

Course C001:

Introduction to Object Oriented

The aim of the course is to introduce the Object Oriented paradigm:
The Main Building Blocks of Object-Oriented Modeling, Principles Of Object-Oriented design, Class Diagrams, Relationship Modeling, Known Class Relationships,
Specifying Classes in a typical Object-Oriented Programming language (C++) while mapping abstract data types to classes.
The course discusses OO concepts, data models, levels of abstraction and some basic OO issues such as Encapsulation, Information Hiding, Message Passing, Inheritance and Polymorphism.

Course C002:

UML (Unified Modeling Language)

Unified Modeling Language (UML) is an emerging standard. UML is designed to facilitate capturing of the static structure of a system as well as its dynamic behavior. As is the case with graphical modeling languages and tools, modeling a system is based on a collection of discrete objects with their interactions to represent the system under study from different perspectives. UML is a nonproprietary, discrete, universal, general-purpose modeling language. It is not a programming language.

UML is based on a set of diagrams used to visualize a system from different perspectives. These diagrams are:

1. Class diagram
2. Object diagram
3. Use case diagram
4. Sequence diagram
5. Collaboration diagram
6. Statechart diagram
7. Activity diagram
8. Component diagram
9. Deployment diagram

The proposed UML training course will offer the trainee the following:

1. An introduction to object oriented approach
2. The concepts of models, modeling, and modeling languages
3. The Unified Modeling Language including its elements and diagrams
4. A hands-on experience for designing UML models based on one of the UML development tools.

Course C003:

O.O. Analysis & Design

This course presents the concepts and techniques of object-oriented analysis and design using UML.

The course theme views the software development process as an iterative software lifecycle, which uses UML throughout to capture and communicate analysis and design information. Upon completion of this course, an attendee will be able to

- 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

Course C004 :

O.O Implementation

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 model into the 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. The main focus of this course is to teach participants techniques and different options to mapping an object-oriented design, developed using UML, to an implementation. Participants will gain knowledge and practical hands-on to implement basic classes, static behavior, dynamic behavior, and complex relationships (for example inheritance, aggregation, and other complex relationships).

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

- Identify implementation requirements in an 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.