Introduction to OOAD and the UML

Instructor: Dr. Hany H. Ammar
Dept. of Computer Science and Electrical Engineering, WVU
OUTLINE

- The development process
- Reviewing Object Oriented Analysis and Design
- Visual modeling and the Unified Modeling Language UML
OUTLINE

- The development process
  - Phases of system development
  - The Unified Process

- Object Oriented Analysis and Design

- Visual Modeling and the Unified Modeling Language UML
The Development Process

New or changed requirements → Software Engineering Process → New or changed system
Phases of System Development

- **Requirements**: Develop the Requirements Model
- **Analysis**: Develop the Logical Model
- **Design**: Develop the Architecture Model
- **Implementation**
- **Testing**
The IEEE 12207 Development Process

One example of applying 12207 to the Waterfall development strategy

Process Implementation Activity

DPP, SDSD

Software Item 1:
- Software Reqs. Analysis
- Software Arch. Design
- Software Detailed Design
- Software Code & Test
- Software Integration
- Software Qualification

Software Item 2:
- Software Reqs. Analysis
- Software Arch. Design
- Software Detailed Design
- Software Code & Test
- Software Integration
- Software Qualification

Hardware items

Supporting Processes: Documentation, CM, QA, Verification, Validation, Joint Review, Audit, Problem resolution

SCMP, SCMR, SCIR, SQAP, SQAR, SVRR, PR/PRR

Organizational Processes: Management, Infrastructure, Improvement, Training
The Unified Process
(The Rational Unified Process (RUP), adopted by IBM for system development)

- Supports System Development Using the Unified Model Language (UML)
- Evolutionary process where the system is built iteratively and incrementally in several builds starting from the requirements phase
- Architecture-centric
The Unified Process

**Inception**: Define the scope of the system (identify all external entities with which the system will interact and define the nature of the interactions)

**Elaboration**: Specify features and develop the architecture

**Construction**: Build the system

**Transition**: Transition Product to its users
An *iteration* is a sequence of activities with an established plan and evaluation criteria, resulting in an executable release.
The Unified Process
The UP develops the architecture iteratively in successive Refinements during the Elaboration phase.
The development process

- Reviewing Object Oriented Analysis and Design
  - Object-Oriented Analysis OOA
  - Object-Oriented Design OOD

- Visual Modeling and the Unified Modeling Language UML
Object Oriented Analysis and Design (OOAD)
Review of OOAD Basic Concepts

- Develops a system model using a set of interacting objects

- A Class:
  - A class is a description used to instantiate objects

- An Object:
  - Is an instance of a class, it has a name, attributes and their values, and methods
  - An object models an idea found in reality, (tangible or abstract)
Basic Concepts (cont’d)

- Attributes of a class
- Methods of a class (Services, Actions, Messages)
- Information hiding and Encapsulation: A technique in which an object reveals as little as possible about its inner workings (Private and Public methods or attributes).
- Inheritance defines a class hierarchy based on abstraction
OUTLINE

- The development process
- Reviewing Object Oriented Analysis and Design
  - Object-Oriented Analysis OOA
  - Object-Oriented Design OOD
- Visual Modeling and the Unified Modeling Language UML
Object Oriented Analysis

OOA Develops a Logical Model of the system as a set of interacting domain objects

- The model consists of two views
  - The *static* view: defines the classes and their dependencies
  - The *dynamic* view: models the scenarios of interactions between objects

Class A

Class B

Requires Service From Class B

Obj s:
Class A

Obj s:
Class B

Set_Alarm(message)
Example:
The Static Analysis Model
Class diagram

The dynamic Model: A Scenario Of Interactions
OOA (cont.)

OOA starts by identifying domain objects from the requirements model (Use-Case Models)

1. Discovering Objects
   - The Data Perspective
     - In the problem space or external systems
     - Physical devices (sensors, actuators)
     - Events that need to be recorded (ex. Measurements)
     - Physical or geographical locations
The Functional Perspective
- What responsibilities does the object have? Ex. An event handler, a controller, monitors, sensors, etc.

