Chapter 1

The software development process using Unified Modeling Language (UML)

Challenges of Software Development

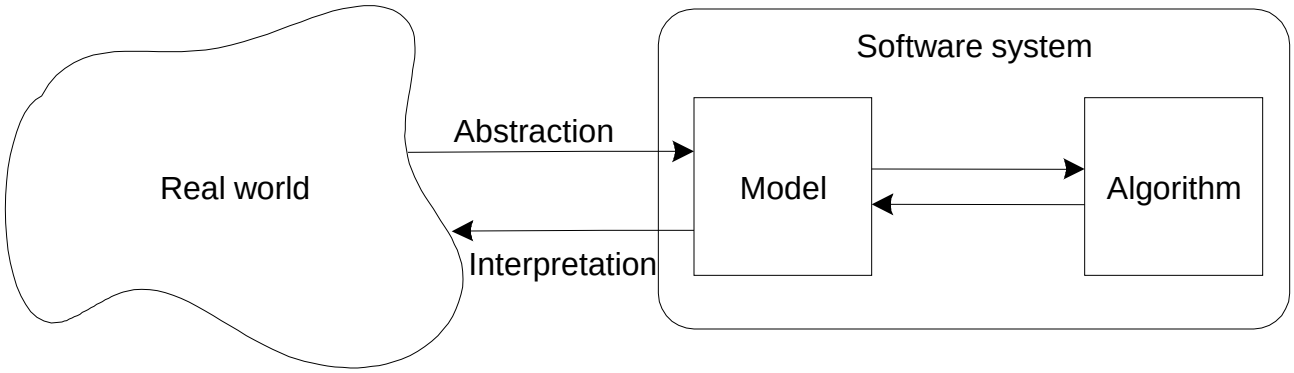
- Complexity of software systems
- Longevity and evolution of software systems
- High user expectations

Software Engineering

- Engineering discipline concerned with all aspects of developing and delivering high-quality and useful software in a cost-effective manner

Modeling the Real World

- A software system provides a solution to a problem in the real world.
- It consists of two essential components:
  - Model: abstraction of a part of the real world
  - Algorithm: captures the computations involved in manipulating or processing the model.



How to Model Real World?

- Programming languages
  - Tools to describe computer models
- Programming models
  - Computation-oriented model (50s ~ 60s)
  - Data-oriented model (70 ~ 80s)
  - Object-oriented model (90s ~)
    - Balanced view between data and computation

Object-Oriented System Development

- OO information system development involves:
  - OOA
    - Using an OO approach to system analysis
  - OOD
    - Using an OO approach to system design
  - OOP
    - Using an OO approach to programming

