Chapter 4
Introduction to
Unified Modelling Language

UML Supports Application Development

Objects

Business Objects

Relationships

large scale system

ORDBMS
Oracle

Classes

application partitioning

Components
Microsoft

Scenarios

CORBA
OMG

Use Cases

ActiveX/COM
Microsoft

Business Process

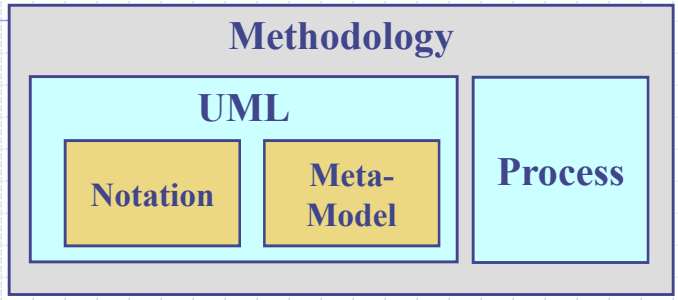
What is the UML?

The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems [OMG01].

A modeling language for modeling systems in an object-oriented way.

consists of a graphical notation and a meta-model

The UML is a Modeling Language



- UML is a modeling language, not a methodology
- The modeling language is the notation and the meta-model that methods use to express designs.
- The process is the advice on what steps to take in doing analysis and design.

UML: The Language

- Language = syntax + semantics
  - Syntax = rules by which language elements (e.g., words) are assembled into expressions (e.g., phrases, clauses)
  - Semantics = rules by which syntactic expressions are assigned meanings.
- UML Notation: defines UML's graphic syntax
- UML Semantics: defines UML's Semantics

UML: A Language for Visualizing

- Writing models in the UML facilitates communication.
- The UML is a graphical language.
- The UML is more than just a bunch of graphical symbols.

UML: A Language for Specifying

- In this context, specifying means building models that are precise, unambiguous, and complete.
- In particular, the UML addresses the specification of all the important analysis, design, and implementation decisions that must be made in developing and deploying a software-intensive system.

UML: A Language for Constructing

- It is possible to map from a model in the UML to a programming language or relational database or object-oriented database.
- Things that are best expressed graphically are done so graphically in the UML.
- This mapping permits forward engineering.
- The UML is sufficiently expressive and unambiguous.

UML: A Language for Documenting

- The UML addresses the documentation of a system's architecture and all of its details.
- The UML also provides a language for expressing requirements and for tests.
- Finally, the UML provides a language for modeling the activities of project planning and release management.

Goals of the UML

- The primary design goals of the UML are as follows:
  - Provide users with a ready-to-use, expressive visual modeling language to develop and exchange meaningful models.
  - Furnish extensibility and specialization mechanisms to extend the core concepts.
  - Support specifications that are independent of particular programming languages and development processes.
  - Provide a formal basis for understanding the modeling language.
  - Encourage the growth of the object tools market.
  - Support higher-level development concepts such as components, collaborations, frameworks and patterns.
  - Integrate best practices.

Scope of the UML

- First and foremost, the Unified Modeling Language fuses the concepts of Booch, OMT, and OOSE. The result is a single, common, and widely usable modeling language for users of these and other methods.
- Second, the Unified Modeling Language pushes the envelope of what can be done with existing methods.
- Third, the Unified Modeling Language focuses on a standard modeling language, not a standard process.

UML Provides

- The Unified Modeling Language provides the following:
  - Semantics and notation
  - Semantics to address certain expected future modeling issues, specifically related to component technology, distributed computing, frameworks, and executability.
  - Extensibility mechanisms.
  - Semantics to facilitate model interchange among a variety of tools.
  - Semantics to specify the interface to repositories for the sharing and storage of model artifacts.

