

## Real-Time Software Design with UML

---

|                        |                                                                                 |
|------------------------|---------------------------------------------------------------------------------|
| Course category        | UML                                                                             |
| Training area          | Design Techniques                                                               |
| Course code            | OO-504                                                                          |
| Duration               | 5 days                                                                          |
| Additional information | Available for on-site delivery only. Can be delivered remotely or Face-to-Face. |

Modern embedded software design is constrained by an ever-growing set of constraints:

- Increasing software complexity
- Decreasing time-to-market
- The need to produce flexible, maintainable systems
- The need to have rigorous engineering in critical systems.

These factors are driving the need for cohesive design methodologies and the use of modelling.

This is a detailed software design course which focuses on designing Real-Time Embedded Systems, using UML 2 notation to document the proposed design.

The focus on design principles and methodologies make this course significantly different to most UML courses, which focus on notation.

### Course objectives:

- To provide an understanding of Object Oriented design principles.
- To show how to develop real-time software in a rigorous and systematic manner.
- To enable attendees to develop their own practical design skills.
- To teach effective application of UML notation.

### Delegates will learn:

- The fundamental concepts and terminology of real-time software.
- The diagrammatic and modelling underpinnings provided by UML for Object Oriented development.
- How to apply the design principles in real-time applications.
- The basics of an integrated, traceable and consistent approach in the development of software for real-time systems.

**Pre-requisites:**

- Some understanding of technical software development methods.
- Knowledge of typical embedded programming languages (like C) is useful.

**Who should attend:**

- Designers new to the area of real-time software design.
- Developers with some non-embedded UML experience.
- Designers embarking on projects using UML-based techniques for the first time

**Duration:**

- Five days

**Course materials:**

- Delegate handbook
- All worked examples and solutions

**Course workshop:**

Approximately 50% of the course involves practical application of the techniques discussed. Delegates undertake a complete embedded system case study that leads them through the software development process and modelling techniques introduced in the course.

The course specifically does not make use of a CASE tool. From our experience, a CASE tool distracts delegates from learning design issues and UML. However, the workshops clearly demonstrate the benefits and disadvantages of CASE tools, thus aiding CASE tool selection.

**Introduction to modelling****Why do we model**

- Left-shifting

- Models for discovery, understanding and construction
- Views and model consistency

### **The software development process**

- The PRAGMA methodology
- Evolutionary vs Adaptive models

### **Requirements model**

#### **System Scope**

- The context diagram

#### **Stakeholder analysis**

- Project stakeholders
- The stakeholder analysis framework

#### **Use cases**

- The use case diagram
- Scenario modelling
- Use case descriptions
- Organising use cases

#### **Use case interaction modelling**

- Sequence diagram basics
- Fragments
- Loops
- Scope-level sequence diagrams
- Selecting scenarios to model

#### **Ideal object model**

#### **Modularisation**

- 'Correct' vs 'good' software
- Principles of modularisation

## **Object Oriented design**

- Object-Oriented terminology
- Object-based design

## **Object modelling fundamentals**

- Finding objects
- Scenario-based design
- The CRC methodology
- Ideal object scenarios

## **Specification model**

## **Behavioural modelling**

- States machine
- Activity diagram
- Mapping internal and external behaviour

## **Concurrency**

- Active and passive objects
- Selecting active or passive elements
- The concurrency architecture

## **Classes**

- Class notation
- Associations
- Specialisation

## **Composite structures**

- Composite structure notation
- Ports
- Concurrency and composites

## **Interfaces**

- Dependency Inversion Principle

- Required and Provided interfaces

## **Implementation Factors**

### **Model transformation rules**

- The need for consistent transformation idioms
- Model-to-code transforms in C++ (C available on request)

**Feabhas Ltd** - PO Box 4259, Marlborough, SN8 9FJ, UK • [info@feabhas.com](mailto:info@feabhas.com) • [www.feabhas.com](http://www.feabhas.com)