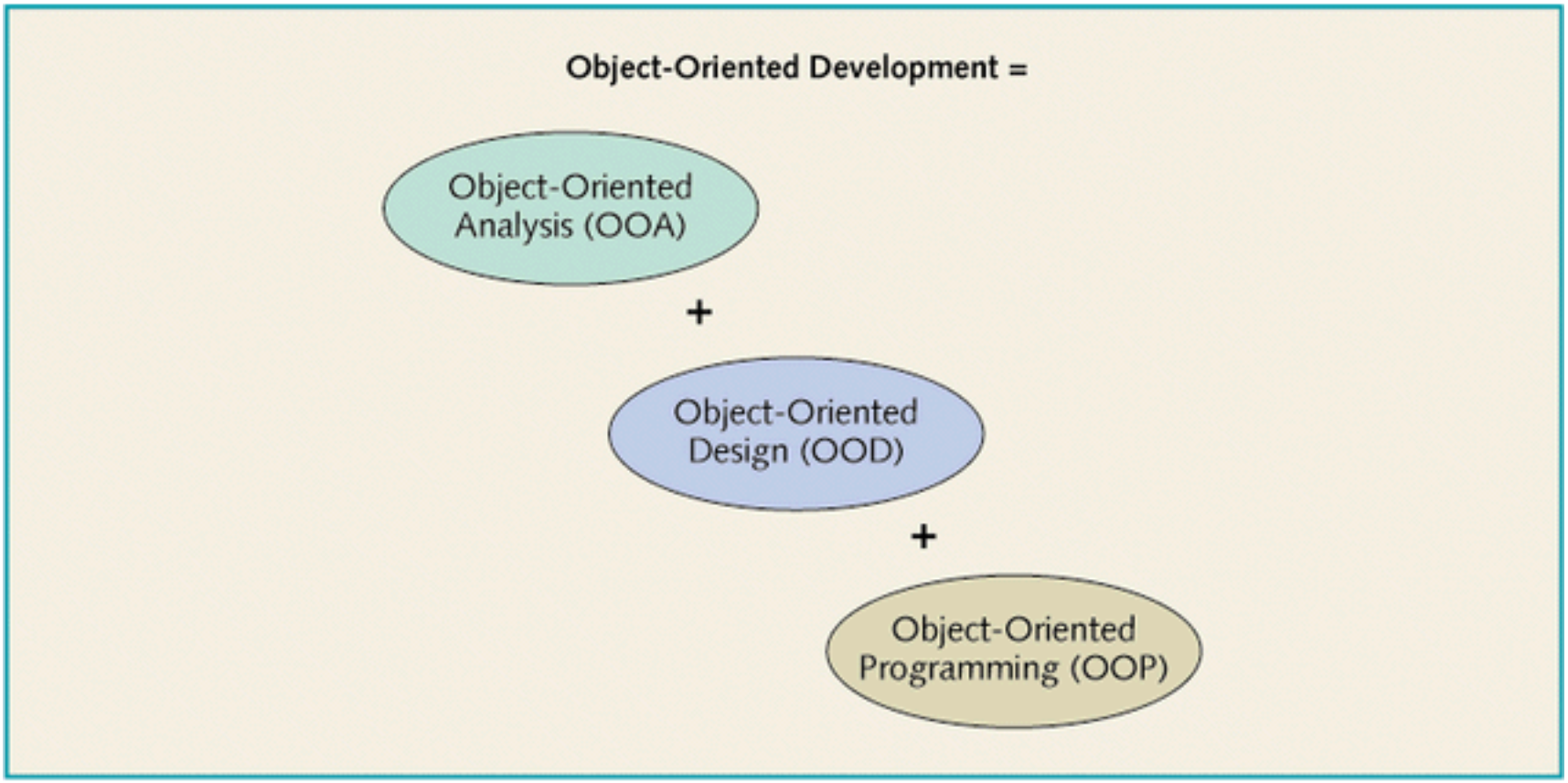


Figure 1-3 Object-oriented development

Object-Oriented Concepts

- **Object-oriented (OO)** design techniques are becoming popular:

– Inception in early 1980's and nearing maturity.

– Widespread acceptance in industry and academics.

– **Unified Modeling Language (UML)** already an ISO standard (ISO/IEC 19501**)** for modeling OO systems.

Understanding OO Development

- OO Analysis and Design
  - Unified Modeling Language (UML)
    - Standard OOA&D modeling notation
    - Defined by: Grady Booch, James Rumbaugh & Ivar Jacobson
    - Uses model-driven approach:
      - Enables creation of graphical models of the system requirements and system design

Understanding OO Development

- OO Analysis and Design
  - System Development Life Cycle (SDLC)
    - Project management framework that defines project phases and activities
    - Phases:
      - Planning
      - Analysis
      - Design
      - Implementation
      - Support

Understanding OO Development

- OO Analysis and Design
  - Prototyping
    - Creating a working model of one or more parts of the system for user evaluation and feedback
  - Joint Application Development (JAD)
    - Key system stakeholders and decision makers work together to rapidly define system requirements and designs

Understanding OO Development

- OO Analysis and Design
  - Other requirements:
    - Project management
    - Interviewing
    - Data collection
    - User interface (UI) design
    - Testing
    - Conversion techniques