The Behavioral Perspective
- Who does the object interact with? How?
- Use a State Transition Diagrams to describe the object behavior
OOA (cont’d): Identifying Domain Objects from the requirements model

In the statements of the requirements:

- An object may appear as a noun (ex. Measurement) or disguised in a verb (to measure)
- A method might appear as a verb (ex. Investigate) or disguised in a noun (investigation)
- Attributes describe some kind of characteristics for the object (adjectives). Attributes can be simple or complex. Complex attributes may lead to forming new objects. Attributes can also be nouns.
OOA (cont’d): *Object Types*

- External Entities and their interfaces: Sensors, actuators, control panel, devices, operators, pilots
- Information Items: Displays, Commands, Requests, etc.
- Entities which establishes the context of the problem: Controller, monitors, schedulers
2. Define Class Hierarchies

- Generalization
  - Display → Login Display

- Specialization (IS_A)
  - Temperature_Sensor → Sensor

```
Sensor

Brake Sensor

Engine Sensor
```
3. Class Relationships

- Types
  - Association
    - General form of dependency
  - Aggregation
    - An object may consist of other objects
  - Inheritance

- Cardinality (Multiplicity)
  - (Binary, Many, ..)
Example of identifying Class diagrams with Relationships, Multiplicities, Attributes, and operations (E-Commerce)
4. Object Attributes

- Discovering attributes of classes
- Attribute types
  - Naming: Ex. SensorID, Account
  - Descriptive Ex. Card expiration date
  - Referential Ex. Referring to other objects
5. The Dynamic View: Object Behavior

- Discovering states, transitions between states, and conditions and actions
- Building the state diagrams of objects

![State Diagram](image)
6. Object Services

- Implicit Services (create, modify, search, delete, etc.) ex. constructors
- Services associated with messages
- Services associated with object relationships
- Services associated with attributes (accessor methods ex. get, set ... )
OUTLINE

- The development process
- Reviewing Object Oriented Analysis and Design
  - Object-Oriented Analysis OOA
  - Object-Oriented Design OOD
- Visual Modeling and the Unified Modeling Language UML
Object Oriented Design OOD

1. Architecture Design
   - The static view: structural description (defining the components and subsystems)
   - The Dynamic view (defining the interactions between components and subsystems)

2. Detailed Design: Define detailed Class and object description
   - Visibility (Private, protected, .. )
   - Containment (ex. Packages or Components)
   - Concurrency
OOD: Architecture Design

- Define the subsystems/components and their dependencies
- Interactions between components are defined in design sequence diagrams

**Figure 3.3: Subsystems in the sound recorder**
OOD: Detailed Design

Define the detailed design of each subsystem/component

Figure 3.4: Audio subsystem class diagram
Define design sequence diagrams for scenarios defined in the requirements model.

Figure 3.6: Play message sequence diagram
Design Refinement: Enhance Design Goodness Criteria (e.g., using design patterns)

- Coupling:
  - The manner and degree of interdependence between classes (objects)

- Cohesion:
  - The degree and manner to which the services or tasks performed by a component or an object are related to each other.

- Modularity
  - Understandability
  - Decomposability

- Clarity
  - Simple classes, messages, methods
Summary of the Object-Oriented Analysis and Design (OOA) Methodology

- Based on describing the logical model of the system and the environment as a set of interacting objects
- Defines the external objects (actors) interacting with the system as well as the internal objects that the system must contain
- Defines the static architecture of objects and the dynamic behavioral interactions between them
- Defines the internal dynamic behavior of objects
OUTLINE

- The development process

- Reviewing Object Oriented Analysis and Design

- Introducing visual modeling and the Unified Modeling Language UML
Visual Modeling and the Unified Modeling Language UML

- What is the UML?
- UML Concepts
- UML Development - Overview
What is the UML?

