

## Agenda

- Why Model?
- History of OO Modeling Methodologies
- Object Modeling Technique (OMT)
- Unified Modeling Language (UML)

---

---

---

---

---

---

---

## Why Model

*def'n: simplification of reality*

- Create a Successful Product
- Aids in Better Understanding of System
- Attack Hard Problem through Smaller, Solvable Problems
- Presents Need for Common Language for Modeling

---

---

---

---

---

---

---

## Four Fundamental Principles of Modeling

- The choice of what models to create has a profound influence on how a problem is attacked and how the solution is shaped
- The best models are connected to reality
- Every model may be expressed at different levels of precision/abstraction
- No single model is sufficient. Every nontrivial system is best approached through a small set of nearly independent models

---

---

---

---

---

---

---

## History of OO Modeling Methods

- '70s-'80s: Three Prominent Methods Appear (Booch, OOSE, OMT)
- mid-90's: evolving towards each other
- OC94: Rumbaugh (OMT) joins Rational
- OC95: Jacobsen (OOSE) joins Rational
- JA97: UML 1.0 released
- NO98: UML 1.3 released

---

---

---

---

---

---

---

## Object Modeling Technique (OMT)

### Object Model

- Describes the structure of objects in the system-their identity, relationships to other objects, attributes, and operations
- Captures those concepts from the real world that are important
- Diagrams: *class diagrams* and *object diagrams*

---

---

---

---

---

---

---

## Object Modeling Technique (OMT)

### Dynamic Model

- Describes those aspects of a system concerned with time and the sequencing of operations
- Captures the control aspects of a system
- Diagrams: *state diagrams*

---

---


---

---

---

---

---



### Object Modeling Technique (OMT)

Functional Model

- Describes those aspects of a system concerned with transformations of values
- Captures what the system does, without regard for how or when it is done
- Diagrams: *dataflow diagrams*

---

---


---

---

---

---

---



### Unified Modeling Language (UML)

Structural Family: *Class Diagrams*

- Shows set of classes, interfaces, collaborations, and relationships
- Most common diagram for OO systems
- Addresses static view of system

---

---


---

---

---

---

---



### Unified Modeling Language (UML)

Structural Family: *Packages*

- Not really a separate diagram, but rather a general-purpose mechanism for organizing elements
- Shown as a tabbed folder

---

---


---

---

---

---

---



### Unified Modeling Language (UML)

Structural Family: *Object Diagrams*

- Shows a set of objects and their relationships
- Shows instantiation of class diagram. That is, shows static snapshot of implementation of class diagram
- Addresses static view of system

---

---


---

---

---

---

---



### Unified Modeling Language (UML)

Behavioral Family: *Statechart Diagrams*

- Shows a state machine consisting of states, transitions, events, and activities
- Addresses dynamic/behavioral view of system

---

---


---

---

---

---

---



### Unified Modeling Language (UML)

Behavioral Family: *Activity Diagrams*

- Shows flow from activity to activity within a system
- Special type of statechart diagram
- Important for modeling the flow of control among objects

---

---


---

---

---

---

---



### Unified Modeling Language (UML)

Behavioral Family: *Use Case Diagrams*

- Shows a set of use cases and actors
- Addresses static use case view of system

---

---


---

---

---

---

---



### Unified Modeling Language (UML)

Behavioral Family: *Interaction Diagrams*

- Shows an interaction consisting of a set of objects, relationships, and messages dispatched among them
- Made up of *sequence diagrams* and *collaboration diagrams*
  - Sequence diagrams: interaction diagram emphasizing the time-ordering of messages
  - Collaboration diagrams: interaction diagram emphasizing structural organization of objects that send and receive messages

---

---


---

---

---

---

---



### Unified Modeling Language (UML)

Architectural Family: *Component Diagrams*

- Shows organizations and dependencies among a set of components
- Static implementation view of a system
- Closely linked to class diagram. That is, there is often a mapping from classes to components

---

---


---

---

---

---

---



## Unified Modeling Language (UML)

Architectural Family: *Deployment Diagrams*

- Shows the configuration of run-time processing nodes and the components that live on them
- Deployment view of architecture

---

---

---

---

---

---

---