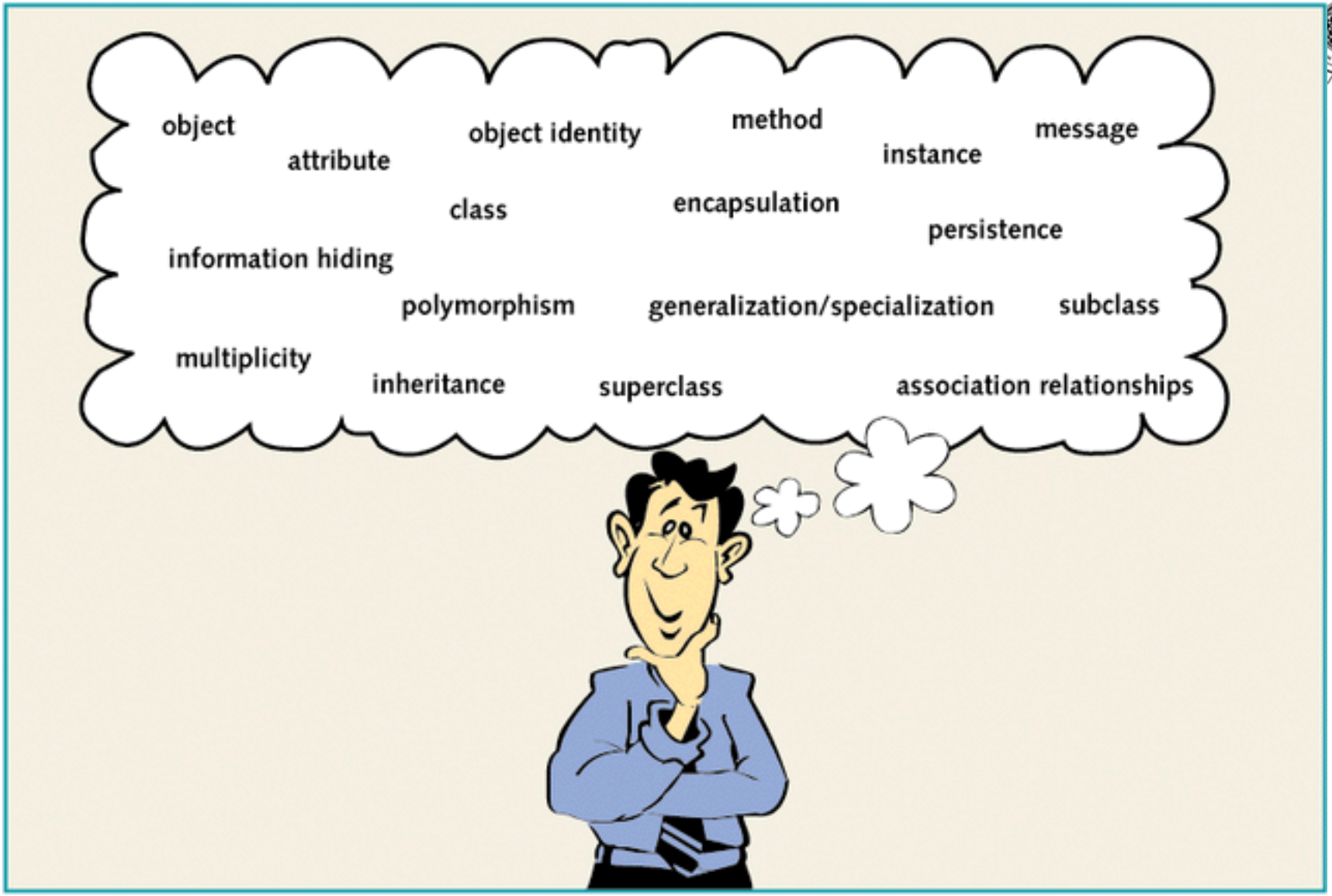


Figure 1-5 Key OO concepts

Object-oriented Design - Objectives

- To explain how a software design may be represented as a set of interacting objects that manage their own state and operations
- To describe the activities in the object-oriented design process
- To introduce various models that can be used to describe an object-oriented design
- To show how the UML may be used to represent these models