- UML stands for Unified Modeling Language
- The UML is the standard language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
- It can be used with all processes, throughout the development life cycle, and across different implementation technologies.
The UML may be used to:

- Develop a Requirements Model
  1. Use Case diagrams - Define the scope, and display the boundary of a system & its major functions using *use cases* and *actors*
  2. System Sequence diagrams - Illustrate use case realizations or scenarios of interactions between the actors and the system

- Develop the Analysis model
  1. Class diagrams - Represent a static structure of a system
  2. State Charts - Model the behavior of objects
UML Concepts

- Develop the architecture design model
  1. Class diagrams: Represent the static architecture using packages or subsystems
  2. Design Sequence diagrams – Represent the dynamic interactions between the design objects

- Develop the physical architecture implementation model
  - component & deployment diagrams - Reveal the physical implementation architecture
Visual Modeling and the Unified Modeling Language UML

- What is the UML?
- UML Concepts
- UML Development - Overview
UML Development - Overview

Requirements Engineering

- Requirements Elicitation
- Analysis
  - Specify Domain Objects
- Architectural Design
  - Include Design Objects
- Detailed Design
  - Object Design

Implementation

- Analysis Class Diagram(s)
- Scenarios
- Actors
- Use Cases

Time

- System/Object Sequence Diagrams
- Statechart Diagrams
- Operation Contracts
- Subsystem Class or Component Diagrams
- Deployment Diagram
- Design Sequence Diagram
- Design Diagrams
- Implementation Choices
- Implementation Activity Diagrams
- Program
A Model of embedded systems development

Figure 9.1: Embedded systems design class diagram
Visual Modeling and the Unified Modeling Language (UML)

- What is the UML?
- UML Concepts
- UML Development – Overview
- Requirements Engineering
- Requirements Elicitation
UML Development - Overview

Requirements Engineering

ANALYSIS
Specify Domain Objects

ACTORS

SCENARIOS

ANALYSIS CLASS DIAGRAM(S)

USE CASES

StateChart DIAGRAMS

OPERATION CONTRACTS

System/Object SEQUENCE DIAGRAMS

DEPLOYMENT DIAGRAM

DESIGN SEQUENCE DIAG.

DESIGN DIAGRAMS

IMPLEMENTATION

IMPLEMENTATION CHOICES

IMPLEMENTATION Activity DIAGRAMS

PROGRAM

ARCHITECTURAL Design
Include Design Objects

Detailed DESIGN
Object Design

DATABASE DICTIONARY
UML Use Case Diagrams: The Requirements Model

Defining Actors (External objects)
- An actor is an object that must interact with the system under development
Defining Use Cases

- A use case captures the user requirements, it is a pattern of behavior the system exhibits
  - Each use case is a sequence of related interactions performed by an actor and the system in a dialogue

- Actors are examined to determine their needs
  - Each actor must have association with at least one use case
  - Use cases can be related to each other
UML Use Case Diagrams: The Requirements Model

Figure 2.3: Use Case diagram
Documenting Use Cases

- A flow of events document is created for each use case.
  - Written from an actor's point of view.
- Details what the system must provide to the actor when the use case is executed.
- Typical contents:
  - How the use case starts and ends.
  - Normal flow of events.
  - Alternate flow of events.
  - Exceptional flow of events.
The use case diagram presents an outside view of the system.

Interaction diagrams capture the scenarios of the functional requirements.

They describe how use cases are realized as interactions among societies of objects (objects interact to accomplish a function of the system).

UML supports two types of interaction diagrams: Sequence diagrams, and Collaboration diagrams.
A sequence diagram displays object interactions arranged in a time sequence capturing a specific *scenario* of interactions in a *use case* supported by the system.

