Unified Modeling Language

Object Oriented Methods

- What are object-oriented (OO) methods?
  - OO methods provide a set of techniques for analyzing, decomposing, and modularizing software system architectures
  - In general, OO methods are characterized by structuring the system architecture on the basis of its objects (and classes of objects) rather than the actions it performs
- What are the benefits of OO?
  - OO enhances key software quality factors of a system and its constituent components
- What is the rationale for using OO?
  - In general, systems evolve and functionality changes, but objects and classes tend to remain stable over time

UML

- Is a *language*. It is not simply a notation for drawing diagrams, but a complete language for capturing knowledge (semantics) about a subject and expressing knowledge (syntax) regarding the subject for the purpose of communication.

- It is the result of *unifying* the information systems and technology industry’s best engineering practices (principals, techniques, methods and tools).

UML (contd)

- Unified because it ...
  - Combines main preceding OO methods (Booch by *Grady Booch, OMT by Jim Rumbaugh and OOSE by Ivar Jacobson)*
- Modeling because it is ...
  - Primarily used for visually modeling systems. Many system views are supported by appropriate models
- Language because ...
  - It offers a syntax through which to express modelled knowledge

UML Elements

- *Functional requirements view*
  - Emphasizes the functional requirements of the system from the user's point of view.
- *Static structural view*
  - Emphasizes the static structure of the system using objects, attributes, operations, and relationships.
- *Dynamic behavior view*
  - Emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects.

UML (again)

- *What it is*
  - • A language for capturing and expressing knowledge
  - • A technology for visual development modeling
  - • A set of well-founded guidelines
  - • A milestone generator
  - • A popular (therefore supported) technology
- What it is not
  - • A visual programming language or environment
  - • A database specification tool
  - • A development process (i.e. an SDLC)
  - • A silver bullet
  - • A quality guarantee

UML Diagrams

- **Functional Requirements**
  - **Use-Case** *(relationship between actors and system functions)*

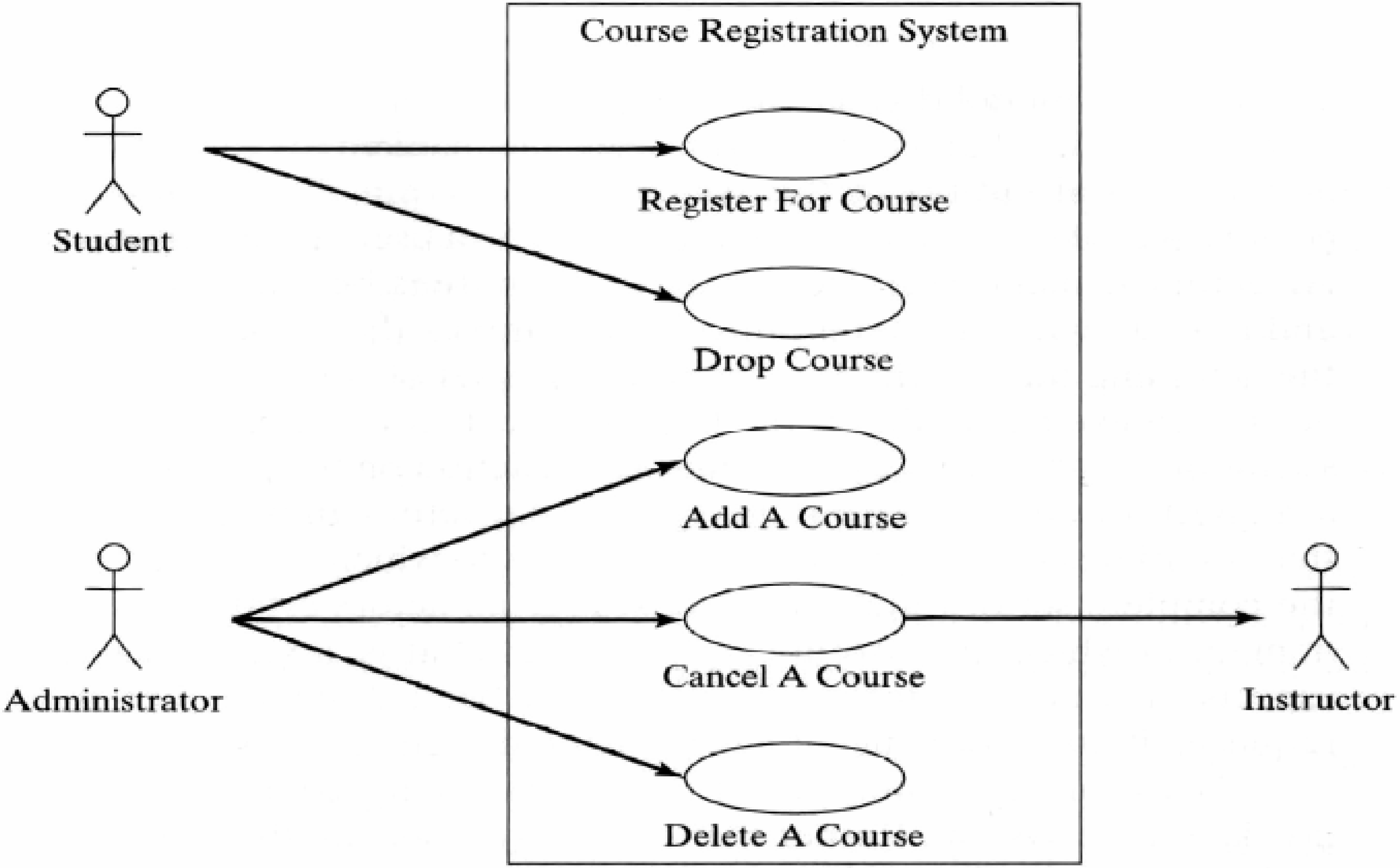
- **Structure**
  - **Class** *(static class structure)*
  - **Object** *(same as class - only using class instances – i.e. objects)*
  - **Package** *(logical grouping of classes)*
  - **Component** *(code structure)*
  - **Deployment/Implementation** *(mapping of software to hardware)*