Object-Oriented Development

- Approach
  - Focuses on improving the *maintainability* and *reusability* of software systems through a set of techniques, notations, tools, and criteria.
- Activities
  - Conceptualization
  - Object-oriented analysis and modeling
  - Object-oriented design
  - Implementation
  - Maintenance

Detailed Activities

- Conceptualization
  - To establish the vision and core requirements of the software system to be developed.
- Object-oriented analysis and modeling
  - To build models of the system's desired behavior, using notations such as the *Unified Modeling Language* (*UML*).
  - To capture the essential relevant aspects of the real world and to define the services to be provided and/or the problems to be solved.
  - To simplify reality to better understand the system to be developed.

Detailed Activities (Contd.)

- Object-oriented design
  - To create an architecture for implementation.
  - Represented in terms of objects and classes and the relationships among them.
- Implementation:
  - To implement the design by using an object-oriented programming language (e.g., Java)
- Maintenance:
  - To manage postdelivery evolution effectively.

Advantages of OOD

- Easier maintenance. Objects may be understood as stand-alone entities.
- Objects are potentially reusable components.
- For some systems, there may be an obvious mapping from real world entities to system objects.

Object Oriented methodologies.

- The Rumbaugh OMT

- The Booch methodology

- Jacobson's methodologies

Rumbaugh's Object Modeling Technique (OMT)

The object modeling techniques is an methodology of object oriented analysis, design and implementation that focuses on creating a model of objects from the real world and then to use this model to develop object–oriented software. Object modeling technique (OMT) was developed by James Rambaugh. Now-a-days, OMT is one of the most popular object oriented development techniques. It is primarily used by system and software developers to support full life cycle development while targeting object oriented implementations.

Four phases of OMT (can be performed iteratively)

- ***Analysis****:* objects, dynamic and functional models
- ***System Design****:* Basic architecture of the system.
- ***Object Design****:* static, dynamic and functional models of objects.
- ***Implementation****:* reusable, extendible and robust code.

Analysis

- Analysis is the first phase of OMT methodology. The aim of analysis phase is to build a model of the real world situation to show its important properties and domain. This phase is concerned with preparation of precise and correct modelling of the real world. The analysis phase starts with defining a problem statement which includes a set of goals. This problem statement is then expanded into three models; an object model, a dynamic model and a functional model. The object model shows the static data structure or skeleton of the real world system and divides the whole application into objects. In others words, this model represents the artifacts of the system. The dynamic model represents the interaction between artifacts above designed represented as events, states and transitions. The functional model represents the methods of the system from the data flow perspective. The analysis phase generates object model diagrams, state diagrams, event flow diagrams and data flow diagrams.

Three different parts of OMT modeling

- An object model - object model & data dictionary
- A dynamic model - state diagrams & event flow diagrams
- A functional model - data flow & constraints

Object Model

- structure of objects in a system.
- Identity, relationships to other objects, attributes and operations.
- Object diagram

Object Diagram

- Classes interconnected by association lines
- Classes- a set of individual objects
- Association lines- relationship among classes (i.e., objects of one class to objects of another class)

OMT Dynamic Model

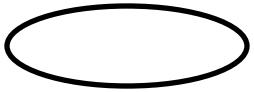
- States, transitions, events and actions
- OMT state transition diagram-network of states and events

OMT Functional Model

- DFD- (Data Flow Diagram)
- Shows flow of data between different processes in a business.
- Simple and intuitive method for describing business processes without focusing on the details of computer systems.

Data Flow Diagram

- Four primary symbols



Process- any function being performed

Data Flow- Direction of data element movement

Data Store – Location where data is stored

External Entity-Source or Destination of a data element

System design

- The system design phase comes after the analysis phase. System design phase determines the overall system architecture using sub-systems, concurrent tasks and data storage. During system design, the high level structure of the system is designed. The decisions made during system design are:
  - The system is organized in to sub-systems which are then allocated to processes and tasks, taking into account concurrency and collaboration.
  - Persistent data storage is established along with a strategy to manage shared or global information.
  - Boundary situations are checked to help guide trade off priorities.