*Figure 2.4: Playing message scenario*
Visual Modeling and the Unified Modeling Language UML

- What is the UML?
- UML Concepts
- UML Development – Overview
  - Requirements Engineering
  - Requirements Elicitation
- The Analysis Model
UML Development - Overview

Requirements Engineering

ANALYSIS
Specify Domain Objects

ACTORS

SCENARIOS

USE CASES

SYSTEM/OBJECT SEQUENCE DIAGRAMS

ANALYSIS CLASS DIAGRAM(S)

StateChart DIAGRAMS

OPERATION CONTRACTS

ARCHITECTURAL DESIGN
Include Design Objects

SUBSYSTEM CLASS OR COMPONENT DIAGRAMS

DEPLOYMENT DIAGRAM

DESIGN SEQUENCE DIAG.

DESIGN DIAGRAMS

DETAILED DESIGN
Object Design

IMPLEMENTATION CHOICES

IMPLEMENTATION Activity DIAGRAMS

PROGRAM
A class diagram shows the existence of classes and their relationships in the logical view of a system.

- UML modeling elements in class diagrams
  1. Classes and their structure and behavior
  2. Association, aggregation, and inheritance relationships
  3. Multiplicity and navigation indicators
  4. Role names
UML Class Diagrams: The Analysis Model

Define Classes, Relationships, Multiplicities, Attributes, and operations
UML Class Diagrams:
The Analysis Model
Digital Sound Recorder Case Study

Figure 3.2: Sound Recorder class diagram
UML State charts:
The Analysis Model

The State of an Object

- A state transition diagram shows
  - The life history of a given class
  - The events that cause a transition from one state to another
  - The actions that result from a state change

- State transition diagrams are created for objects with significant dynamic behavior
UML State charts: The Analysis Model

Figure 10.19 Hierarchical Cruise Control statechart with activities and exit action
UML State charts:
The Analysis Model
Digital Sound Recorder Case Study

**Figure 4.3:** AudioOutput statechart

**Figure 4.4:** Microphone statechart
Visual Modeling and the Unified Modeling Language UML

- What is the UML?
- UML Concepts
- UML Development – Overview
  - Requirements Engineering
  - Requirements Elicitation
  - The Analysis Model
- The Design Model
UML Development - Overview

**Requirements Engineering**

- **ANALYSIS**
  - Specify Domain Objects

**Architectural Design**

- Include Design Objects
- **Detailed DESIGN**
  - Object Design

**IMPLEMENTATION**

- **ACTORS**
- **SCENARIOS**
- **ANALYSIS CLASS DIAGRAM(S)**
- **USE CASES**
- **StateChart DIAGRAMS**
- **OPERATION CONTRACTS**
- **SUBSYSTEMS / CLASS/ OR COMPONENT DIAGRAMS**
- **DEPLOYMENT DIAGRAM**
- **DESIGN SEQUENCE/ or COLLABORATION DIAGRAMS**
- **DESIGN DIAGRAMS**
- **IMPLEMENTATION CHOICES**
- **IMPLEMENTATION Activity DIAGRAMS**
- **PROGRAM**
Object Oriented Design (OOD)

1. Architecture Design (Subsystem/Component Diagrams)
   - The static view: structural description (defining the components and subsystems)
   - The Dynamic view (defining the interactions between components and subsystems)

2. Detailed Design: Define detailed Class and object description
   - Visibility (Private, protected, .. )
   - Containment (ex. Packages or Components)
   - Concurrency
UML Class Diagrams:
Architecture Design: The static view
Digital Sound Recorder Case Study

- Define the subsystems/components and their dependencies
- Interactions between components are defined in design sequence diagrams

Figure 3.3: Subsystems in the sound recorder
UML Class Diagrams:
Detailed Design: The static view
Digital Sound Recorder Case Study

Define the detailed design of each subsystem/component

Figure 3.4: Audio subsystem class diagram
Recall: OOA (cont.)

