ICON
TECHNOLOGY CONSULTING
Technology & Process for Software Development

IconTechnologyConsulting.com
(636) 530-7776

Test Driven Development (TDD) Workshop

The practice of Test Driven Development (TDD) allows an agile development team to efficiently convert user defined requirements into a well-designed, fully tested working implementation that not only meets the business need, but will also be easy and safe to extend, maintain, and enhance over time. Rather than spending weeks or months on an extended "design phase," agile teams use TDD to evolve and optimize their design and implementation over time to meet the team's evolving understanding of the business requirements.

You will learn the basic skills and gain the hands-on experience required to implement TDD in your agile project. You'll be introduced to TDD related skills and concepts in context while working to turn example business requirements into working code.

As a technical workshop, this course emphasizes hands-on coding and the use of widely adopted development tools and testing frameworks for the Java development language. The course can be modified to use real-life examples and situations from the students' own experiences. It would also benefit a new Agile project team that attends as a group, providing a common foundation of knowledge and experience for everyone on the team.

Learning Objectives:

- Sketching an implementation model from user requirements, emphasizing the assignment of responsibilities to classes, components, and sub-systems.
- The TDD Test-Code-Refactor cycle.
- Coding by intention, and triangulating to arrive at an optimal solution.
- Using unit tests as executable specifications that document the behavior of the code that they test.
- Detecting common design deficiencies or "code smells" and improving them through refactoring.
- Isolating units to be tested using test doubles and mocking frameworks.
- The relationship between user stories, confirmations, acceptance tests, integration tests, and unit tests.

Audience:

This course is primarily intended for hands-on developers. While relatively little prior experience in TDD or writing unit tests is assumed, participants should be equipped and prepared to engage in hands-on coding during the course.

Pre-requisites:

Participants should be proficient with the Java programming language and its commonly used libraries and APIs, and at least somewhat familiar with the Eclipse IDE.

ICON
TECHNOLOGY CONSULTING
Technology & Process
for Software Development

IconTechnologyConsulting.com
(636) 530-7776

System Requirements:

JDK (Java Development Kit version 1.5 or later, Eclipse IDE version 4.0 or later, JUnit version 4.X and JUnit plug-in for Eclipse, Mockito framework version 1.9 or later.

Duration:

3 days

Outline:

1. **Conceptual Overview**
   - TDD What and Why
   - The Test-Code-Refactor Cycle
   - TDD in an Agile Development Process
2. **Unit Test Mechanics**
   - Setting Up the Environment
   - The Structure of a Unit Test Class
   - Assertions and Matchers
   - Handling Error Conditions
3. **Isolating Code with Test Doubles**
   - What are Fakes, Stubs, and Mocks
   - When and Why to Use Doubles
   - Basic Usage of the Mockito Framework
   - Advanced Mocking Scenarios
4. **Intro to Emergent Design**
   - Identifying Common "Code Smells"
   - Applying Object Oriented Design Principles
   - Applying Common Refactorings
   - Refactoring to Design Patterns
5. **A Basic TDD Episode**
   - Analyzing a Simple Requirement
   - Writing a First Test by Intention
   - Making the Test Pass
   - Completing the Specification with Triangulation
   - Refactoring to Improve the Design
6. **Exercise 1: Test Driving a Story Part 1 (Code Along with the Instructor)**
   - Our Example User Story
   - Sketching an Approach
   - Test Driving the First Confirmation
   - Using Test Doubles
   - Refactoring to Improve the Design
7. **Exercise 2: Test Driving a Story Part 2 (Code Along with the Instructor)**
   - Test Driving More Confirmations
   - Managing Test and Mock Code
   - Design Principals and Patterns
   - Managing and Hiding Complexity
   - More Refactorings
8. **Exercise 3: Test Driving a Story Part 3 (Code In Pairs)**
   - More Confirmations
   - Extending the Solution Safely
   - Clarifying Conversations with the Product Owner
   - Independent Solution Review
9. **Unit, Integration, and Functional Tests**
   - Leveraging Unit Test Code for Integration Testing
   - The Relationship Between Confirmations, Functional Tests, and Unit Tests
   - Key Collaborations Between Developers, QA, and Product Owner
10. **TDD in an Agile Process**
    - Automatic Unit Testing During Builds
    - Unit Testing Version Control, and Continuous Integration
    - Defect Elimination Using TDD