Object design

- The object design phase comes after the system design phase is over. Here the implementation plan is developed. Object design is concerned with fully classifying the existing and remaining classes, associations, attributes and operations necessary for implementing a solution to the problem. In object design:
  - Operations and data structures are fully defined along with any internal objects needed for implementation.
  - Class level associations are determined.
  - Issues of inheritance, aggregation, association and default values are checked.

Implementation

- Implementation phase of the OMT is a matter of translating the design in to a programming language constructs. It is important to have good software engineering practice so that the design phase is smoothly translated in to the implementation phase. Thus while selecting programming language all constructs should be kept in mind for following noteworthy points.
  - To increase flexibility.
  - To make amendments easily.
  - For the design traceability.
  - To increase efficiency.

The Booch Methodology

- Widely used OO method
- Uses the object paradigm
- Covers the design and analysis phase of an OO system
- Criticized for his large set of symbols

Diagrams of Booch method

- **Class diagrams**

  describe roles and responsibilities of objects
- **Object diagrams**

  describe the desired behavior of the system in terms of scenarios
- **State transition diagrams**

  state of a class based on a stimulus
- **Module diagrams**

  to map out where each class & object should be declared
- **Process diagrams**

  to determine to which processor to allocate a process
- **Interaction diagrams**

  describes behavior of the system in terms of scenarios

Jacobson Methodology

- Use Cases.
- Object Oriented Software Engineering.
- Object Oriented Business Engineering.

Use Cases

- Understanding system requirements
- Interaction between Users and Systems
- The use case description must contain
  - How and when the use case begins and ends.
  - The Interaction between the use case and its actors, including when the interaction occurs and what is exchanged.
  - How and when the use case will need data stored in the system.
  - Exception to the flow of events
  - How and when concepts of the problem domain are handled.

