

Component-Based Development with UML

An introductory course on the design of software systems using components

Delivery: On-site

Course Length: 3 days

Course Approach: Lecture, discussions, exercises

Level: Introductory

Audience

This course focuses on Architect, Designer and Developer competencies and skills but is suitable for all development team members.

Prerequisites

None

Recommended Experience

Working knowledge of object-oriented development concepts and some exposure to the Unified Modeling Language (UML) is recommended

Related Courses

For students interested in continued learning in this field, we suggest:

- Use-Case Modeling
- Principles of Software Architecture

Course Description

Learn how to develop extensible solutions from software components.

Components are a widely used technique for managing complexity and delivering robust and flexible solutions. Today, most new software is component-based, but there are still many challenges to finding and developing good components. This course provides the skills necessary to develop systems by identifying, designing and using components.

The course follows a practical step-by-step process for moving from an understanding of the requirements, through the use of architecture and analysis techniques for identifying good components with well-defined responsibilities, to the specification, implementation and testing of a set of components to meet the specified requirements.

The course uses a simple subset of the Unifying Modeling Language (UML) as a standard language for expressing the system requirements, architecture and design, but the underlying practices and skills are applicable to any component-based development project.

As part of the course the delegates contribute to the analysis and design of an evolving component-based development case-study.

Objectives

Upon completion of the course, delegates will be able to:

- Work from requirements captured as UML use cases
- Analyze requirements to identify components and their responsibilities
- Structure complex systems to support defined architecture goals
- Manage software complexity and efficient team-based development
- Design component systems that are modular and extensible
- Design for reliability, performance and other cross-cutting concerns
- Separate interface specifications from component implementations
- Design the internals of a component
- Identify and design tests for components and component-systems

Topics Covered

1. Principles of Component-Based Development
2. Component-Based Development Roadmap
3. Analysing the System Requirements
4. Structuring the Solution Architecture
5. Identifying Components
6. Describing Components
7. Platform-Independent Design
8. Maintaining Design Independence
9. Platform-Specific Design
10. Test-Driven Development
11. Implementing Cross-Cutting Concerns
12. Extending Component Systems

Europe

+44 (0)20 7025 8070

info-eur@ivarjacobson.com

Americas

978-649-2856

info-usa@ivarjacobson.com

Australia

1300 567 280

Info-aus@ivarjacobson.com

Asia

+8610 82486030

info-asia@ivarjacobson.com