

# Chapter 1: Object-Oriented Analysis and Design

1) What is OOAD? .....	1-3
2) Approaches to System Analysis .....	1-4
3) Object-Oriented Methodologies .....	1-5
4) History of UML .....	1-7
5) What is UML? .....	1-9
6) Models and Architectural Views .....	1-10
7) Common Features of UML Diagrams.....	1-11
8) Characteristics of a UML Process .....	1-13

**training/etc**  
*The Art of Knowledge.*

Evaluation  
Copy

## Exercise

### 1. Class Discussion:

- ▶ What is analysis?
- ▶ What is design?
- ▶ What is decomposition?
- ▶ What kinds of techniques and drawings have you used in the past for analysis and design?



Evaluation  
Copy

## What is OOAD?

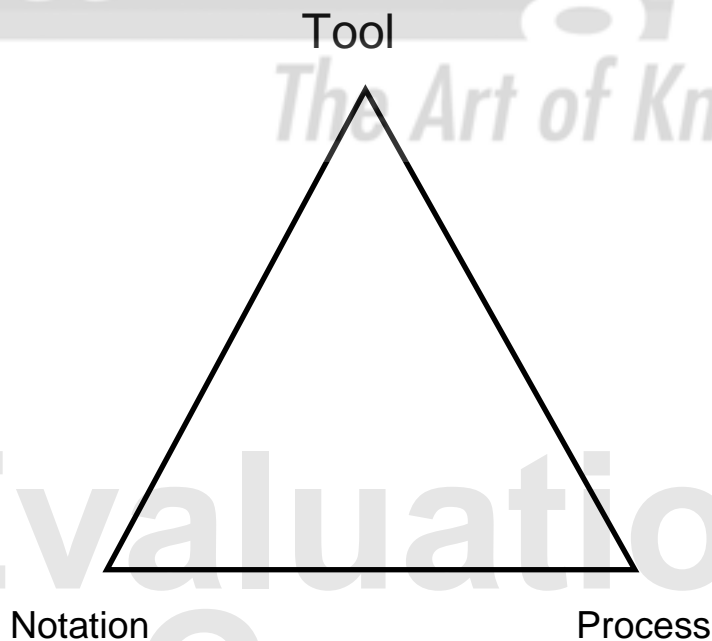
- **Object-Oriented Analysis (OOA)**, like other kinds of systems analysis, clarifies and documents the requirements of a computer system.
- OOA focuses on studying and understanding the problem first, initially ignoring the concerns of an actual implementation. Many terms are used to describe the key real-world concepts of an application:
  - ▶ Problem domain
  - ▶ Application domain
  - ▶ Business objects
  - ▶ Domain objects
  - ▶ Problem essence
  - ▶ Key classes
- The chief OOA activity is discovering and documenting the key classes for a particular problem domain.
- **Object-Oriented Design (OOD)** is viewed as an extension of analysis, more so than a distinct activity. OOD adds the detailed design for each class:
  - ▶ Names, data types, and access specifiers for attributes
  - ▶ Names, return types, and parameter lists for all methods
  - ▶ Associations and collaborations with other classes

## Approaches to System Analysis

- Functional Decomposition
  - ▶ Maps problem domain to function and sub-functions.
  - ▶ Processing steps are used as primary organizational framework during analysis.
  - ▶ Domain understanding is neither explicitly expressed nor verified for its accuracy.
  
- Data Flow Diagrams
  - ▶ Maps the real world into data flows.
  - ▶ Data flows are an unnatural, computer-related method of organization.
  
- Information Modeling
  - ▶ Models the world in data (objects + attributes + relationships).
  - ▶ Closest to the object-oriented approach.
  - ▶ Helps capture problem domain content, but only partially.
  
- Object-Oriented Analysis
  - ▶ Directly maps problem domain and system responsibility into a model.
  - ▶ Classes and Objects + Inheritance + Communication with Messages
  - ▶ Same underlying representation is used in analysis and design.