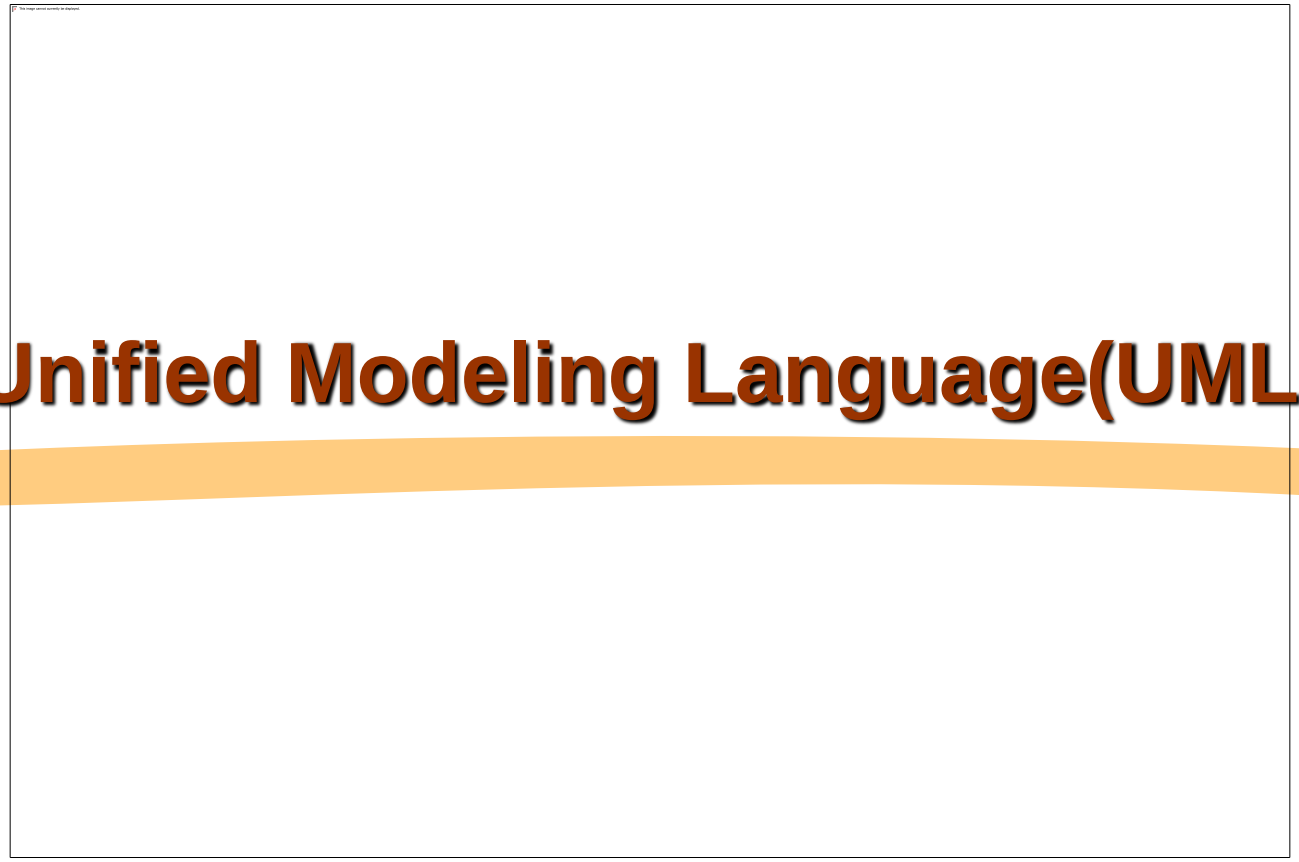
OOSE

- Object Oriented Software Engineering.
- Objectory is to built models
  - Use case model
  - Domain object model
  - Analysis object model
  - Implementation model
  - Test model

OOBE

- Object Oriented Business Engineering
- OOBE is object modeling at the enterprise level.
  - Analysis phase
  - Design and Implementation phase
  - Testing phase
    - E.g. Unit testing, integration and system testing.

Unified Modeling Language(UML)



Object Modeling Using UML

- UML is a modeling language
- Not a system design or development methodology
- Used to document object-oriented analysis and design
- Independent of any specific design methodology

UML Origin

- In late 1980s and early 1990s different software development houses were using different notations.
- Developed in early 1990s to:

– Standardize the large number of object-oriented modeling notations.

UML Lineology

Based Principally on

– OMT [Rumbaugh 1991]

