The Unified Modeling Language

by

Peter DePasquale

8/30/99

UML Background

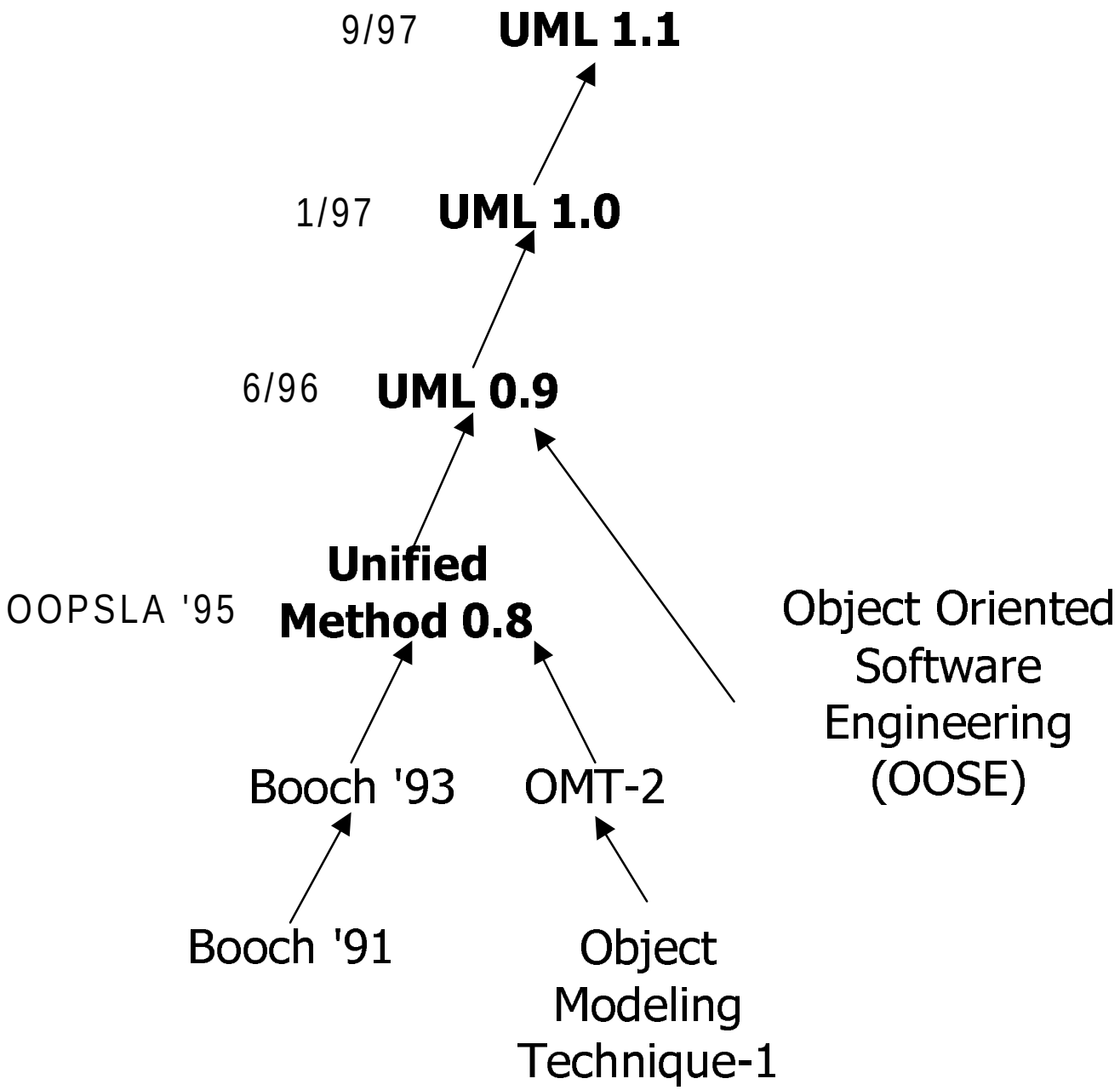
- **What is the UML? --** "The UML is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system"