UML Usage Overview

- The UML may be used to:
- Represent the **Elements** of a system or a domain and their **Relationships** in a **Static Structure** using ***class and object diagrams***
- Model the Behavior of objects with ***state transition diagrams***
- Reveal the **Physical Implementation Architecture** with ***component & deployment diagrams***
- Display the **Boundary of a System & its major Functions** using ***use cases*** and ***actors***
- Illustrate **Use Case Realizations** with ***interaction diagrams***

Features of the UML

- extensibility mechanisms (stereotypes, tagged values, and constraints),
- threads and processes,
- distribution and concurrency (e.g., for modeling ActiveX/DCOM and CORBA),
- patterns/collaborations,
- activity diagrams (for business process modeling),
- refinement (to handle relationships between levels of abstraction),
- interfaces and components, and
- a constraint language.

Outside the Scope of the UML

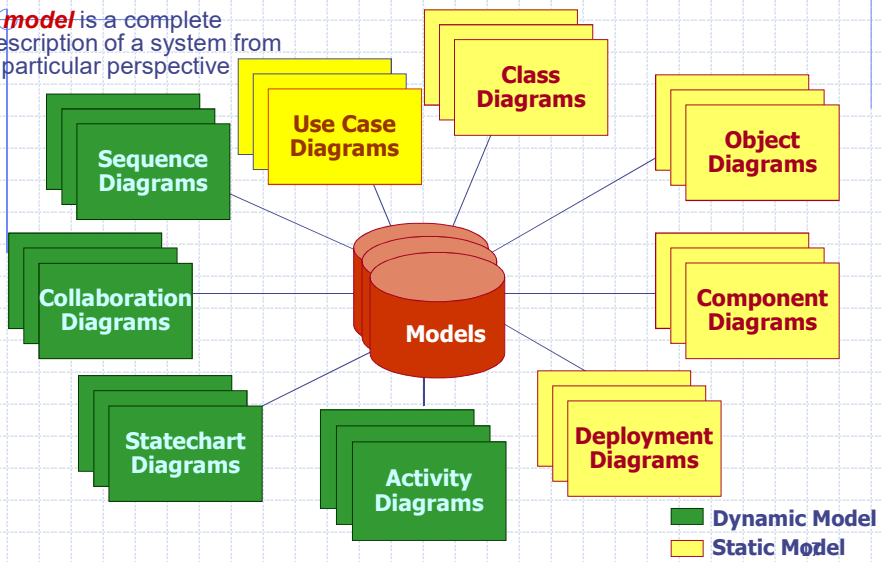
- Programming Languages
  - The UML, a visual modeling language, is not intended to be a visual programming language, in the sense of having all the necessary visual and semantic support to replace programming languages.
- Tools
  - The UML defines a semantic metamodel, not a tool interface, storage, or run-time model, although these should be fairly close to one another.
- Process
  - The UML is intentionally process independent, and defining a standard process was not a goal of the UML or OMG's RFP.

Primary Artifacts of the UML

- UML-defining Artifacts: UML Semantics, UML Notation Guide, and UML Standard Profiles
- Development Project Artifacts: In terms of the views of a model, the UML defines the following graphical diagrams:
  - use case diagram
  - class diagram
  - behavior diagrams: statechart diagram, activity diagram
  - interaction diagrams: sequence diagram, collaboration diagram
  - implementation diagrams: component diagram, deployment diagram

9 UML Diagrams

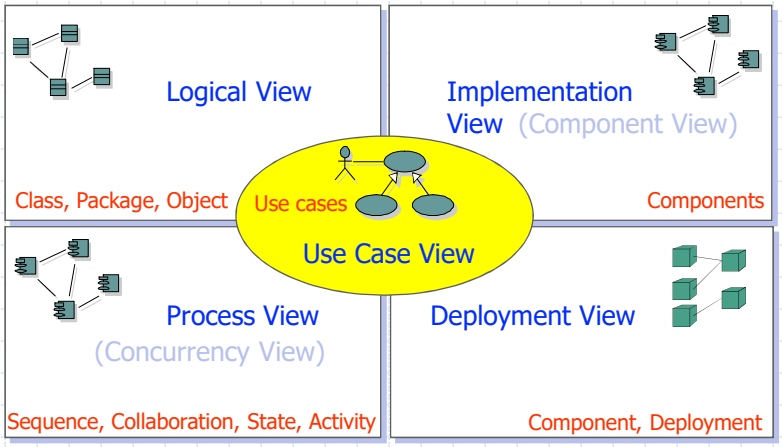
A ~~***model*** is a complete~~ description of a system from a particular perspective