- **Behavior**
  - **State** *(states of objects in a particular class)*
  - **Sequence** *(Object message passing structure)*
  - **Collaboration/Communication** *(same as sequence but also shows context - i.e. objects and their relationships)*
  - **Activity** *(sequential flow of activities i.e. action states)*

Main 4 UML Diagrams

- Use-Case
- Class
- Sequence
- State/Statechart

The Use-Case Diagram

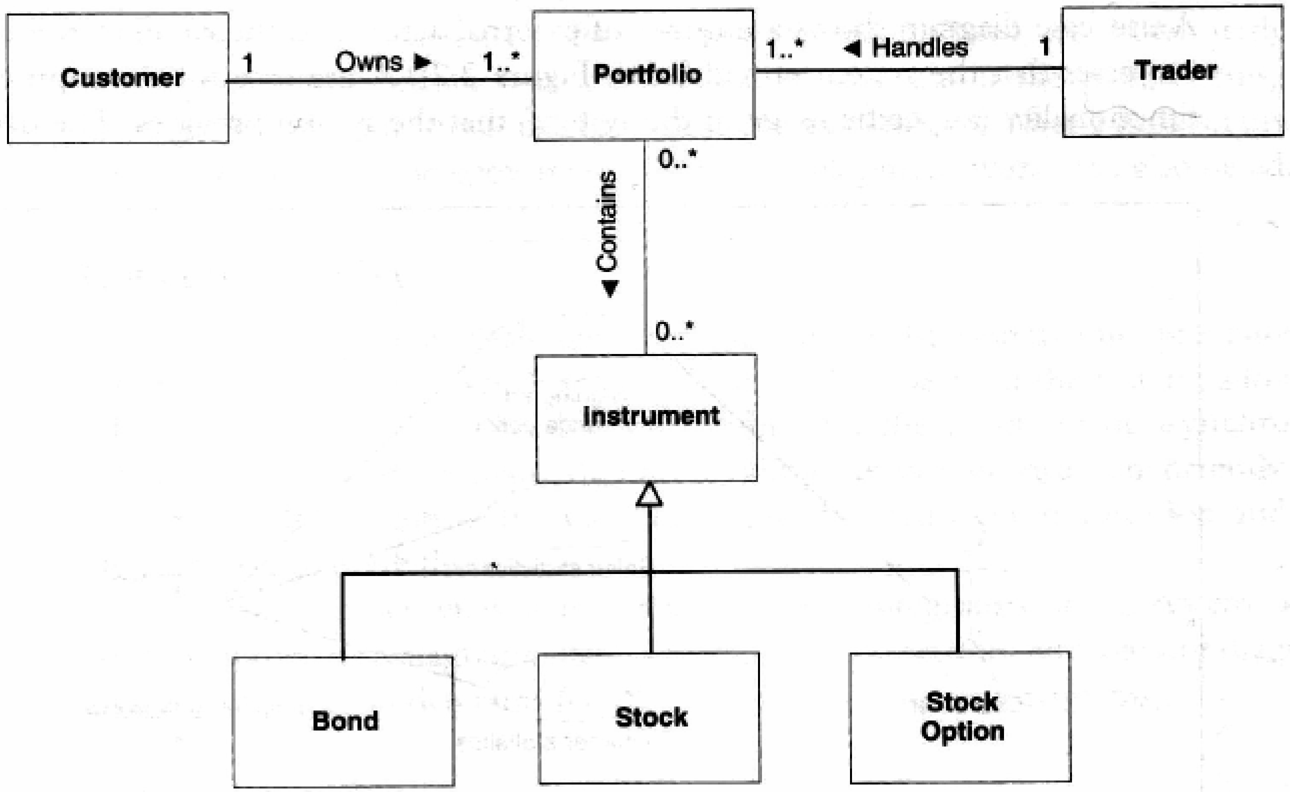


Use Case Diagram

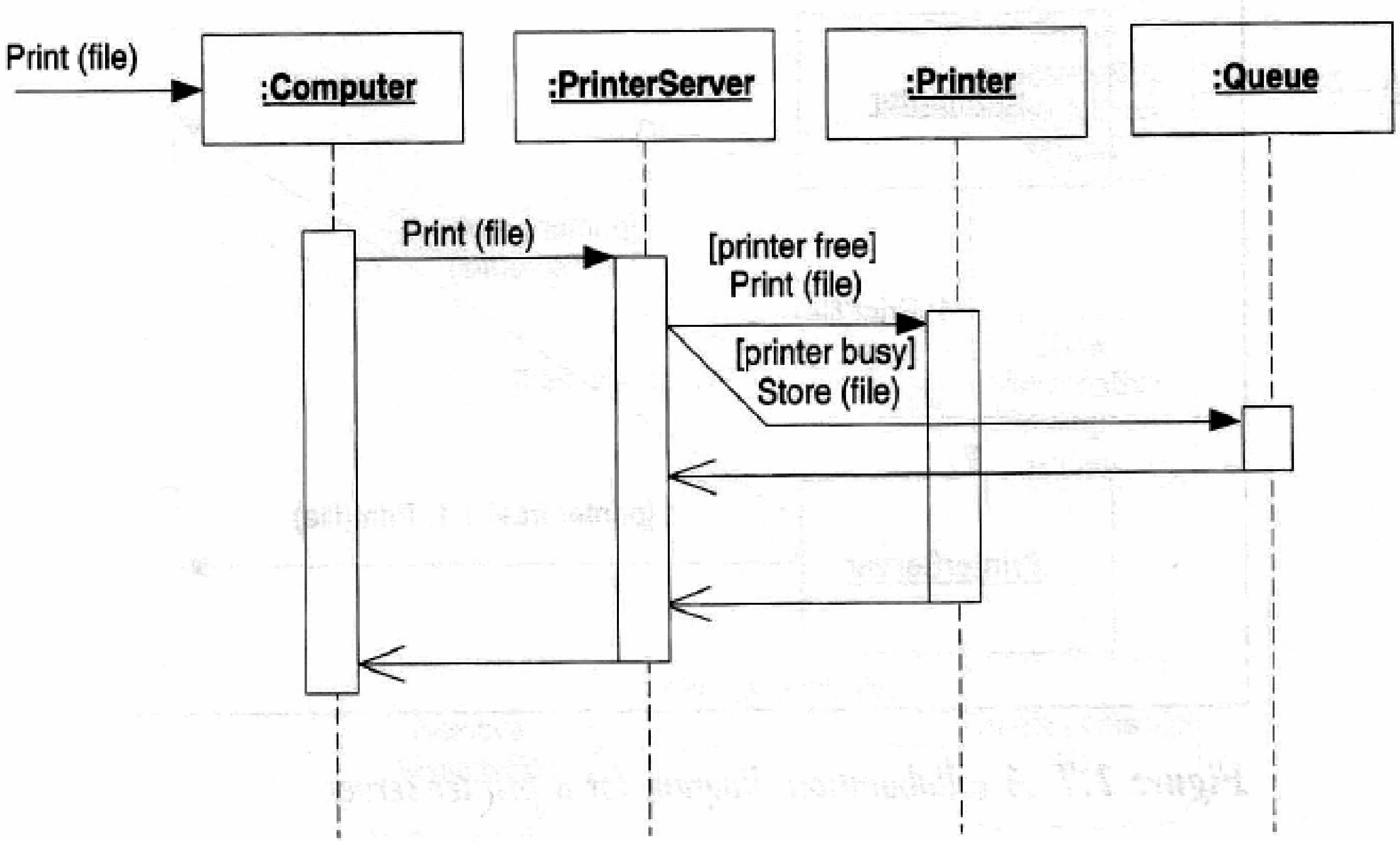
- Components
  - Actor
  - Goals
  - Relationships