- **Why model? --**
  - helps enforce communication among teams
  - assures architectural soundness

Goals of the UML

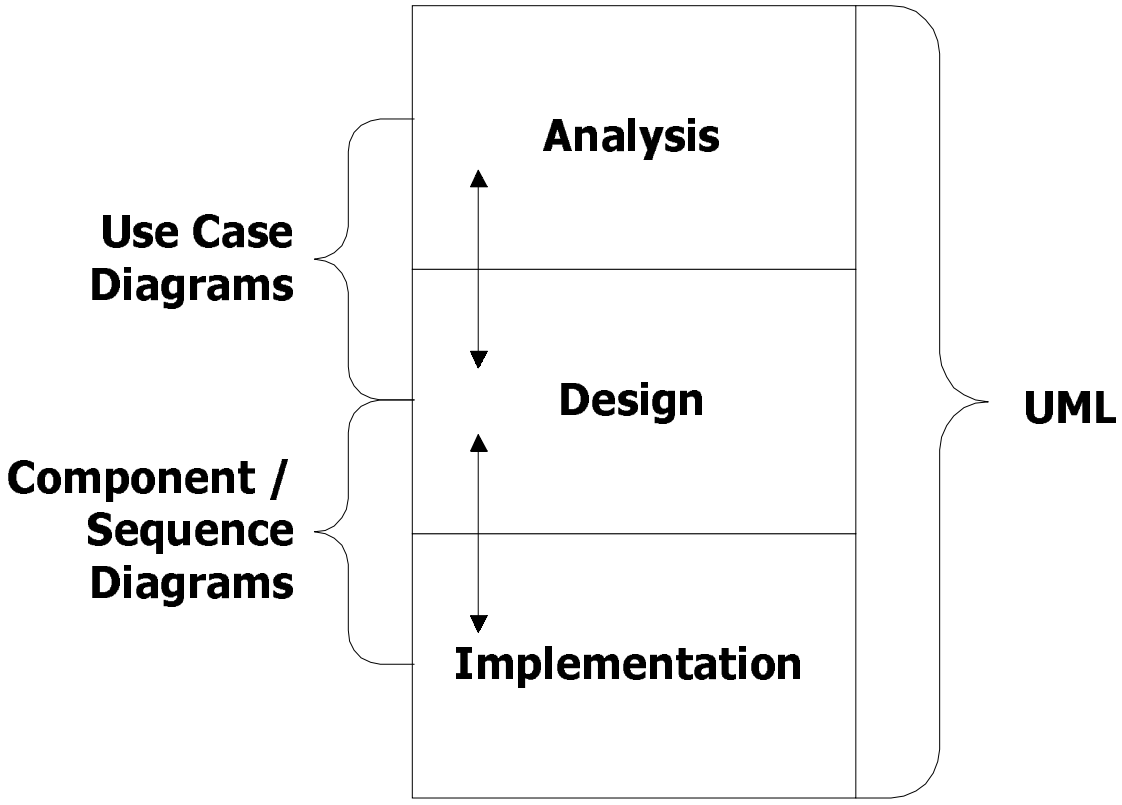
- Be independent of particular programming language and development methodology
- Support higher-level development concepts such as collaborations, frameworks, patterns and components
- Provide extensibility and specialization mechanisms to extend the core concepts

Brief history of UML



UML's Goal

- Represent an Object-Oriented software system throughout its development lifecycle

