Unified Modeling Language (UML)

Duration	3 days (extendable to 4 days)
Audience:	Software architects, development team managers, project managers, product managers, software developers, programmers, technical writers
Pre-requisites:	Participants should be familiar with the basic principles of software engineering and have some experience with object-oriented programming
Brief Description:	The intent of the course is to provide attendees with a solid understanding of UML and how it can be used. This course covers most of the UML standard, although it places greater emphasis on those elements of UML that are more frequently used in industrial practice.

Description:

This course covers most of the UML standard, although it places greater emphasis on those elements of UML that are more frequently used in industrial practice. All topics are illustrated using practical examples, and, where time and resources permit, lab exercises are included using actual computer-based UML authoring tools (e.g., StarUML, Papyrus, IBM RSA, IBM Rhapsody).

Reduced versions of the course are also based on this outline but with more advanced units or advanced topics within specific units omitted. The latter are identified by *italics*.

The instructor was a member of the UML 1 and UML 2 core design teams.

Target Audience

Software architects, development team managers, project managers, product managers, software developers, programmers, and technical writers

Course Level

Intermediate

Course Pre-requisites

Participants should be familiar with the basic principles of software engineering and have some experience with object-oriented programming

Course Objectives

The intent of the course is to provide attendees with a solid understanding of UML and how it can be used. Because significant emphasis is placed on explaining the rationale behind the language and its constructs, attendees are likely to come away with a much deeper insight and greater ability to use UML than can be obtained from reading most popular textbooks or attending other UML courses. This applies even for slimmed down versions of the course.

Course Outline:

1. UNIT: Introduction to Model-Based Engineering (MBE) of Software and Systems

This context-setting unit introduces the basic concepts of model-based engineering and describes how these are applied to the development of modern complex software-intensive systems. The role of MBE in both traditional software development processes and more recent agile methods is discussed. Relevant technologies (languages and tools) and standards are also discussed. A brief overview of industrial experience with MBE is also provided.

2. UNIT: Introduction to UML

2.1. Lecture

This unit provides a general overview of UML and related technologies:

- What is UML?
- History and evolution of the UML standard
- How UML is used in practice
- The meaning of UML models
- General principles of the UML language
- The architecture and key components/diagrams of the UML language
- Example UML model
- *UML tools overview*

2.2. Lab

Hands-on lab that introduces a UML authoring tool of choice.

3. UNIT: Requirements modeling with UML

3.1. Lecture

This is an introduction to the use-case concept embedded within UML:

- *An overview of methods of specifying system requirements in industrial practice*
- An introduction to use cases as a means for specifying requirements
- UML use case diagrams (concepts: actor, use case, use case extension, use case inclusion)
- *Practical methods of exploiting use case diagrams (with examples)*

3.2. Lab

Using a UML authoring tool of choice to capture sample system requirements with use case diagrams

4. UNIT: Modeling Structure with UML – Part I: Class diagrams

This unit provides an in-depth introduction to the most frequently used type of UML diagram, the class diagram.

4.1. Lecture

This lecture introduces the core structure modeling concepts of UML and describes how they are represented using UML; simple illustrative examples are used throughout

- Core structural concepts: value, *data structure*, object (identity), link, attribute, operation
- UML object (instance) diagrams
- Basic class diagram concepts: class, *data type*, association, multiplicity
- How to correctly interpret UML class diagrams (and how not to interpret them)

4.2. Lab

Hands on: basic class modeling using a UML authoring tool of choice

4.3. Lecture

This lecture introduces more advanced class modeling concepts

- Composition and aggregation associations
- *N-ary associations, association classes*
- Supplementary concepts: constraint, dependency
- The concept of classification: UML generalizations/specialization (inheritance), classifier concept
- UML interfaces
- *UML flow modeling information flow, item)*

5. UNIT: Organizing UML models

This unit covers ways of organizing UML models into modules via the package mechanism

5.1. Lecture

- UML package diagrams (concepts: namespaces, package import, element import, visibility, package containment, *UML models, model libraries*)

6. UNIT: Modeling Structure with UML – Part II: Composite structure diagrams

This unit covers composite structure diagrams (collaboration, structured class), which are particularly useful for modeling technical systems and software architectures.

6.1. Lecture

This lecture covers the concept of generalized instance modeling and contrasts it with class modeling

- Limitations of class diagrams for modeling structure
- UML collaboration concepts and collaboration diagrams (role, connectors, collaborations, collaboration uses)
- *Modeling design patterns with UML collaborations*
- UML structured class modeling (port, part, internal structure)
- *Composite structure and generalization*

6.2. Lab

Hands on: basic class modeling using a UML authoring tool of choice (but ones that support composite structure modeling – not all do)

7. UNIT: Modeling Behaviour with UML – Part I: Actions and Activity diagrams

This unit first introduces the relationship between structure and behaviour in UML and the model of causality in UML (i.e., how things happen in UML). It then introduces the basic unit of behaviour: a UML action and then explains how these are combined. The recently-adopted UML action language is also briefly discussed. Finally, it explains the basic concepts of UML activity diagrams.

