Software System Engineering

Dr. M.E. Fayad, Professor
Computer Engineering Department, Room #283I
College of Engineering
San José State University
One Washington Square
San José, CA 95192-0180
http://www.engr.sjsu.edu/~fayad
Lesson 6:
Unified Modeling Language (UML) Overview
Lesson Objectives

- The value of the UML
- Contribution to the UML
- Building Blocks of the UML
- Diagrams in the UML
The Value of the UML

- Is an open standard
- Supports the entire software development lifecycle – Debate
- Supports diverse applications areas
- Is based on experience and needs of the user community
- Supported by many tools
Contributions to the UML

Guess who?
Overview of the UML

The UML is a language for
- visualizing
- specifying
- constructing
- documenting
the artifacts of a software-intensive system
Building Blocks of the UML

- Things
- Relationships
- Diagrams
Things in the UML

- Structural things
  - mostly static parts of a model
    - class, interface, collaboration, use case, active class, component, node
- Behavioral things
  - dynamic parts of UML models
    - interaction, state machine
- Grouping things
  - organizational parts of UM
    - package, subsytem
- Other things
  - explanatory parts of UML
    - note
Relationships (1)

- **Dependency**
  - a semantic relationship between two things in which a change to one thing (the independent thing) may affect the semantics of the other thing (the dependent thing)

- **Association**
  - a structural relationship that describes a set of links
Relationships (2)

- Generalization
  - a specialization/generalization relationship in which the child shares the structure and the behavior of the parent

- Realization
  - a realization is a relationship in which one classifier, such as an interface or a use case, specifies a "contract" that another classifier, such as a class or a collaboration, guarantees to carry out
Diagrams in UML

Statechart Diagrams
Use Case Diagrams
Class Diagrams
Object Diagrams
Component Diagrams
Sequence Diagrams
Activity Diagrams
Deployment Diagrams

Models

+
Diagrams

- A diagram is a view into a model
  - Presented from the aspect of a particular stakeholder
  - Provides a partial representation of the system
  - Is semantically consistent with other views

- In the UML, there are nine standard diagrams
  - Static views: use case, class, object, component, deployment
  - Dynamic views: sequence, collaboration, statechart, activity
Use Case Diagram

- Shows a set of use cases and actors and their relationships

```
Use scheduler

Place phone call

Place conference call

Receive phone call

Receive additional call

Cellular telephone

Actor

Cellular network

User

association

extend

extend

use case

system boundary
```
Use Case Diagram

- Captures system functionality as seen by users
- Built in early stages of development
- Purpose
  - Specify the context of a system
  - Capture the requirements of a system
  - Validate a system’s architecture
  - Drive implementation and generate test cases
- Developed by analysts and domain experts
Class Diagram (1)

- Shows a set of classes, interfaces, and collaborations and their relationships
Class Diagram (2)

- Captures the vocabulary of a system
- Addresses the static design view of a system
- Built and refined throughout development

Purpose
- Name and model concepts in the system
- Specify collaborations
- Specify logical database schemas

Developed by analysts, designers, and implementers
Object Diagram (1)

- Shows a set of objects and their relationships
Object Diagram (2)

- Represents static snapshots of instances of the things found in class diagrams
- Addresses the static design view or static process view of a system
- Built during analysis and design
- Purpose
  - Illustrate data/object structures
  - Specify snapshots
- Developed by analysts, designers, and implementers
Component Diagram (1)

- Shows the organizations and dependencies among a set of components
Component Diagram (2)

- Addresses the static implementation view of a system
- Built as part of architectural specification
- Purpose
  - Organize source code
  - Construct an executable release
  - Specify a physical database
- Developed by architects and programmers
Deployment Diagram (1)

- Shows the configuration of run-time processing nodes and the components that live on them
Deployment Diagram (2)

- Captures the topology of a system’s hardware
- Built as part of architectural specification
- Purpose
  - Specify the distribution of components
  - Identify performance bottlenecks
- Developed by architects, networking engineers, and system engineers
Activity Diagram (1)

- Shows the flow from activity to activity within a system
Activity Diagram (2)

- Captures dynamic behavior (activity-oriented)
- A special kind of statechart diagram
- Purpose
  - Model the function of a system
  - Model the flow of control among objects
  - Model business workflows
  - Model operations
Sequence Diagram (1)

- Emphasizes the time-ordering of messages

![Sequence Diagram](image)
Sequence Diagram (2)

- Captures dynamic behavior (time-oriented)
- A kind of interaction diagram
- Purpose
  - Model flow of control
  - Illustrate typical scenarios
Collaboration Diagram (1)

- Emphasizes the structural organization of the objects that send and receive messages

```
start c : Client
\[\text{link}\]
\[\text{message}\]
\[\text{«local»}\]
\[\text{transaction}\]
\[\text{object}\]
\[\text{«global»}\]
\[\text{2.1 : setValues(d, 3.4)}\]
\[\text{2.2 : setValues(a, "CO")}\]
```

```
\[\text{1 : «create»}\]
\[\text{2 : setActions(a, d, o)}\]
\[\text{3 : «destroy»}\]
```
Collaboration Diagram (1)

- Captures dynamic behavior (message-oriented)
- A kind of interaction diagram
- Purpose
  - Model flow of control
  - Illustrate coordination of object structure and control
Statechart Diagram (1)

- Shows a state machine, consisting of states, transitions, events, and activities
Statechart Diagram (2)

- Captures dynamic behavior (event-oriented)
- Purpose
  - Model object lifecycle
  - Model reactive objects (user interfaces, devices, etc.)
Architecture and the UML

Design View
- Classes, interfaces, collaborations

Use Case View
- Use cases

Implementation View
- Components

Process View
- Active classes

Deployment View
- Nodes

Organization
- Package, subsystem

Dynamics
- Interaction
- State machine
UML Software Development Life Cycle (1)

- **Use-case driven**
  - use cases are used as a primary artifact for establishing the desired behavior of the system, for verifying and validating the system’s architecture, for testing, and for communicating among the stakeholders of the project

- **Architecture-centric**
  - a system’s architecture is used as a primary artifact for conceptualizing, constructing, managing, and evolving the system under development
UML Software Development Life Cycle (2)

- **Iterative**
  - one that involves managing a stream of executable releases

- **Incremental**
  - one that involves the continuous integration of the system’s architecture to produce these releases
Lifecycle Phases

- Inception
  - Define the scope of the project and develop business case

- Elaboration
  - Plan project, specify features, and baseline the architecture

- Construction
  - Build the product

- Transition
  - Transition the product to its users
Discussion Questions

T/F

- UML is a model for developing software.
- A use case diagram shows a set of use cases, actors, and their relationships.
- Use case diagrams are dynamic models.
- An interaction diagram shows the flow of activities within the system.
- Sequence diagrams emphasizes the time-ordering of the messages.
Questions for the Next Lecture

- Use Case Models
- Use Case Diagrams
- Use Case Templates
- Use Cases