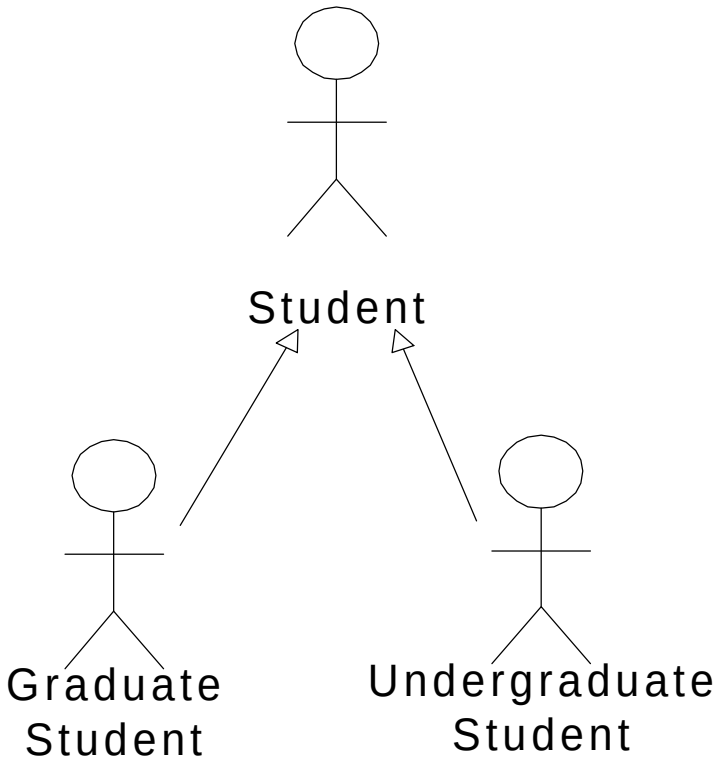


“Use Case” Diagrams

- **Objectives**
  - Specifies the behavior of the system (or part of it) and is description of a set of sequences of actions that a system performs to yield an observable result of value to an actor
  - Shows the relationship between the system and its environment

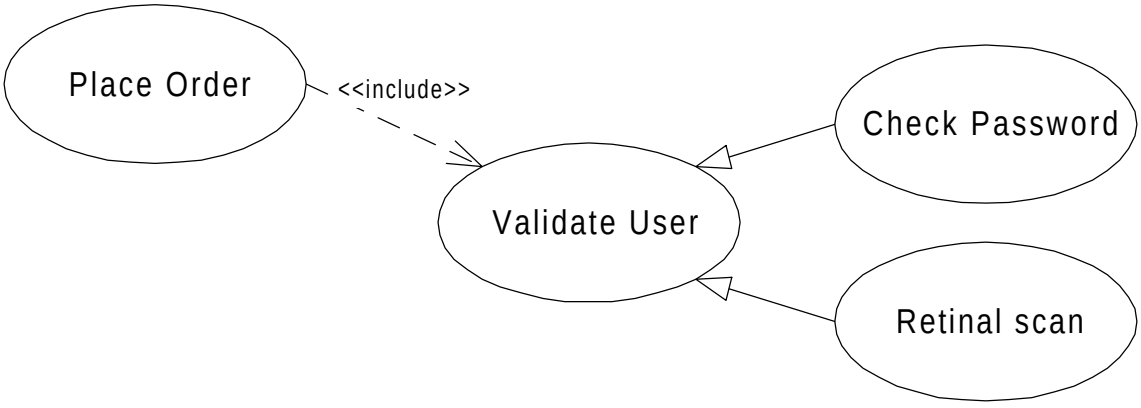
Use Case Diagrams

- Notation:
  - *Actors:*
    - represent role played by person or thing interacting with system
    - represented as stick figures
    - can be specialized/generalized