## Object-Oriented Methodologies

- Many methodologies have been proposed for object-oriented software development. A **methodology** usually includes:
  - ▶ Notation – graphical representations of classes and their relationships and interactions
  - ▶ Process – suggested set of steps to carry out for transforming requirements into a working system
  - ▶ Tool – software for drawings and documentation



## Object-Oriented Methodologies

- Some popular methodologies include:
  - ▶ Object Modeling Technique (OMT)
    - James Rumbaugh
  - ▶ The Booch Method
    - Grady Booch
  - ▶ Object-Oriented Software Engineering (OOSE)
    - Ivar Jacobson
  - ▶ Object-Oriented Analysis (OOA)
    - Peter Coad
    - Ed Yourdon
  - ▶ CRC (Classes-Responsibilities-Collaborations) Method
    - Rebecca Wirfs-Brock
  
- The number of OO methodologies increased from less than 10 to more than 50 between 1989 and 1994. Methodologists battled for market share (the "method wars"), while many developers had trouble finding complete satisfaction with any single method.

## History of UML

- In October 1994, Rumbaugh joined Booch's company (Rational Software Corporation) to unify the Booch and OMT methods. A draft was released in October 1995, then called the Unified Method.
- In 1995 Jacobson also joined the unification effort, merging in the OOSE method. Preliminary documents for the **Unified Modeling Language (UML)** were released during 1996.
- The UML Version 1.0 definition was finalized in January 1997, including the work of the UML Partners consortium.

DEC	MCI Systemhouse
HP	Microsoft
i-Logix	Oracle
Intellicorp	Rational Software
IBM	TI
ICON Computing	Unisys

- UML 1.1 was adopted by the Object Management Group (OMG) in November 1997. OMG is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications.

## History of UML

- IBM acquired Rational Software Company in 2002.
- Revisions to the UML specification through Version 1.5 were released in March 2003.
- OMG is currently upgrading UML to Version 2.0. Adopted in late 2003 and posted on OMG's website labeled "UML 2.0 Final Adopted Specification," the upgraded version won't replace UML 1.5 as the official "Available UML Specification" until it completes its initial maintenance revision sometime around the end of 2004.

<http://www.omg.org>

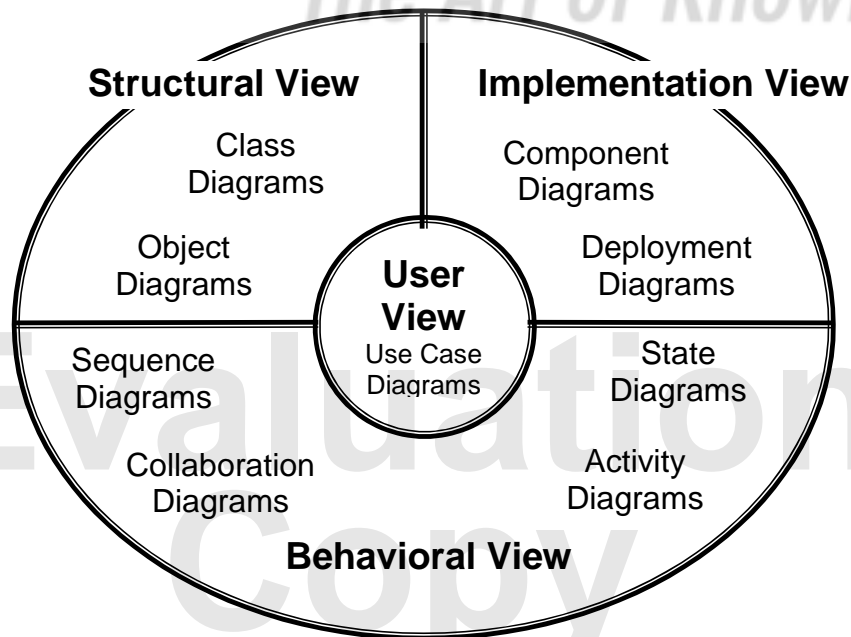
Evaluation  
Copy

## What is UML?

- UML focuses on a standard modeling language, not a standard process.
- UML fuses the concepts of Booch, Rumbaugh, and Jacobson.
- UML is nonproprietary.
  - ▶ UML Version 1.0 was submitted to the **Object Management Group** (OMG) Analysis and Design Task Force for consideration as the industry standard.
  - ▶ In November 1997, UML was adopted into the Object Management Architecture (OMA).
- This course introduces UML as a standard graphical notation for object-oriented software design, and as a vehicle for presenting important analysis and design concepts. The following UML features will be studied:
  - ▶ Use Case Diagrams
  - ▶ Class Diagrams
  - ▶ Sequence Diagrams
  - ▶ Collaboration Diagrams
  - ▶ State Diagrams
  - ▶ Activity Diagrams
  - ▶ Component Diagrams
  - ▶ Deployment Diagrams