Satellite Control Example

Example:
The Static Analysis Model
Class diagram

The dynamic Model:
A Scenario Of Interactions
UML Development – Overview

Detailed Design: The dynamic view, Design Sequence Diagrams

Requirements Engineering

Requirements Engineering

ANALYSIS
Specify Domain Objects

ANALYSIS CLASS DIAGRAM(S)

SYSTEM/OBJECT SEQUENCE /COllABORATION DIAGRAMS

DESIGN SEQUENCE /or COLLABORATION DIAGRAMS

ARCHITECTURAL DESIGN
Include Design Objects

SUBSYSTEM CLASS/ OR COMPONENT DIAGRAMS

DEPLOYMENT DIAGRAM

DESIGN DIAGRAMS

OBJECT DESIGN
Implementation ChOICES

IMPLEMENTATION

ACTORS

USE CASES

TIME

SCENARIOS

StateChart DIAGRAMs

OPERATION CONTRACTS

IMPLEMENTATION

PROGRAM
Define design sequence diagrams for scenarios defined in the requirements model.
Detailed Design: The dynamic view, UML Collaboration Diagrams

This diagram has similar information as in sequence diagrams with no time axis.

Digital Sound Recorder Case Study
UML Component and Deployment Diagrams

- Component diagrams illustrate the organizations and dependencies among software components.

- A component may be:
  - A source code component
  - A run time components or
  - An executable component
UML Development – Overview

Detailed Design: The dynamic view, Design Sequence Diagrams

- **ACTORS**
- **USE CASES**
- **SCENARIOS**
- **ANALYSIS CLASS DIAGRAM(S)**
- **StateChart DIAGRAMS**
- **OPERATION CONTRACTS**
- **SUBSYSTEM CLASS OR COMPONENT DIAGRAMS**
- **DEPLOYMENT DIAGRAM**
- **DESIGN SEQUENCE /or COLLABORATION DIAGRAMS**
- **DESIGN DIAGRAMS**
- **IMPLEMENTATION CHOICES**
- **IMPLEMENTATION Activity DIAGRAMS**
- **PROGRAM**
Course registration
System example

Component Diagram
The interfaces to a component may be shown on a component diagram.
UML Development – Overview

Detailed Design: The dynamic view, Design Sequence Diagrams

Requirements Engineering

ANALYSIS
Specify Domain Objects

ARCHITECTURAL DESIGN
Include Design Objects

DETAILED DESIGN
Object Design

IMPLEMENTATION

- ACTORS
- USE CASES
- SCENARIOS
- ANALYSIS CLASS DIAGRAM(S)
- StateChart DIAGRAMs
- OPERATION CONTRACTS
- SUBSYSTEM CLASS OR COMPONENT DIAGRAMS
- DEPLOYMENT DIAGRAM
- DESIGN SEQUENCE /or COLLABORATION DIAGRAMS
- DESIGN DIAGRAMS
- IMPLEMENTATION CHOICES
- IMPLEMENTATION Activity DIAGRAMS
- PROGRAM
Deploying the System

- The deployment diagram shows the configuration of run-time processing elements and the software processes living on them.
- The deployment diagram visualizes the distribution of components across the enterprise (the servers of a distributed network).
Deployment Diagram

Defines the HW Network configuration
Figure 5.1: Hardware architecture of the digital sound recorder
Extending the UML

- Stereotypes can be used to extend the UML notational elements.
- Stereotypes may be used to classify and extend associations, inheritance relationships, classes, and components using the notation <<stereotype>>.
- Examples: 1. Class stereotypes: Interface, control, entity, utility, exception,
  - 2. Use Case relation stereotypes: includes and extends,
  - 3. Component stereotypes: subsystem
  - 4. Design pattern instances
Class and Components stereotypes
e.g., <<external timer>>, <<coordinator>>, <<control>>
Use Case relation stereotypes

<<extend>>
Component stereotypes: subsystem

<<client subsystem>>, <<server subsystem>>
Summary of UML


- The UML is the standard language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
  - It can be used with all processes, throughout the development life cycle, and across different implementation technologies.