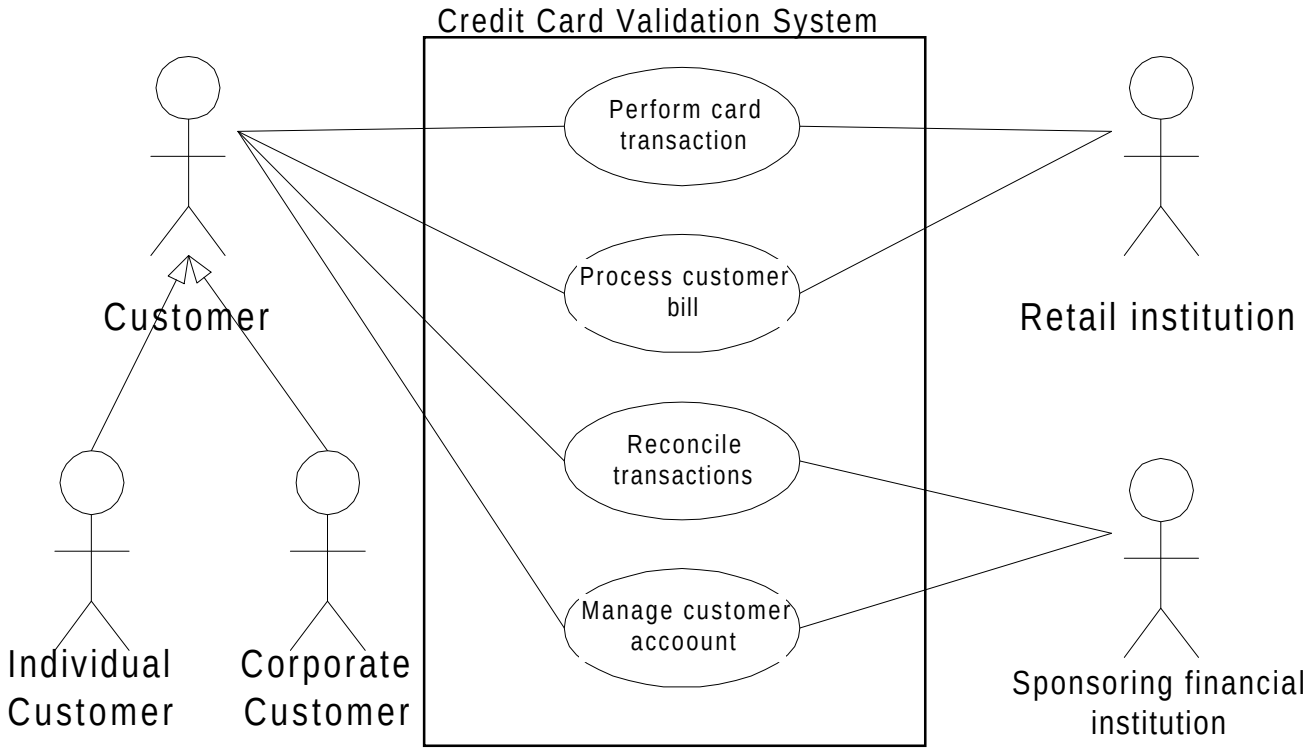
Use Case Diagrams

- Notation:
  - *Use Cases:*
    - captures intended behavior of system
    - behavior specified by describing flow of events (informal structured text, pseudocode, formal structured text)
    - graphically represented as oval
    - can be specialized/generalized/included



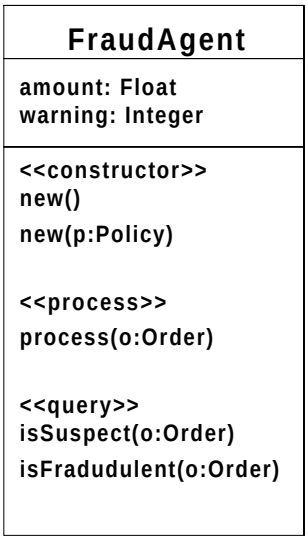
Use Case Diagram example

- Sample ATM banking system



Class Diagrams

- Objective:
  - To represent a set of classes, interfaces and collaborations and their relationships
- Notation:
  - Class representation: rectangle with 3 regions: name, attributes and operations



