Agile Software Development

Methodology, Values, Principles, User Stories, Pairing, Individual/Team Dynamics, Test First, Refactoring

Agile software development is a highly productive strategy for organizing project teams. It allows software to be quickly written for today's needs and that can easily evolve to keep up with an every-changing business environment. It exploits the best of individual and team creative efforts. Agile shuns the heavy "ceremony" of documentation-rich rigid software development processes; and instead advocates a code-centric approach with just enough documentation to genuinely satisfy all stakeholders' needs. This approach is more satisfying for developers and is more finely aligned with customers' true requirements. It results in much richer contributions from all project participants. It advocates short iterations that fulfills the "release early and release often" aspiration. Teams are based on a flat organizational structure with a high-degree of face-to-face communication with a "product owner". It treats the software development business as an iterative subscription model, rather than a once-off event. It is a simpler approach to writing software – and easier for a team to use. In today's highly competitive environment, whichever software team can write quality code fastest wins. This course is ideally suited to dev teams making the agile move to gain competitive advantage.

Target Audience

All members of software project teams – managers, architects & programmers - and the product owner.

Prerequisites

Good understanding of software development issues

Significant project experience (either as a specifier or an implementer) .

Contents of One-Day Training Course

Efficient Light Methodology

Current state of play in s/w projects
Need for new approach to programming
Families of agile methodologies and selecting best techniques from each
"An ecosystem that ships software"

Values

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Custom collaboration over contract negotiation
Responding to change over following a plan

Principles

Satisfying customers, welcoming changes, delivering working software, work together, projects by motivated individuals, face-to-face communications, progress measured by running code, sustainable development, quality matters, simplicity, self-organizing teams, reflection

Individual & Team Dynamics
High communication levels
Shared knowledge spaces
Benefits of higher-skilled developers
Handling a creative team
Collaborative & competitive

Design Documentation

Doc strategy – succinct yet sufficient
Documentation for all project stakeholders
Useful templates for project documents

Agile Projects

Agile project management
Iterations and releases

Requirements via User Stories

What is a user story and how to create one?
Certain number completed each iteration

Design

Architecture with a little 'a'
UML as a sketch
Design today for today; refactor tomorrow

Test-First

Written by programmers (unit tests) & customers (acceptance tests)
Putting the concept into practice

Pair Programming

"A driver and a navigator" (think rally car racing) - not "a driver and a passenger"
Duties of the navigator
Continuous code review

Refactoring

What happens when software matures?
"Improve the design after the software has been written"
Business Issues
Fixed price/fixed-scope contracts, responsibilities, outsourcing, etc.

Tools

Continuous Integration/Delivery tooling
Unit Test tools
Refactoring tools

Sample Project

Exploration of the use of agile processes in a case study development project