Introduction to Agile Development

Duration:	3 days
Type:	beginner

Description

This course is a two part introduction to Agile Development. The first part is a theoretical discussion of what distinguishes Agile methods from traditional processes. The second part is an in-depth practical introduction to Agile tools and techniques. Examples are given from many different environments, including Java, C#, Ruby and Perl.

The first part of the course is suitable for anyone involved in softeare development. The second is designed for software developers only.

Prerequisites

For the second part of the course delegates should be software developers with experience of working on medium and large scale projects.

List of Modules

Existing Development Methods

- Why no one really does waterfall development
- When unstructured development works and fails
- When bureaucratic development works and fails
- The good news about established methods
- The weaknesses of existing methods

Summarizing Agile Development

- Delivering software is everything
- Agile methods value developers
- Agile methods value customers
- Iteration is at the core of Agility
- Test, test, test and test again
- Automate absolutely everything

Comparing Three Agile Processes

- Feature Driven Development (FDD)
- The SCRUM Methodology
- Extreme Programming (XP)

When Agility Isnt Enough

- When being Agile is not worthwhile
- Can Agility work in large projects?
- Can Agility survive corporate culture?

Key Skills for Agile Developers

- Coding using Test Driven Development
- Continuous integration and source control
- Build automation and the 'Big Red Button'
- Refactoring to increase code quality
- Rapid modelling using sketches

Introducing Test Driven Development

- Defining your intent through tests
- Writing just enough code to pass
- Adding tests and refining the code
- Testing up to the point of boredom
- Triangulating on hard problems
- Moving up and down the gears
- You aren't going to need it
- Building a suite of test cases

Refactoring in Depth

- Refactoring as the 'second hat'
- Refactoring is essential to TDD
- Support for refactoring in the editor
- Refactoring to keep the code alive
- Detecting smells in code and tests
- The most productive refactorings

Automating the Build Process

- Introduction or review of Ant/NAnt/MSBuild
- Triggering Unit Tests and creating reports
- Calculating test coverage of your code-base
- Choosing and generating metrics with value

Architecture and Modelling in an Agile Project

- Common myths about Agility and architecture
- Defining an architecture in small increments
- Estimating and coping with technical risks
- How Agile developers model their designs
- Models as sketches rather than blueprints

Lightweight Containers and Mock Objects

- Problems testing classes with dependencies
- Using mocking to replace dependencies
- Different types of data collection in mocks
- Errors and 'Crash Test Dummy' mocks
- Using the test as the mock (Self Shunt)
- Automatically generating mock objects
- Externalizing dependencies using Factories
- Dependency Injection and Inversion of Control
- Using Spring to manage class dependencies