UML 2.0 Diagram Types

- UML defines 13 types of diagrams, divided into two categories: six represent static application structure and seven represent different aspects of dynamic behavior.
  - Structural
    - Class diagram
    - Object diagram
    - Component diagram
    - **Composite Structure diagram**
    - Deployment diagram
    - Package diagram
  - Behavioral
    - Use Case diagram
    - State Machine diagram
    - Activity diagram
    - Interaction diagrams
      - Sequence diagram
      - **Interaction Overview diagram**
      - Communication diagram
      - **Timing diagram**

UML: 4 + 1 Architecture Views

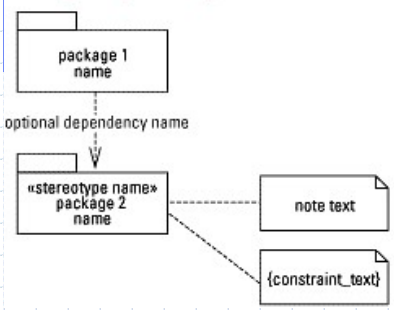


UML Quick Reference (1/9)

GENERAL-PURPOSE CONCEPTS

Can be used on various diagram types

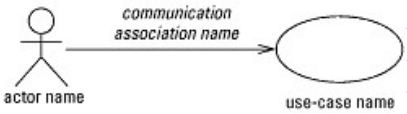
Package, dependency, note



USE-CASE DIAGRAM

Shows the system's use cases and which actors interact with them

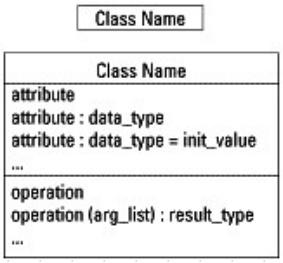
Actor, use case, and association



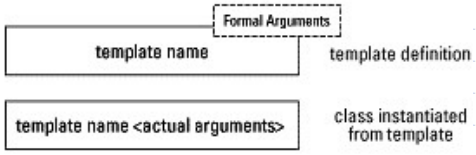
UML Quick Reference (2/9)

CLASS DIAGRAM Shows the existence of classes and their relationships in the logical view of a system

Class

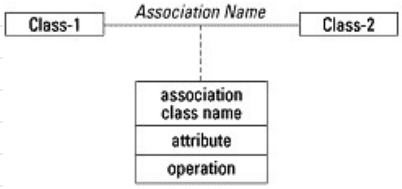


Parameterized class

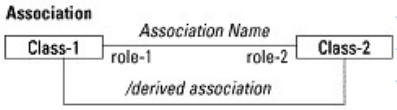


UML Quick Reference (3/9)

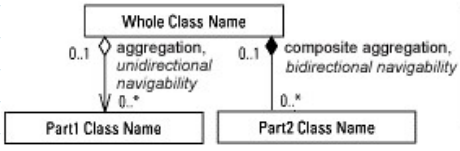
Association classes



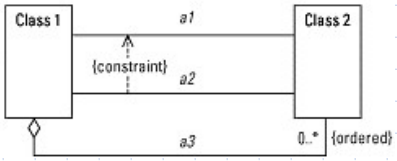
Role names and derived associations



Aggregation, navigability, and multiplicity

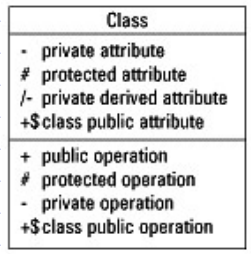


Constraints

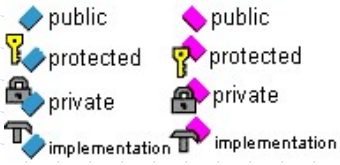


UML Quick Reference (4/9)