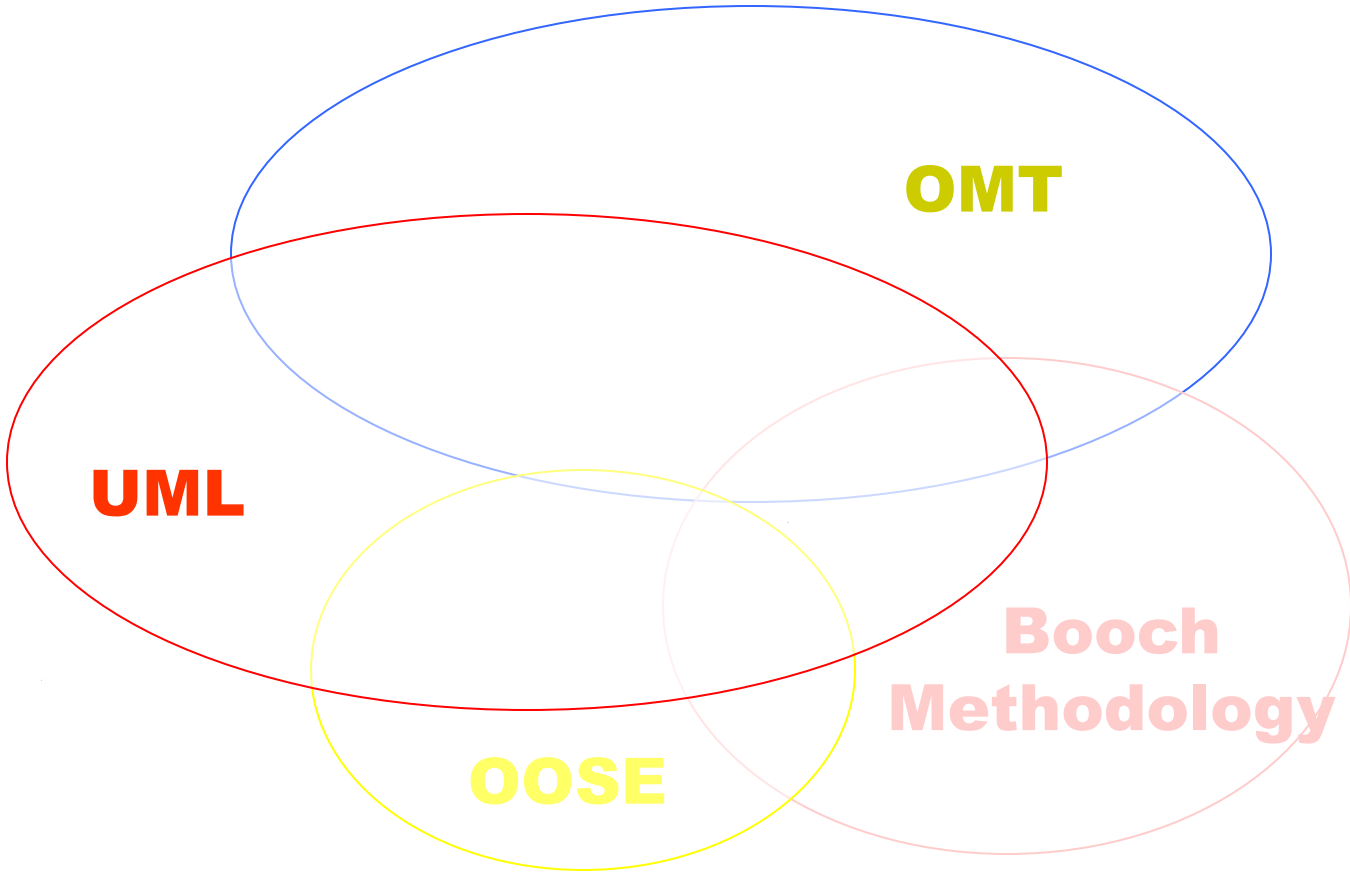
– Booch's methodology[Booch 1991]

– OOSE [Jacobson 1992]

– Odell's methodology[Odell 1992]

– Shlaer and Mellor [Shlaer 1992]

UML



Different object modeling techniques in UML

UML as A Standard

- Adopted by Object Management Group (OMG) in 1997
- OMG is an association of industries
- Promotes consensus notations and techniques
- Used outside software development

– Example: car manufacturing

UML Definition by OMG

- The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.

- The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components.

Why are UML Models Required?

- A model is required to capture only important aspects
- UML a graphical modelling tool
- Easy to understand and construct
- Helps in managing complexity

UML Diagrams

- Nine diagrams are used to capture different views of a system.
- Provide different perspectives of a software system.
- Diagrams can be refined to get the actual implementation of a system.

UML Model Views

- Views of a system:

– User's view

– Structural view

– Behavioral view

– Implementation view

– Environmental view

User's view

- Defines the functionalities(facilities) made available by the system to the users.
- User's view captures external user's view.
- It is black box view of the system.
- Internal structure, the dynamic behavior of the different system components, the implementations are not visible.
- Considered as central view.

Structural view

- Defines the kinds of objects (classes) important to the understanding of a system and to its implementation.
- Captures relationship among classes (objects)
- Static model, not change with time.

Behavioral view

- Captures how objects interact with each other to realize system behavior.
- Dynamic model, change with time.