Sequence Diagrams

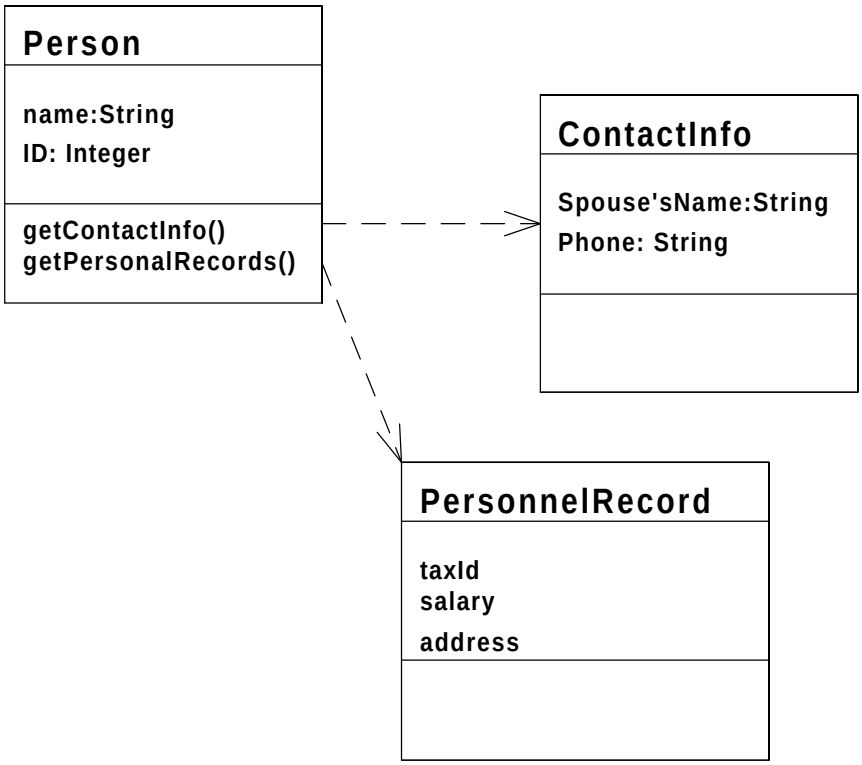
- Objective:
  - Represent object interactions over time

- Notation:
  - *Objects* - named rectangle

  - *Lifelines* - dashed vertical lines showing life of object

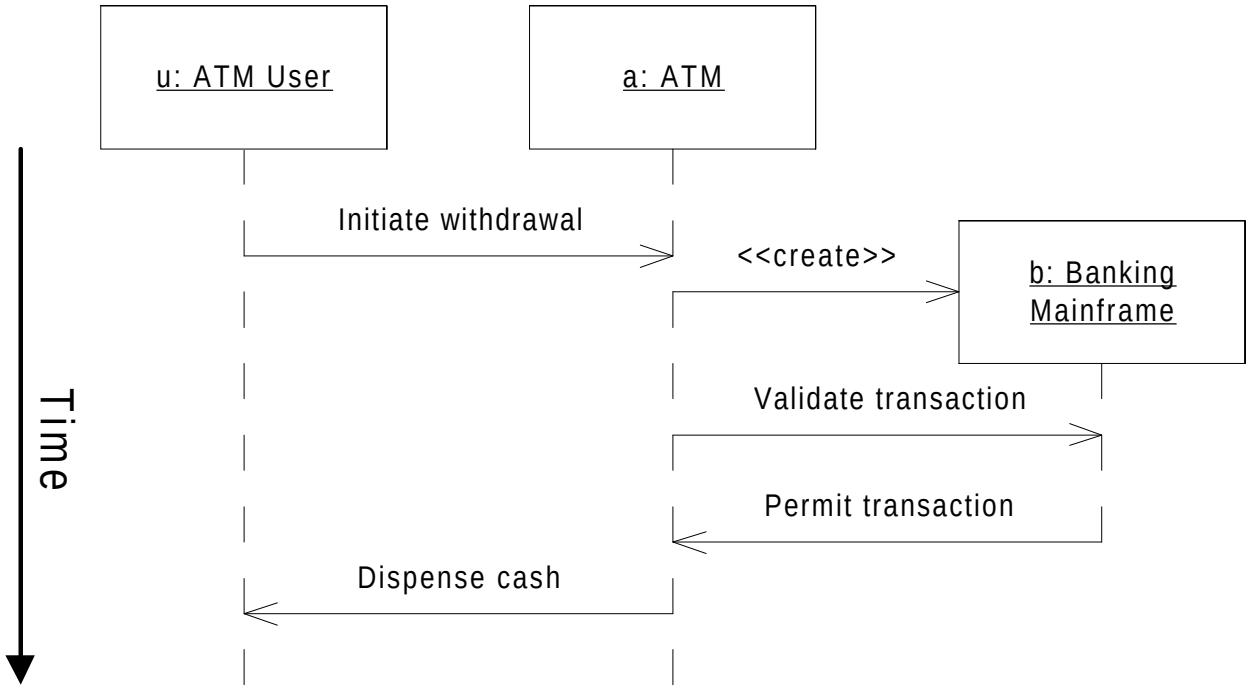
  - *Events* (within system domain) - directional arrows with labels