7.1. Lecture

- *The run-time semantics of UML (how things happen in UML)*
- Active and passive objects in UML; run-to-completion semantics of active objects
- UML actions and their composition into UML program units (action, data flow, control flow)
- *A brief introduction to the UML action language (ALF)*
- UML activities and activity diagrams (activity, structured activity, data store, swimlane)

7.2. Lab

Hands-on exercises with UML activity diagrams using simple examples

8. UNIT: Modeling Behaviour with UML – Part II: Interactions

This unit covers the three principal forms of UML interaction diagrams typically used to model interactions between multiple collaborating objects

8.1. Lecture

- Interaction diagram types
- Interaction diagram concepts (lifeline, event, event occurrence, message, execution occurrence)
- Sequence diagram essentials (interaction frame, interaction occurrence)
- *Combined fragment types and their use*
- *Timing diagrams*
- Communication diagrams
- *Interaction overview diagrams*

8.2. Lab

Hands on: basic sequence diagram exercise

9. UNIT: Modeling Behavior with UML – Part III: State machine diagrams

This unit describes how statecharts can be used to capture event-driven behaviours. The concepts are all illustrated with numerous practical examples

9.1. Lecture

- Basic state machine semantics; run-to-completion
- Basic state machine diagram concepts: states, transitions, triggers, guards, transition behaviours
- *Advanced UML statechart concepts: hierarchical states, group transitions, entry/exit/do behaviours, completion transitions, pseudostates, triggering rules*
- *Orthogonal regions modeling with UML statecharts (region, completion transition)*
- *Submachines*
- *Statechart specialization*

9.2. Lab

Hands-on exercise of statechart modeling using a UML authoring tool of choice

10. UNIT: Advanced UML Techniques – Part I: OCL

OCL is a language supported by many tools which is used to specify constraints

10.1. Lecture

- *OCL and UML class diagrams*
- *Introduction to basic first-order logic*
- *OCL Basics and OCL data types*
- *OCL collection types and collection-based constraints*
- *Using OCL in practice*

10.2. Lab

Whiteboard exercises writing OCL constraints using simple examples

11. UNIT: Advanced UML Techniques – Part II: Templates

UML templates are used to produce generic models that can be customized to fit specific situations.

11.1. Lecture

- *The concept of templates in software*
- *UML classifier templates (template, template parameter, template signature, template binding)*
- *UML package templates*

12. UNIT: Advanced UML Techniques – Part III: Profiles

UML profiles are used to customize UML to fit a specific problem or domain while retaining the ability to use general UML authoring tools.

12.1. Lecture

- *Why specialize UML and how?*
- *The UML metamodel and the Meta-Object Facility (MOF)*
- *Basic profile concepts (stereotype, profile, model library)*
- *Profile definition design patterns and methods*

12.2. Tutorial

Example of practical profile definition using (illustrated by instructor using a specific UML tool).

13. UNIT: Advanced UML Modeling – Part IV: The MARTE profile

The MARTE profile of UML complements standard UML by providing the ability to specify quantitative information such as quality of service data into UML models. This lecture-only unit is mostly intended for designers of real-time and embedded software systems, who may need to make precise predictions on the key performance indicators of their designs (e.g., timeliness, response time, schedulability, etc.).

13.1. Lecture

- *The rationale and purpose of MARTE: Software and platforms*
- *Core MARTE concepts: quality of service, resource, physical types*
- *MARTE Value Specification Language (VSL)*
- *Using MARTE to model computing platforms and applications that are Qos- and resource constrained*
- *Using MARTE for analysis*
- *Extending MARTE*

14. UNIT: An Introduction to SysML

SysML is a standardized profile of UML that is used for general systems modeling, including the modeling of various kinds of physical systems (e.g., electronics, mechanical systems, hydraulic systems, software, etc.). It is often combined with UML to specify embedded or cyber-physical systems that include software as well as hardware of various types. This lecture-only unit is only used to provide a brief (1- to 2- hour) introduction to SysML. Knowledge of UML is a pre-requisite since SysML builds on top of UML.

14.1. Lecture

- *The rationale and purpose of SysML*
- *Relationship between SysML and UML*
- *Basic SysML diagram types and formats*
- *Modeling structure with SysML: blocks, block definition diagrams, internal block diagrams, ports and flows*
- *SysML requirements diagrams*
- *SysML allocations*
- *SysML parametrics*
- *SysML and MARTE*