

C7 - UML Real-Time

UML for embedded and real-time systems

Objectives

- Discover the mechanisms specific to embedded and real-time systems
- Use the UML language, with its real-time extensions, in the various analysis, design and coding phases of the development of an embedded system

During the course, trainees will develop, from its requirement specification, the model for a realistic real-time embedded system.

This will be done using *Papyrus* the free, Eclipse-based UML tool or, upon request, *Enterprise Architect Professional Edition*.

Course environment

- Printed course material (in English)
- One Linux PC for two trainees.

Prerequisite

- Basic knowledge of real-time embedded systems, their design and development.

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

Introduction to Real-Time

- Basic real-time concepts
- The constraints of real-time
- Structured and object programming
- Benefits of the object approach

Object oriented programming and UML

- UML history
- The standard UML diagrams
- The Object-Oriented development cycle

Real-time approach with UML

- Constraining diagram interpretation
- Defining new diagrams
- Development cycle with UML-RT

Modeling

The UML modeling language

- Static modeling
 - Use cases
 - Classes
- Dynamic modeling
 - Sequence diagrams
 - Collaboration diagrams
 - State-Transition diagrams
- Extending UML
 - Accessing the UML meta-model
 - Creating stereotypes
 - Creating Profiles

UML extensions for Real-Time

- Environment: System context diagram
- Constraints
- Behavior: State diagrams
- Timings: Extended sequence diagram, Timing diagram
- Parallelism: Software and Hardware architecture diagrams

The MARTE profile

- Overview
- The MARTE concepts
- Time Management in MARTE

System specification

Static modeling

- Specifying the context of the system
- Defining non-functional constraints
- Describing the use cases
- Identifying high-level classes
- High-level class model
- Iterative refining of each use case

Dynamic modeling

- Precising use cases with scenarios
- Adding time aspects in the sequence diagram
- System behavior and State-Transition diagrams
- Refining the object model by defining operations
- Adding interface objects in sequence diagrams
- Creating an interface object model

System design

Refining specification models

- System organization using subsystems and packages
- Class dynamic description

- Class-specific behavior diagrams
- Refining sequence diagrams
- Refining the class model
- Using third-party libraries (GUI, ...)
- Integrating input/output interfaces
- Handling data persistence and storage

Multi-process and multi-task handling

- The various software architecture kinds
- Identifying tasks
- Allocating subsystems to processors and tasks
- Inter-process communication
- Synchronization
- Creating a Software Architecture diagram

System architecture

- Partitioning in hardware subsystems (CPUs, boards, enclosures, ...)
- Kinds of hardware architectures
- Optimizing architecture choices
- Defining internal system interfaces (Bus, links, protocols, ...)
- Modeling the system with a hardware architecture diagram

Task execution environment

- Criteria for choosing the executive
- Interpretation of the real-time mechanisms depending on the architecture kind

Detailed design and coding

Preparing to code

- Detailed description of methods and attributes
- Optimizing the model
- Refining inheritance
- Abstract classes, templates...
- Association implementation (pointers...)

Incremental coding

- Implementing storage objects
- UML and programming languages
 - transformation UML -> C
 - transformation UML -> C++
 - transformation UML -> Java
- Increment coding
- Increment testing
- Delivery