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 Papyrusthe free, Eclipse-based UML tool or, upon request, Enterprise Architect Professional Edition.

Course Environment

- Theoretical course
  - PDF course material (in English) supplemented by a printed version.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Prerequisite

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

Target Audience

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

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.
- Trainee progress is assessed by quizzes offered at the end of various sections to verify that the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

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
- Prallelism: 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 th euse 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, ...)
- Integrationg 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

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

Detailed desing 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

Renseignements pratiques

Inquiry : 4 days