Visibility and properties

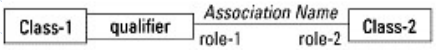


Optional visibility icons

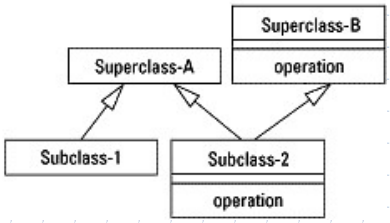


UML Quick Reference (5/9)

Qualified association



Generalization/specialization

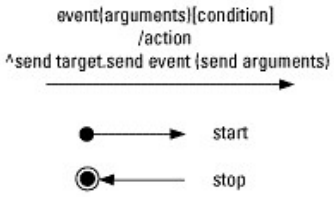


UML Quick Reference (6/9)

STATE-TRANSITION DIAGRAM Shows the state space of a given context, the events that cause a transition from one state to another, and the actions that result

State icon

State transitions

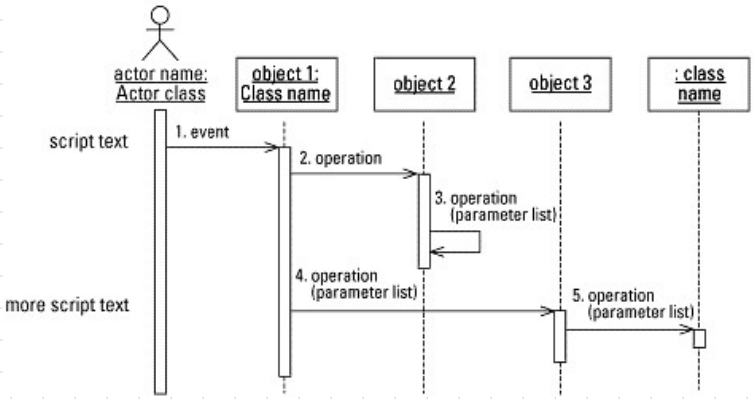


History Ⓗ

UML Quick Reference (7/9)

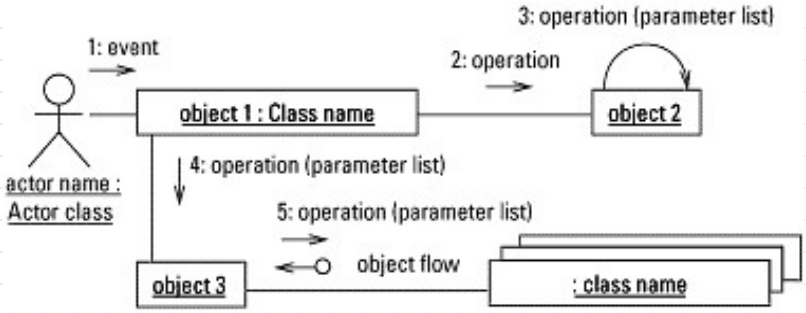
INTERACTION DIAGRAMS Show objects in the system and how they interact

Sequence diagram



UML Quick Reference (8/9)

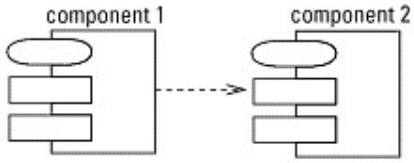
Collaboration diagram



UML Quick Reference (9/9)

