Development and Test-After Development, it does talk about the advantages of automating tests before the software is written and debugged.

Time is split roughly 50/50 between theory/presentation/discussion and hands-on test coding exercises. The exercises are currently available in Java, C# and C++. Additional languages could be added given sufficient demand.

The standard course can be customized to suit your specific needs. For example, it can augmented with additional modules on advanced topics such as Test Strategy, and Data-Driven Tests as described in

Gerard's book, "xUnit Test Patterns – Refactoring Test Code" and with modules on using tools such as FIT to do keyword and data-driven functional test automation.

Learning Objectives

Participants should leave the course with at least a basic understanding of:

What is the role of automated unit testing?

- What is developer testing?
- How to "think like a tester"
- How to determine which tests to write?
- Why should I automate tests?

How to automate tests with XUnit

- Basic mechanics of test automation
- What tests should I automate?
- Proper use of assertions
- Testing of exceptions
- Ways to set up the test fixture and when to use which
- Use of stubs to isolate the class being tested
- Common problems and how to avoid them
- Bad smells (symptoms) and what they mean

How to get the best "bang" for your testing dollar

- When to write the tests
- How to verify the tests
- How to make the tests maintainable and useful as documention
- How to make tests portable (run anywhere)
- How to make tests repeatable and robust (provide consistent results even as the system evolves)

Detailed Course Outline

Each module consists of a presentation of the theory and examples followed by an exercise where the students get to put the theories into practice. On average, about 40% of the time spent on each module is spent doing and debriefing the hands on exercises.

1. Overview
   - An overview of the learning objectives ,the topics that will be covered in the course and the teaching style.
2. Introduction to Testing
   - Introduction to the basic concepts and terminology of software testing:
   - Unit vs Component vs. Functional Tests.
   - Black box vs. White Box tests
   - Acceptance Tests vs Regression Tests
3. Basics of xUnit
   - Introduction to the basic concepts and terminology of the JUnit (or equivalent) software testing framework.
   - Patterns: Simple Success Test, Expected Exception Test
   - Smells: Obscure Test
4. Unit Testing Methodology
   - How to determine the minimal set of test conditions required to test a piece of software sufficiently.
   - When should tests be automated? How does this impact the development process?

- Test Conditions, Test Cases & Test Scenarios
- Equivalence Classes & Boundary Conditions

5. Testing Style
   - How do you ensure tests are understandable and easily found?
   - Economics of test automation; how automated testing can reduce cost.
   - How should test case classes and methods be named and organized?
   - Techniques for verifying the tests: Test Inspections, Testing the Tests, Code Coverage
   - Goals of unit testing: Tests as Specification, Tests as Documentation, Fully Automated Tests
   - Smells: High Test Maintenance Cost, Hard-to-Test Code
   - Patterns: Testcase Class per Class, Testcase Class per Feature, Testcase Class per Fixture
6. Introduction to Functional and Component Testing
   - How to define and automate functional tests.
   - Using use cases as a source of test conditions.
   - Patterns: Subcutaneous Test, Layer Test
   - Smells: Indirect Testing
7. Verifying Expected Outcomes
   - How to use Assertions properly to make tests valuable as a specification.(intent revealing) while verifying that actual outcomes match expected outcomes without over-specifying the system behavior.
   - Verify expected outputs using Guard Assertions, Expected Objects & Custom Assertions (Patterns)
   - Smells: Obscure Test, Fragile Test
8. Setting up the Test Fixture
   - How to initialize the test fixture required to run a test using various fixture setup strategies. Common problems and how to solve them.
   - Test robustness and repeatability
   - Smells: Slow Tests , Erratic Tests (Interacting Tests, Unrepeatable Test, Non-Deterministic, Test Run Wars),
   - Patterns: Fresh Fixtures vs Shared Fixtures, Creation Methods & Finder Methods
9. Testing Indirect Inputs
   - Using stubs and real components to verify proper behavior for all inputs from used components
   - Testing handling of Exceptions thrown by used components (and how to force them to be thrown)
   - Smells: Indirect Testing, Obscure Test, Erratic (Non-Deterministic Tests)
   - Patterns: Test Double, Test Stub
10. Testing Indirect Outputs
    - Using stubs and real components to verify expected outputs via used components
    - Patterns: Test Spy, Mock Object, Fake Object
11. Design for Testability
    - Designing software so that it can be tested easily. Includes how to enable use of stubs.
    - Patterns: Layer Tests, Dependency Inject, Dependency Lookup, Humble Object, Fake Object
    - Smells: Hard-to-Test Code, Test Double
12. Testing Legacy Code
    - What makes software “legacy”?
    - What makes testing legacy software so difficult?
    - How to get started
    - What tests to write first?
    - Breaking Dependencies using Seams
13. Test-Driven Development
    - How to improve quality and reduce cost by writing unit tests before writing code.
    - How can you avoid going into “software debt” as you write software?
    - What is “emergent design” and how is it done?

- The Red-Green-Refactor cycle.

Text Book:

"xUnit Test Patterns – Refactoring Test Code" by Gerard Meszaros
Published by Addison Wesley Professional, May 2007

- ISBN-10**:** 0131495054
- ISBN-13: 978-0131495050

Customizing the Course:

The course may be adjusted to suite the audience several ways. Based on the level of experience of the participants, some modules may be skipped or covered more quickly. Additional modules may be added to cover specific topics including anything covered in "xUnit Test Patterns – Refactoring Test Code". Sample topics include:

- Test Automation Strategy
  - How to define a strategy for testing your system that maximizes test coverage while minimizing test effort.
  - Patterns: Recorded Test, Scripted Test, Data-Driven Test, Keyword-Driven Test, Layer Testing, Round-Trip Test, Layer-Crossing Test, Built-in Test Recording & Playback, Record&Refactor
- Data-Driven Testing
  - When should Data-Driven Testing be used? Advantages and disadvantages.
  - What tools can be used for Data-Driven Testing? Home-built, open-source, commercial.
- Keyword-Driven Testing
  - When should Keyword-Driven Testing be used? Advantages and disadvantages.
  - What tools can be used for Keyword-Driven Testing? Home-built, open-source, commercial.
- Testing Embedded Software
  - How can embedded software be unit tested?
  - How unit testing can reduce the cost and improve the quality of embedded software.

More Information

Please contact Gerard Meszaros at mailto://xunitTraining@gerardMeszaros.com or 1-403-827-2967 for information on pricing and availability

About the Instructor

Gerard Meszaros is an independent software development consultant with 25 years experience in software and a nearly a decade of experience applying agile methods. He is a thought leader in agile user stories, agile usability, test automation patterns, refactoring of software and tests, and design for testability and has applied automated unit and acceptance testing on projects ranging from full-on eXtreme Programming to traditional waterfall development in wide range of industries. He is also a leading expert in the implementation and customization of agile methods such as Scrum and eXtreme Programming and has been one of the early proponents of including usability practices on agile projects.

Gerard has been a frequent presenter of papers and tutorials at major international conferences. He offers on-site training courses on Rightsizing User Stories, Testing for Developers with xUnit and Fit, and Use Cases for Agile Projects. His book xUnit Test Patterns – Refactoring Test Code was published in May 2007 by Addison Wesley Professional in the Martin Fowler Signature Series and won a Jolt Productivity Award in the Best Technical Book category. He was the "star" of the winning pair in the Programming with the Stars competition at Agile 2009 beating out notable agile programming stars such as J.B. Rainsberger and James Grenning.