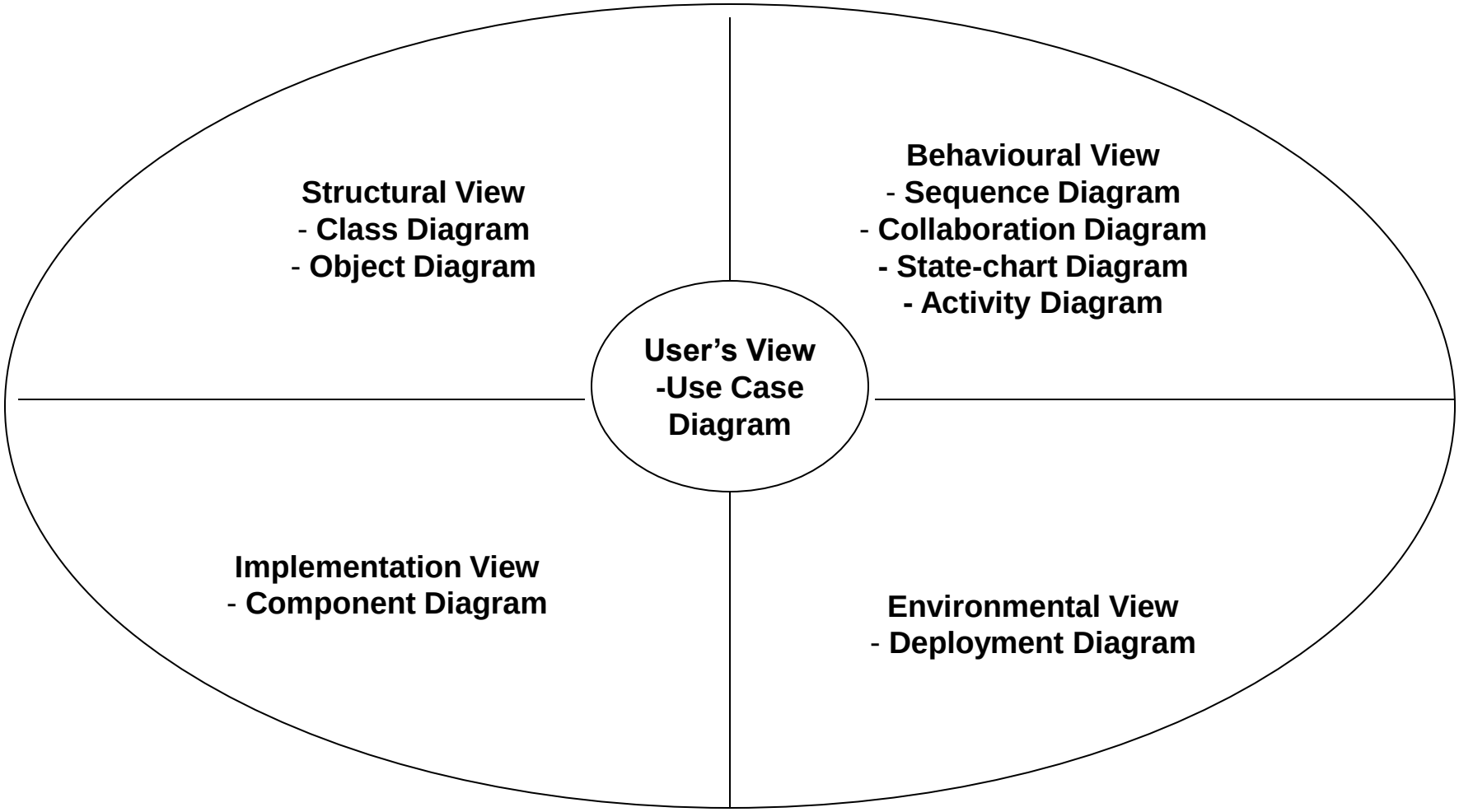
Implementation view

- Captures the important components of the system and their dependencies.

Environmental view

- Shows how the different components are implemented on different pieces of hardware.

Diagrams and views in UML



Are All Views Required for Developing A Typical System?

- NO
- Use case diagram, class diagram and one of the interaction diagram for a simple system
- State chart diagram required to be developed when a class state changes
- However, when states are only one or two, state chart model becomes trivial
- Deployment diagram in case of large number of hardware components used to develop the system

Thank you

Questions?