# Implement the Built-In Quality of SAFe® with Sustainable TDD

## SUMMARY

Among other things, Agile Software Development requires a high degree of flexibility in the coding process. As we get feedback from clients, stakeholders, and end users, we want to be able to evolve the design and functionality to meet their needs and expectations. This implies an incremental process with frequent (almost constant) change to the code. Each change is an opportunity to make the product more appropriate to the needs it is intended to address. However, change can also be dangerous if it is not accomplished in a way that avoids code decay and protects against introducing bugs. This course teaches the skills of unit testing, refactoring, and incremental development. It goes even further: unless developers are trained about which tests to write, how to write them, when to refactor code, and techniques for breaking dependencies, testing can become unsustainable as projects mature..

## DESCRIPTION

Built-in Quality is one of the four Core Values of SAFe®. The ability of the enterprise to deliver new functionality with the fastest sustainable lead time and to be able to react to rapidly changing business environments is dependent on Solution quality. But built-in quality is not unique to SAFe. Rather, it is a core principle of the Lean-Agile Mindset, where it helps avoid the cost of delays associated with recall, rework, and defect fixing. The Agile Manifesto is focused on quality as well saying, "Continuous attention to technical excellence and good design enhances agility". There can be no ambiguity about the value of built-in quality in large-scale systems. It is *mandatory*.

Test-First is a philosophy that encourages teams to reason deeply about system behavior prior to implementing the code. This increases the productivity of coding and improves its fitness for use. It also creates a more comprehensive test strategy so that the understanding of system requirements is converted into a series of tests, typically prior to developing the code itself.

In Test-Driven Development, developers write an automated unit test first, run the test to observe the failure, and then write the minimum code necessary to pass the test. Primarily applicable at the code method or unit (technical) level, this ensures that a test exists and tends to prevent gold plating and feature creep of writing code that is more complex than necessary to meet the stated test case. This helps requirements persist as automated tests and remain up to date.

This course blends lecture, demonstration, and hands-on coding exercises. When you leave this course, you will have performed test-driven development, refactored both new and legacy code, created mock objects and injected them for testing, and understand how to keep your test suite from becoming a maintenance problem as it grows. You will also learn how testing, done correctly, can improve both your analysis and design skills.

For more, see *www.scaledagileframework.com/built-in-quality* and *www.scaledagileframework.com/test-first*.

*Some of this description is adapted and reproduced with permission from Scaled Agile, Inc. Copyright © 2011-2017 Scaled Agile, Inc. All rights reserved.*

### We are ideally suited to deliver this course

Net Objectives has been doing Agile at scale since 2004. We have pioneered dozens of practices that have now become commonplace in the Agile community. Some of have become integrated into the SAFe framework and other have not yet such as ATDD and architecture.

The Net Objectives team has provided thought leadership in TDD, ATDD, and Emergent Design for over 15 years and published multiple award winning books in the area.

*Note: This course is neither authorized by Scaled Agile Inc., nor provides any certification related to SAFe.*

CONTACT US
info@netobjectives.com
1.888.LEAN-244 (1.888.532.6244)

LEARN MORE
www.NetObjectives.com
portal.NetObjectives.com

Copyright © Net Objectives, Inc.

*SAFe® and Scaled Agile Framework are registered trademarks of Scaled Agile, Inc.*

## COURSE OBJECTIVES

This course teaches participants the sustainable approach to Test-Driven Development. The practice of Test-Driven Development, which utilizes Refactoring and Unit Testing in a particularly fine-grained way, is demonstrated. A hands-on TDD project will dominate the third day.

## LEARNING OBJECTIVES

At the end of this workshop, you will be able to:

- Understand why and how to be "test-driven"
- Analyze requirements to obtain better tests
- Write unit tests
- Use mock objects
- Refactor legacy code
- Understand how to use an XUnit testing framework (Junit or NUnit or CxxTest)
- Employ a variety of refactoring techniques using hands-on exercises to solidify this knowledge

## COURSE OUTLINE

1. Techniques
    - A brief overview of the motivations behind Agility, and TDD specifically
    - As a group, we all get the testing framework up and running on all needed machines.
    - Review Code Qualities and how they relate to Testing and Testability
    - Unit-testing and the benefits of TDD
    - Unit-testing/TDD exercise
2. Design
    - Mock Objects (with an exercise)
    - Code Smells and Refactoring
    - Refactoring to the open-closed, just-in-time design
    - Legacy code refactoring exercise and debrief
3. Reality
    - TDD and Design: revisiting the exercise from Day 1
    - TDD and Design: putting it all together. This includes a larger hands-on lab project.

## LEVEL

Intermediate

## TARGET AUDIENCE

Software developers in C#, Java, or C++ who want to learn how to code better and learn new design techniques.

## ATTENDEE MATERIALS

Workshop materials are provided at the start of the class.

## ROOM SETUP AND EQUIPMENT

One computer per two students.

Students usually sit at tables, 4-6 students per table.

Flip chart and whiteboard for the instructor.

A projector with screen.

## PREREQUISITES

Experience in C#, Java, or C++

## COURSE LENGTH

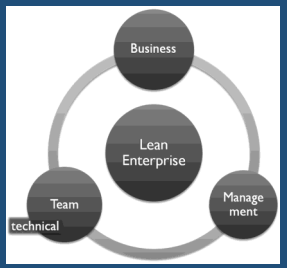
Three days

## MAXIMUM NUMBER IN CLASS

24 (Size depends upon the composition and experience level of the team.)


## NET OBJECTIVES

We are committed to delivering the principles, practices, and perspectives that businesses must know in order to maximize their return on their technology solution and software development efforts. We combine our experience and a time proven approach based on lean thinking to continuously extend the capability of what is possible in creating effective technology delivery organizations (IT or product). We provide these learned methods to our clients to assist them in achieving their goals and in assisting them in making their organizations more successful.




Full course descriptions may be found at
www.NetObjectives.com/training

Lean ● Agile ● Kanban Patterns ● TDD ● ATDD ● Assessments ● Consulting Training ● Coaching