

# CEIS EMBEDDED SYSTEMS DEVELOPMENT COURSE

## USING RHAPSODY

### **Pre-Requisites:**

1. Programming knowledge in C/C++
2. Basic understanding of software engineering concepts, life cycle models
3. Understanding on embedded concepts, hardware board architectures and peripherals

### **Course Objective:**

#### **UML 2 Fundamentals**

This course is about the use of UML 2 for building embedded Real-Time software with Rhapsody. The goal of this training is to understand what UML is and to get familiar with the four most widely used UML diagrams: Use Case Diagrams, Sequence Diagrams, Class / Object / Structure Diagrams and State Machine Diagrams. The rest of the diagrams will be taught only in theory.

#### **Basic Rhapsody**

In this section, we learn how to create a new Rhapsody project, do some basic modeling using classes, attributes, operations, relations and state charts. Also how to generate and compile code and debug the model by injecting events, setting breakpoints, capturing behavior on sequence diagrams and visualizing the state of objects.

#### **Rhapsody case study**

After understanding the basics of using Rhapsody, we can create a small project where we will start with some requirements and gradually implement the project proceeding use case by use case. We will follow a spiral process here and use rapid prototyping.

#### **Advanced Rhapsody**

We will check the expertise of the trainee by trying to create a simple project without any step-by-step instructions. There are also many more topics that any Rhapsody developer needs to understand and so by the end of this section, you should have an understanding of topics such as CM, TestConductor, ATG, ReporterPLUS, COM APIs etc.

#### **Rhapsody hardware integration**

Application developed within Rhapsody would be deployed on ARM7 hardware (Phillips IC LPC2124) through cross compilation and USB connectivity.

## **Course content:**

The entire Rhapsody embedded systems development course is delivered in 3 phases as below:

Phase I : UML / Rhapsody theory, classes and lab excercises (80 hrs - 32 days with 2 hrs 30 mins on each training day)

Phase II : Student projects using Rhapsody (20 hrs)

Phase III : Q&A, Tests, Certification, review classes (20 hrs)

## **PHASE I**

Duration - 80 hrs

### **DAY 1**

- Introductory session
- UML fundamentals
- Introduction of all diagrams
- How to describe structure using UML
- Object discovery strategies

### **DAY 2**

- How to describe behavior using UML part I
- How to describe behavior using UML part II
- How to model communication using UML
- UML test papers and lab excercises

### **DAY 3**

- How to capture requirements using UML
- UseCase identification
- UML 2.1 ports
- UML test papers and lab excercises

### **DAY 4**

- Spiral process and ROPES
- Comparison with water fall model and V model
- UML test papers and lab excercises

## **DAY 5**

- Setting up Rhapsody in C++
- Properties
- Flow charts
- Lab exercise : Hello World

## **DAY 6**

- Design level debugging
- Test script
- Lab exercise : Count Down

## **DAY 7**

- Dishwasher relations
- Webify, web enabled debugging
- Lab exercise : Dishwasher

## **DAY 8**

- UML Interfaces
- Dishwasher interface
- Lab exercise : Dishwasher

## **DAY 9**

- CashRegister requirements analysis
- Rhapsody Gateway interface
- Lab exercise : CashRegister

## **DAY 10**

- UseCase 1 : Configure the products (Architecture modelling)
- Complete UseCase 1
- Lab exercise : Cashregister contd.

## **DAY 11**

- Rhapsody container classes usage
- UseCase 2 : Keep track of selected products
- Lab exercise : Cashregister contd.

## **DAY 12**

- Analysis and design sequence diagrams
- Complete UseCase 2
- Lab exercise : CashRegister contd.

