

# Object Oriented Development Methodologies Training Track Outlines

## C001: Introduction to Object Orientation

### • Objectives:

- Introduce participants to the fundamentals of the Object Oriented Paradigm
- Understand the differences among various programming paradigm
- Understand the advantages of the OO paradigm
- Expose participants to different OO programming languages, and explain the differences between these languages
- Introduce participants to the typical OO development life-cycle

### • Outcomes:

After completing this course, participants should be able to:

- Define the core concepts of the OO paradigm
- Differentiate between the OO paradigm and other programming paradigms
- Understand the advantages and the limitations of the OO paradigm
- Analyze a problem and identify its objects and their relationships
- Explain the typical OO development process
- Compare and contrast the different OO programming languages

### • Contents:

- The Underlying Philosophy of the Object-Oriented paradigm
- Main Building Blocks of Object-Oriented Modelling
- Overview of the Fundamental Concepts of Object-Orientation
- Object Oriented Programming Languages
- Class Diagrams
- Relationship Modeling
- Known Class Relationships
- Abstraction
- Encapsulation/Information Hiding
- Inheritance
- Polymorphism
- Overview of Class Identification and Specification
- Object-Oriented Process Model

• **Duration:** Three Days

• **Pre-Requisites:** None

## C002: Unified Modeling Language (UML)

### Objectives:

- Introduce participants to the concept of software modeling.
- Introduce participants to the Unified Modeling Language (UML)
- Emphasize the benefits and limitations of the UML
- Expose participants to various UML artifacts, their symbolic notations, semantic, and typical usage

### Outcomes:

After completing this course, participants should be able to:

- Develop models using UML
- Develop class diagrams, object models, and sequence diagrams for small and medium software problems
- Understand the differences between various UML models and when they are useful for software developers
- Understand the difference between static and dynamic views of the software

### Contents:

- Review of Basic Object Orientation Concepts
- The Concepts of Model, Modeling and Modeling Language
- Introduction to the Unified Modeling Language (UML)
- UML Diagrams
- Class Diagram
- Object Diagram
- Use Case Diagram
- Sequence Diagram
- Collaboration Diagram
- State Chart Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram
- UML Relationships
- Modeling Dynamic View

**Duration:** Three Days

**Pre-Requisites:** Introduction to Object Orientation

## C003: Object-Oriented Systems Analysis and Design

### Objectives:

- Experience the insights necessary to obtain maximum benefit from object technology
  - Understand the need for, the place of, and aims of, systems analysis and design
  - Become familiar with how to use the unified modeling language (UML) in the analysis and design activities
  - Understand the models and formalisms that should be deployed
  - Understand the process by which models are designed and presented
- 
- Be ready to apply object-oriented techniques appropriately to the production of specifications and designs.

### Contents:

1. Introduction
2. The OO Process and Project
3. OO Analysis Vs. OO Design
4. Requirements Engineering
5. Requirements Elicitation and Analysis
6. **Analysis Modeling Using UML**
7. Use Case Modeling
8. System Sequence Diagram (SSD)
9. Domain (Conceptual) Model
  - Adding Association
  - Adding Attributes
10. Operation Contracts
11. **More Analysis Modeling and Models by Example**
12. Online Shopping Example
13. Activity Modeling
14. State-Chart Modeling
15. Software Architecture
16. **Object Oriented Design**
17. Overview
18. Interaction Diagram
19. Responsibility-Driven Design with the GRASP Patterns
20. Creating Design Class Diagram

**Course Duration:** Four days

**Pre-Requisites:** Unified Modeling Language (UML)

## C004: Object-Oriented Implementation

### Objectives:

This course presents the techniques and practices of transforming object-oriented design artifacts into code. The course is aimed at attendees who are familiar with at least one object-oriented programming language who wish to gain experience/ familiarization in coding of an object oriented design. The course uses C++ (and to a lesser extent Java) because of its widespread use and familiarity. We show you how to map your OO models into C++ constructs using the OO paradigm. The use of C++ or Java does not imply a special endorsement of either. C#, Smalltalk, Python, and many other object-oriented languages are amenable to object design principals and mapping to code presented in this course

Upon completion of this course, an attendee will be able to

- Identify implementation requirements in object-oriented programming languages in general.
- Identify information in UML design models necessary for implementation.
- Implement the basic building blocks of a class.
- Implement the behavior specifications developed during object-oriented design.
- Implement the dynamic behavior developed during object-oriented design.
- Implement generalization/specialization, aggregation, and other complex relationships.
- Exercise the mapping process of an object-oriented design developed using UML to an implementation through a practical case study that puts pieces together.

### Contents:

1. Introduction.
2. Implementing Classes
3. Implementing Static Behavior
4. Implementing Dynamic Behavior
5. Instantiating and Deleting Objects
6. Implementing Inheritance
7. Implementing Aggregation
8. Implementing More Relationships
9. Case Study
10. Software Testing Fundamentals

**Duration:** Three days

**Pre-Requisites:** Object-Oriented Systems Analysis and Design