Certified Scrum Developer Workshop

Length: 3 Days

Learn test-driven development, continuous integration, refactoring, and emergent design for Agile technical excellence.
After teams have been exposed to the principles of Scrum, they are left wondering about the more technical aspects of software development and delivery and how these areas fit into an iterative approach. This course extends important Scrum principles to their logical applications in several technical, foundational areas of software development.

Borrowing from a variety of Agile disciplines, the course will introduce and explain a wide range of available tools and techniques that can offer team flexibility in their Agile approach. Each subject area is accompanied by real-world examples of how they have been used in the past and the success criteria for each instance.

Based on the understanding that theory and principles solidify into practices and habits through experience, the course is comprised of numerous hands-on exercises that demonstrate the use and benefit of each practice area and technique. You will not only understand the mechanics of each technique, but you will also participate in the discovery and discussion of how to ensure the creation of value in your organization.

The goal of this workshop is to put to use what you learned through books and lectures to determine how to utilize that knowledge, including how to ensure that your team is improving over time. With this proven approach, you will leave the class with practical knowledge and a clear roadmap for your team's success.

What You'll Learn

- Implement test-driven development to minimize the possibility of defects reaching the production environment
- Develop the correct technique for continuously integrating your newly developed code into the existing code base
- Apply Agile and Scrum principles and best practices to form the best mix for your team's success
- Different approach to Agile architecture and design that supports a more incremental and emergent project
- Transform your development processes to reflect the most efficient approach given your organization's constraints
- Pitfalls that poorly disciplined Agile teams fall into, contributing to failed Agile adoption and implementation attempts
- How to adopt Agile practices effectively within the context of your existing software development framework
- Utilize refactoring to ensure a longer lifespan of your software
- Gain practice in organizing your group into a self-managed team
- Develop the correct technique for continuously integrating your newly developed code into the existing code base

- Conduct exercises in a real-world Agile development unit and see firsthand how the roles mingle together to get the work done through collaboration
- Discover how continuous, incremental improvement will allow your team to continue growing long after the conclusion of the course
- Using Scrum as the backdrop, acquire techniques for successfully scaling Agile across your teams and the enterprise
- Using the provided working examples of code, dive into Agile Engineering practices no matter what development framework you use.
- How the appropriate level of planning will reduce rework and waste in your architecture designs
- Pattern of behavior for Agile engineering excellence: You will not get lost in the code
- How to adapt from a plan-driven approach to continuous planning, starting with the inclusion of actual velocity-driven aspects into your planning, scheduling, and tracking
- Coaching and communicating skills of a ScrumMaster and the differences between what a ScrumMaster does compared to a traditional Project Manager
- Run multiple iterations using real-life scenarios
- Make your first hands-on experience with Agile a classroom experience not a production experience

Who Needs to Attend: Business and technical product analysts, project managers, software engineers/programmers, development and project managers, product managers and analysts, QA engineers
Prerequisites: Experience with software development, engineering, designing, and testing

COURSE CONTENT

1. AGILE PRINCIPLES & PRACTICES

2. FEEDBACK & PLANNING

Triple Constraints

Five Levels of Planning

User Stories

Relative Sizing

Sprint Execution

Sprint Demo

Team Retrospective

3. COLLABORATION

Customer Collaboration

Team Collaboration

Pair Programming & Pairing

4. ARCHITECTURE, DESIGN, & SHARED UNDERSTANDING

Architecture as a Concern

Design Principles

Coding Standards

Collective Code Ownership

Simple Design

System Metaphor

Testability as a Driving Concern

5. TEST DRIVEN DEVELOPMENT (TDD)

Test First vs. Test Last

TDD Rhythm: Red, Green, Refactor

TDD influence on Design

Unit Testing Principles

6. REFACTORING

Safety Net of Tests

Refactoring Patterns

Refactoring for Maintainability

7. CONTINUOUS INTEGRATION

Attitude of Continuous Integration

How and Why You Must Develop a Single Command Line Build

Automating the Build

The 10-Minute Build

Benefits & Practices of Continuous Integration

8. TESTING PRACTICES

Testing Quadrants

Automation

Unit Tests

Integration Tests

Acceptance Tests

Functional Tests

9. ACCEPTANCE TESTING

Acceptance Criteria

Writing Acceptance Tests

Acceptance Test Driven Development

Automating Acceptance Tests

10. ADOPTING SCRUM DEVELOPER PRACTICES

Recap Essential Scrum Developer Practices

Team Ground Rules to Start off on the Right Foot

Develop a Roadmap Leveraging Scrum Developer Practices

EXERCISES:

EXERCISE 1: DEFINING OUR CHALLENGE

Course participants will openly discuss challenges facing their current development teams. We will discuss common myths and misperceptions of both the Agile discipline and the Agile developer.

EXERCISE 2: FEEDBACK & CONTINUOUS IMPROVEMENT

Course participants will work through a set of exercises aimed at reinforcing the importance of feedback throughout the lifecycle of an Agile project from requirements to execution.

EXERCISE 3: SELF-ORGANIZING TEAMS

Through a series of interactive exercises, course participants will experience the chaos created by individuals without common goals. Ultimately the team will be allowed to self-organize with minimal direction and experience the harmony and balance of a self-organized team.

EXERCISE 4: PAIRING

Using the techniques introduced by pair programming, course participants will solve a problem in pairs. This exercise will demonstrate the value and effectiveness of pairing to develop better solutions and to increase shared knowledge of the team.

EXERCISE 5: CODING STANDARDS

Course participants will discuss thoughts and current practices regarding coding standards. Instructor will provide a working software example for the team to evaluate in the absence of coding standards. The class will discuss the potential waste effort that can exist in the absence of coding standards. Course participants will then work in teams to develop a simple set of coding standards. After applying the agreed upon coding standards they will look at the inherent benefits.

EXERCISE 6: TEST DRIVEN DEVELOPMENT

Using Test Driven Development (TDD), course participants will develop a Test List and follow the TDD Red, Green, Refactor cycle to develop software and meet the instructor's requirements. Course participants will experience the cadence and rhythm of the TDD process.

EXERCISE 7: REFACTORING

Instructor will provide working software and a test suite of unit tests. Using refactoring methods and patterns, course participants will incrementally refactor the software to achieve a simpler design and increase quality and maintainability.

EXERCISE 8: CONTINUOUS INTEGRATION

Participants will define and execute a Continuous Integration process using a popular open-source Continuous Integration product. This exercise will reinforce the key tenants and practices of Continuous Integration and serve as a discussion opportunity on how to effectively utilize and leverage Continuous Integration to support the developer and the team.

EXERCISE 9: ACCEPTANCE TESTING

Course participants will develop a set of Acceptance Tests (Story Tests) from a sample set of User Stories containing Acceptance Criteria. Using Acceptance Test Driven Development (ATDD), course participants will use these Acceptance Tests to drive development of the sample User Stories.

EXERCISE 10: CREATE A ROADMAP AND ACTION PLAN

Course participants will prioritize the Agile engineering practices that they might want to introduce to their current projects, teams, and organizations. Using the three highest priority concepts, course participants will create a plan to bring these practices into action. The class will compare and discuss action plans.