## **DAY 13**

- Multithreading exercise
- Advanced Rhapsody
- Introduction to Rhapsody framework
- Lab exercise : OXF

## **DAY 14**

- Generating reports
- Introduction to Configuration Management
- Using COM APIs and helpers
- Lab exercise : ReporterPLUS

## **DAY 15**

- Stopwatch exercise (without trainer help)
- Rhapsody OXF session
- Target deployment – running Rhapsody application on hardware / simulator
- Lab exercise : PIM (Platform Independent Models)

## **DAY 16**

- Useful tips, common pitfalls
- Wrap Up session
- COM API exercise
- Lab exercise : Miscellaneous

## **DAY 17**

- UML Testing Profile (Theory)
- Overview of Design For Testability (DFT)
- Requirements based testing
- Sequence diagram test cases
- Lab exercise 1 : Stopwatch

## **DAY 18**

- Prepare a Test Plan
- Generate test cases with Test Conductor
- Run the tests via the command line
- Generate via the ReporterPLUS a test report
- Lab exercise 2 : Bluetooth Headset

## **DAY 19**

- Code based test cases
- Flowchart test cases
- Test report generation
- What is the ATG?
- Testing Strategies
- Configuration Management
- Limitations
- Lab exercise 2 : Code based TCs

## **DAY 20**

- Connecting requirements to test cases
- Additional Constructs
- Limitations
- Lab exercise : Radio (Assignments)
- Lab exercise : Home Alarm (Assignments)
- Lab exercise : Tetris (Assignments)

## **DAY 21**

- ReporterPLUS Training Setup
- Essential ReporterPLUS
- Generating Reports From Rhapsody
- Lab exercise : Creating a Template

## **DAY 22**

- Iterations and Tables
- Iteration Conditioning
- Iteration Sorting
- Lab exercise : Handling Missing Data

## **DAY 23**

- Bookmarks and Links
- Advanced ReporterPLUS
- Q-Language Expressions
- Graphics
- Lab exercises : ReporterPLUS template editor

## **DAY 24**

- Linking External Documents
- Documentation Standards
- Custom Word Templates and VBA
- Useful Tips
- Lab exercise : ReporterPLUS template editor

## **DAY 25**

- Overview of Requirements
- Gateway Overview
- Basic Gateway
- Importing Requirements from Word, DOORS etc
- Lab exercise : Gateway IDE

## **DAY 26**

- Analysing Results
- Taking Snapshots
- Producing Reports
- Advanced Gateway
- Lab exercise : Gateway editor

## **DAY 27**

- Creating & Managing Types with the Types Editor
- Attributes
- Filtering
- Adding a Prefix
- Lab exercise : Gateway editor

## **DAY 28**

- Tracing To Code
- Multiple User Working
- Troubleshooting & FAQ
- Rules

## **DAY 29**

- Rhapsody real time support
- PIM (Platform Independent Models)
- Multiple compiler support (Borland, MingW, etc.,)
- OXF(Object Execution Framework) study
- Lab exercise : Running on Linux as target

## **DAY 29**

- Hardware board setup and installations (ARM7 LPC2124)
- Setting up cross compilation project
- Flash programming
- Introduction of Rhapsody IDF
- Lab exercise : LED (ARM7 LPC2124)

## **DAY 31**

- Event Driven Framework
- Handling interrupts
- Lab exercise : LCD Stopwatch (ARM7 LPC2124)

## **DAY 32**

- Rhapsody integrations with other CASE tools
- Rhapsody integrations with other Rational products

## **PHASE II**

Duration – 20 hrs

The students would be assigned project work to develop embedded real time applications based on provided requirements specification. Rhapsody and its add-ons shall be used to analyse requirements, prepare high level and detailed design, coding and testing the application on host as well as target boards. Mentoring and guidance will be provided by the trainer throughout.

Students are required to demonstrate the project, prepare project reports and plans using ReporterPLUS.

## **PHASE III**

Duration – 20 hrs

This is the final phase of the course and would involve Q&A sessions, answering queries from students, Rhapsody written tests and certification.

Future of modeling RT and career guidance on Rhapsody embedded systems development would be given along with some Industrial case studies in automotive and avionics.

Review classes would be conducted on students' understanding of theory and lab assignments ending with a wrap up session.