Total Development Workshop in C#

Hands-on Agile Development from Requirements to Code

Current trends in development processes emphasise agility and responsive design over bureaucracy and big up-front design. At a high-level the (Rational) Unified Process (RUP) provides a suitable approach for agile development when taken together with a number of complementary practices such as Test-Driven Development (TDD) and Scrum. The Unified Modeling Language (UML) is the industry standard notation for describing object-oriented systems. The notation includes a number of diagram types that can be applied to many development activities. RUP provides a common framework intended for UML-based development processes, both heavy and light. It outlines four basic phases in a lifecycle: *inception*, *elaboration*, *construction* and *transition*.

The *Total Development Workshop in C#* course introduces a useful subset of the core modelling notation in UML and follows the first three RUP phases in its structure, with TDD forming the backbone of the construction phase. Groups of course attendees frame requirements and high-level designs which they then take through to code and tests, over four construction mini-iterations. The workshop balances taught material with practice, introducing sufficient modelling notation, use case techniques, testing practices and refactorings as necessary. This workshop structure allows course attendees to see and understand how all the activities fit together in practice, consolidating concepts that are often taught theoretically and separately.

Objectives

- Outline an agile development lifecycle based on RUP phases and Scrum iterations
- Present a useful working subset of UML notation, highlighting common techniques and pitfalls
- Describe test-driven and continuous design practices
- Learn to write use cases and estimate and plan against them
- Put the concepts into practice following the development lifecycle for a simple project

Audience

The course is suitable for software developers with C# experience who wish to combine their existing programming skills with techniques and tools for agile development. Any previous experience with UML, patterns and agile development is an advantage but not a requirement.

Content

Overview Managing change · Agility · Informal and continuous design · The role of testing versus debugging · Iterative and incremental development · RUP and RUP-based processes · XP · Scrum · The role of UML · The inception, elaboration, construction and transition phases

Inception System scope and requirements · Context diagrams · Actors · Use cases and use case diagrams · Identifying use case goals · Common use case and diagramming pitfalls · Capturing the information model of the problem domain with class diagrams

Elaboration Defining use cases in more detail · Activity diagrams · Interaction diagrams · Sequence diagram pitfalls · Using class diagrams to capture high-level design ideas and discussions · Using state-machine diagrams to explore object lifecycles · Outlining a loosely coupled baseline architecture with package and component diagrams · The role of prototyping · Estimating against the package model · Use cases and the product backlog

Construction Iteration planning and the sprint backlog · Test-driven development · Overview of NUnit · Unit-testing techniques · Coding guidelines · Dependency management patterns · Continuous integration · Sufficient design · Common refactorings · Executing four iterations · Expanding use cases into more detailed scenarios · Handling new requirements

Additional Details

Duration 4 days

Setup Projection facilities for a laptop · Whiteboards, flip charts and pens · Enough room for groups of three or four people to work together comfortably · Workstations (one per pair of developers) with suitable development environment installed

Contact Kevlin Henney · kevlin@curbralan.com · Curbralan Limited · +44 117 942 2990