## Models and Architectural Views

- **Models** are blueprints of systems used for system construction and renovation.
- Models are used to understand and manage complexity within systems.
- **Architectural views** map models to types of diagrams. Different types of architectural views include:
  - ▶ User view
  - ▶ Structural view
  - ▶ Behavioral view
  - ▶ Implementation view



## Common Features of UML Diagrams

- UML diagrams are two-dimensional drawings. Most UML diagrams are graphs containing nodes connected by paths.
- There are four basic diagram elements:
  - ▶ Shapes - two-dimensional symbols of variable width and height that can expand to hold other symbols or strings
  - ▶ Paths - sequences of line segments with optional terminators
  - ▶ Icons - graphical figures (that do not hold contents)
  - ▶ Strings
- Class vs. object (instance) notation

**Class-name**

represents  
the class

Person

**: Class-name**

represents *any*  
object of the class

: Person

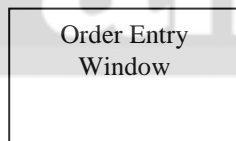
**Object-name : Class-name**

represents a *specific*  
object of the class

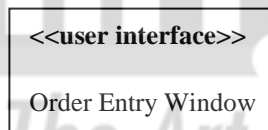
Alan : Person

## Common Features of UML Diagrams

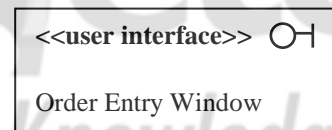
- UML can be extended through the use of **stereotypes**.
  - ▶ A stereotype is an adaptation of an existing element, with some extra semantics added.
  - ▶ A number of stereotypes are predefined.
    - These stereotypes "adjust" existing elements instead of defining new ones; this helps keep UML simple.
  - ▶ A stereotype is denoted by a string enclosed in guillemets.
    - A stereotype can also have a graphical representation.
  - ▶ Example:



Class symbol



Class symbol  
with stereotype



Class symbol  
with stereotype  
and associated icon

- A **note** can be added to any diagram.
  - ▶ The symbol for a note is a rectangle with a "bent corner" in the upper right corner.



## Characteristics of a UML Process

- UML provides a consistent language for specifying, visualizing, and documenting object-oriented software. It does not prescribe a process or method for working with the language.
  - ▶ Rational Software offers the Rational Unified Process as their methodology.
  
- Basic characteristics of any UML process should include:
  - ▶ Iterative
    - Instead of trying to define all the details of a model at once, several "passes" are made.
    - Each iteration adds more detail.
  - ▶ Incremental
    - The system evolves through a set of increments.
    - Each increment adds more functionality.
  - ▶ User-centric
    - Analysts specify functionality with use cases.
    - Customers confirm use cases.
    - Designers and implementers realize use cases.
    - Testers verify the system with use cases.
  - ▶ Well-defined architecture
    - The system is partitioned into subsystems.
    - Logical and physical views of the system are separated.

## Ductwork Estimating System

The Ductwork Estimating System (DES) allows the user to enter dimensions of sheet metal air ducts and then computes the amount and cost of the required materials and accessories. There are three major steps involved in using DES:

- (1) The user inputs all of the dimensions for the pieces of ductwork required for a particular project.
- (2) The computer calculates the total surface area of metal required and total number of accessories needed.
- (3) The computer applies unit cost factors and determines total cost of materials.

There are three types of ductwork handled by DES: square, rectangular, and round. For each air duct, the user will first select which type and then enter dimensions. One dimension is required for square or round, and two dimensions are required for rectangular. All of the following information is entered regardless of the duct type:

- Length
- Type of metal (galvanized or stainless steel)
- Thickness of metal (8, 10, or 12-gauge)
- Insulation type (interior, exterior, or none)

Surface area is calculated using standard formulas. Accessories are calculated based on the total length of all ductwork, regardless of type or size. For each 10' of ductwork, the computer will allow for one duct hanger and four duct hanger screws.

In the final processing step, DES will determine total cost of all sheet metal and accessories. The unit cost factors will be available in a data file that can be updated by the user.

## Exercise

1. Working in small groups, construct a very top-level (i.e., don't worry about details) design for the Ductwork Estimating System. In particular, decompose the problem in some way so that the work can be divided among a number of programming teams. Your "solution" may take any form that you feel is best: an outline, a paragraph, a drawing or chart.



Evaluation  
Copy

**This Page Intentionally Left Blank**

Evaluation  
Copy



Evaluation  
Copy