COMPONENT DIAGRAM

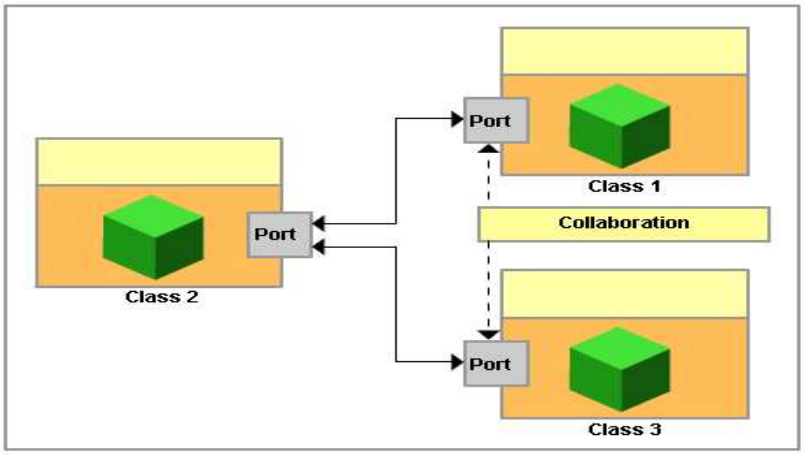
Shows the dependencies between software components



DEPLOYMENT DIAGRAM

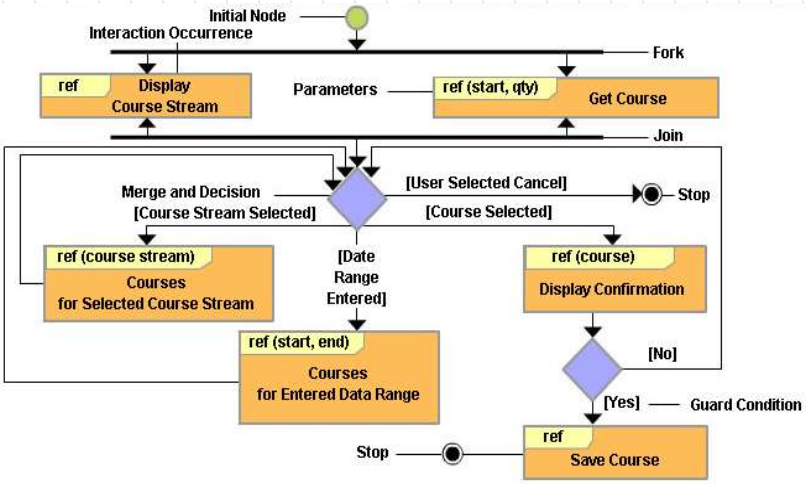
Shows the configuration of runtime processing elements

Composite Structure Diagram

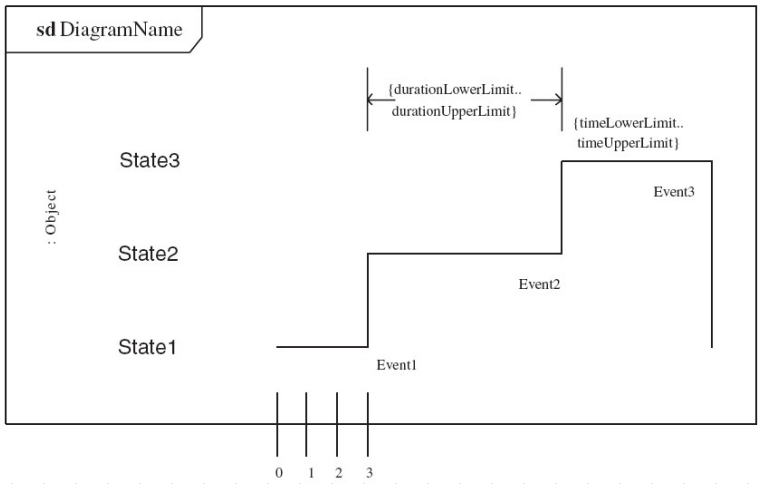


Composite Structure Diagram

Interaction Overview Diagram



Timing Diagram



Summary

- The UML is the standard language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
- The UML consists of 12 diagrams and can represent with 4+1 Views