Class Diagram Example



Sequence Diagram example

- Simple description of interactions of use case objects



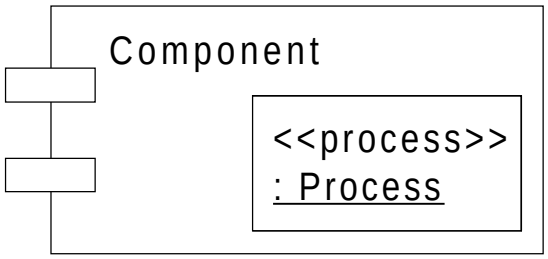
Component Diagrams

- Objective: - Model the static implementation view of a system:
  - source code
  - executable releases
  - physical databases
  - adaptable systems

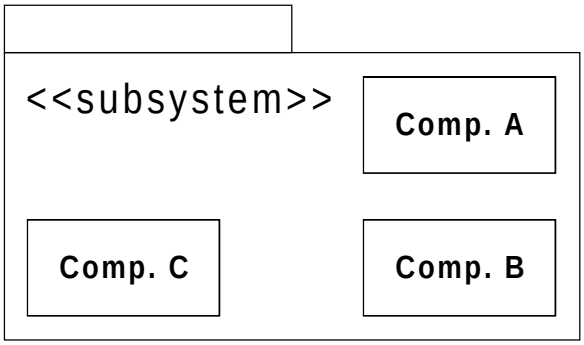
- Use to visualize static aspects of system's physical and their relationships/specify their details for construction

Component Diagrams

– *Processes*

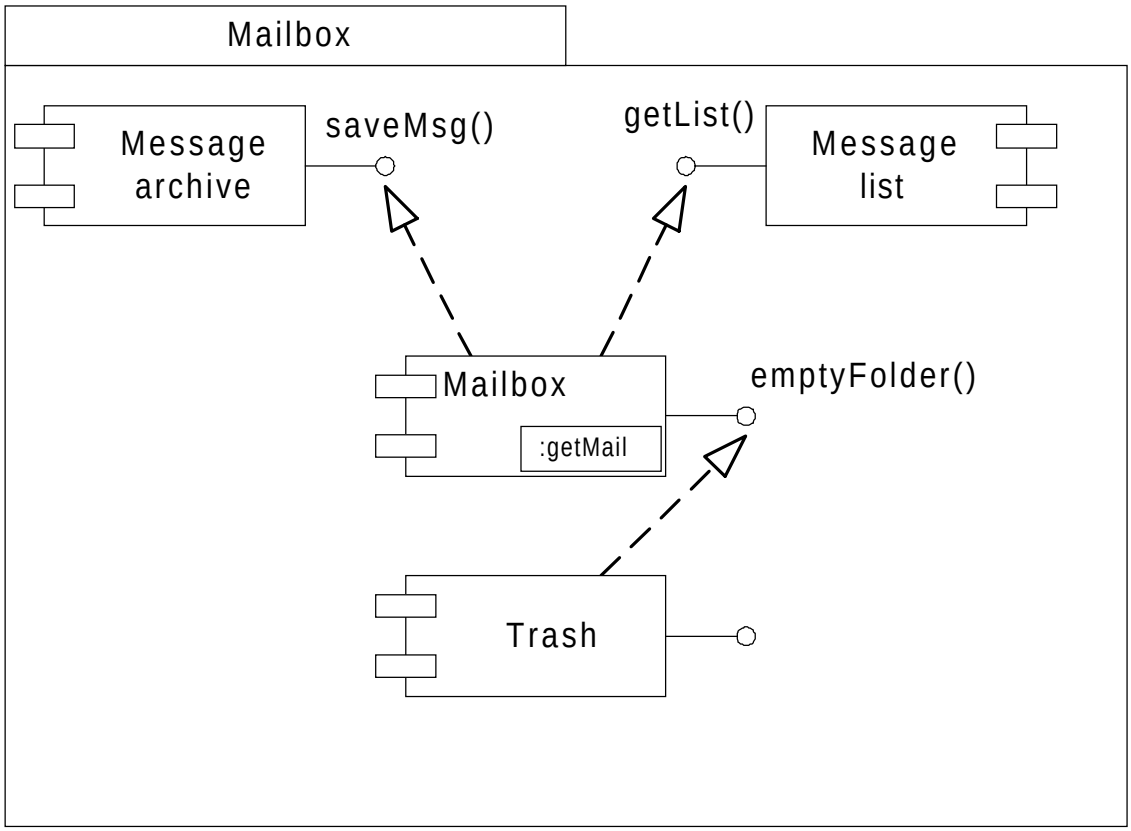


– *Subsystems*



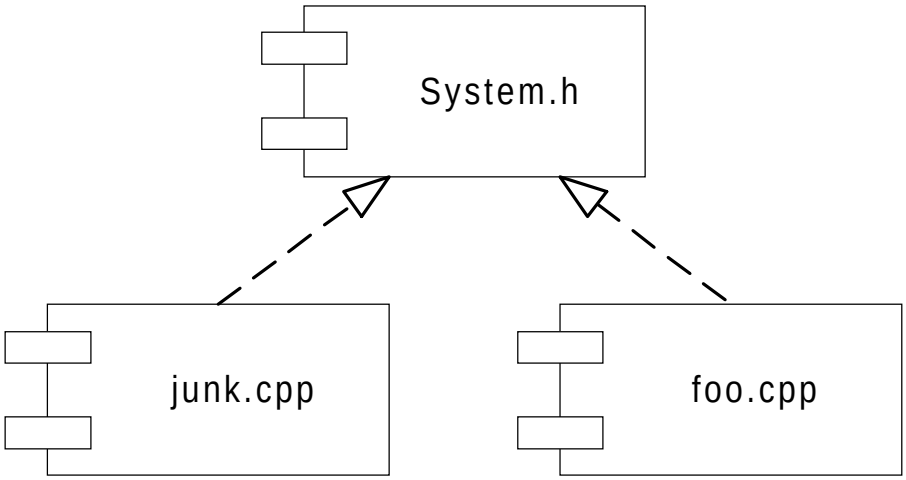
Calling Dependency Diagram example

- Showing component calling dependencies

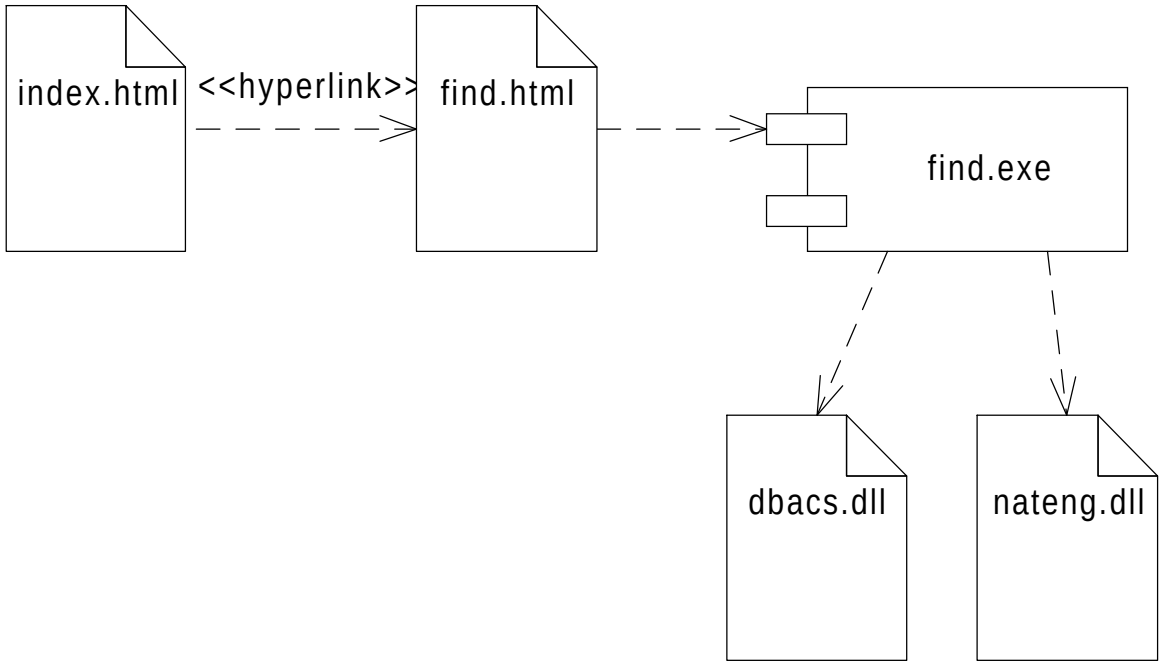


Compilation Dependency Diagram example

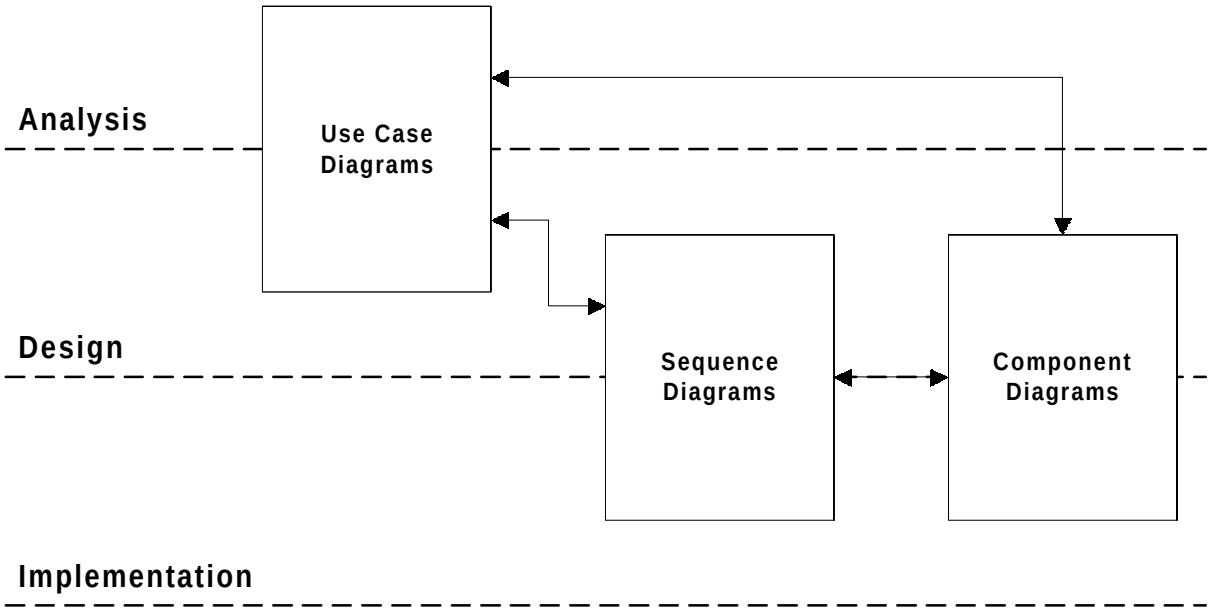
- Showing dependencies during compilation



Component Dependency Diagram example



Partial UML Architecture



Summary

- UML supports representation of system throughout its development (analysis, design and implementation)

- Use Case diagrams - describe system's behavior during requirements analysis and design

- Component diagrams - shows relationships between software components during design and implementation

- Sequence diagrams - represents object interaction from a temporal standpoint