- Use-case Relationship
  - Uses – Consumes
  - Generalization - Specialization
  - Extends - Inherits

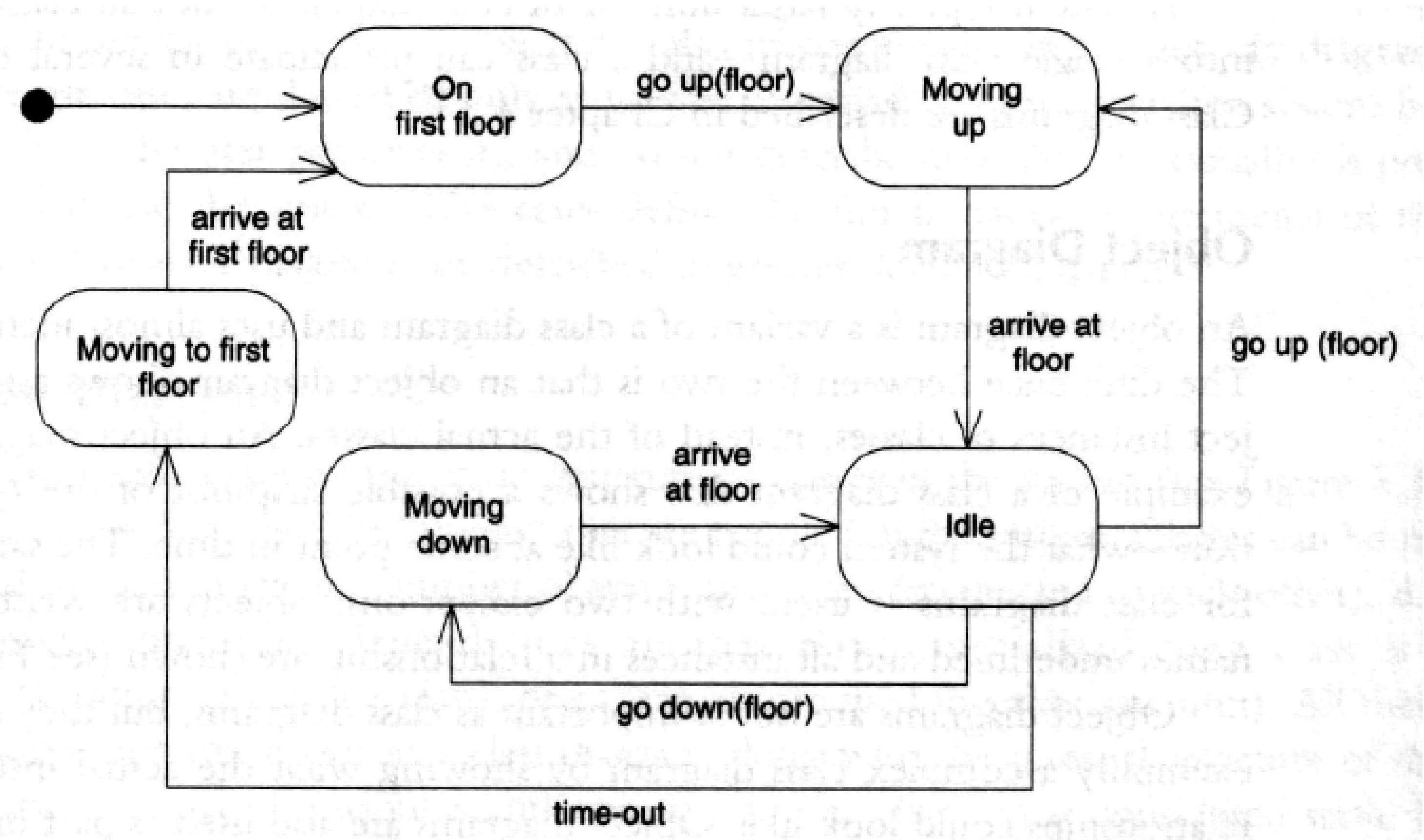
The Class Diagram



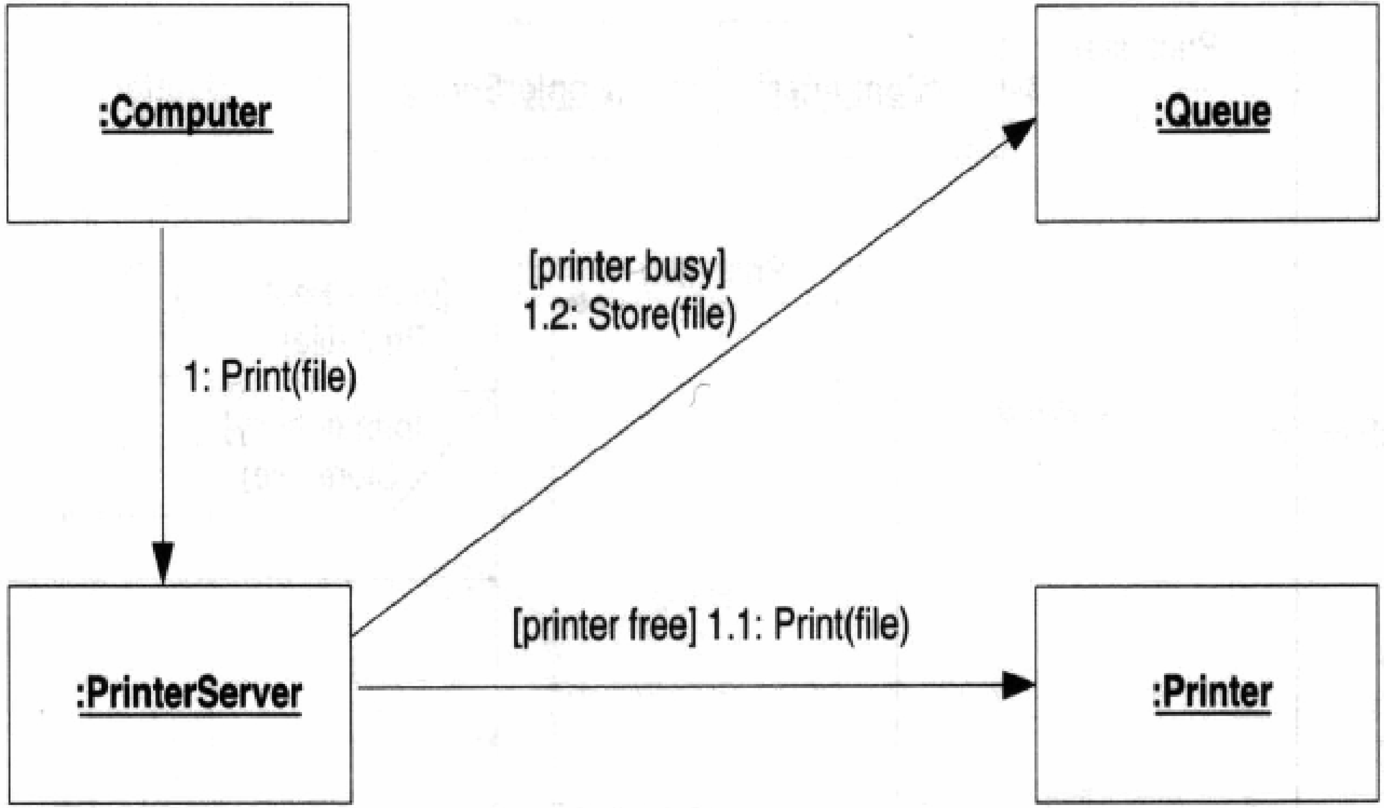
The Sequence Diagram



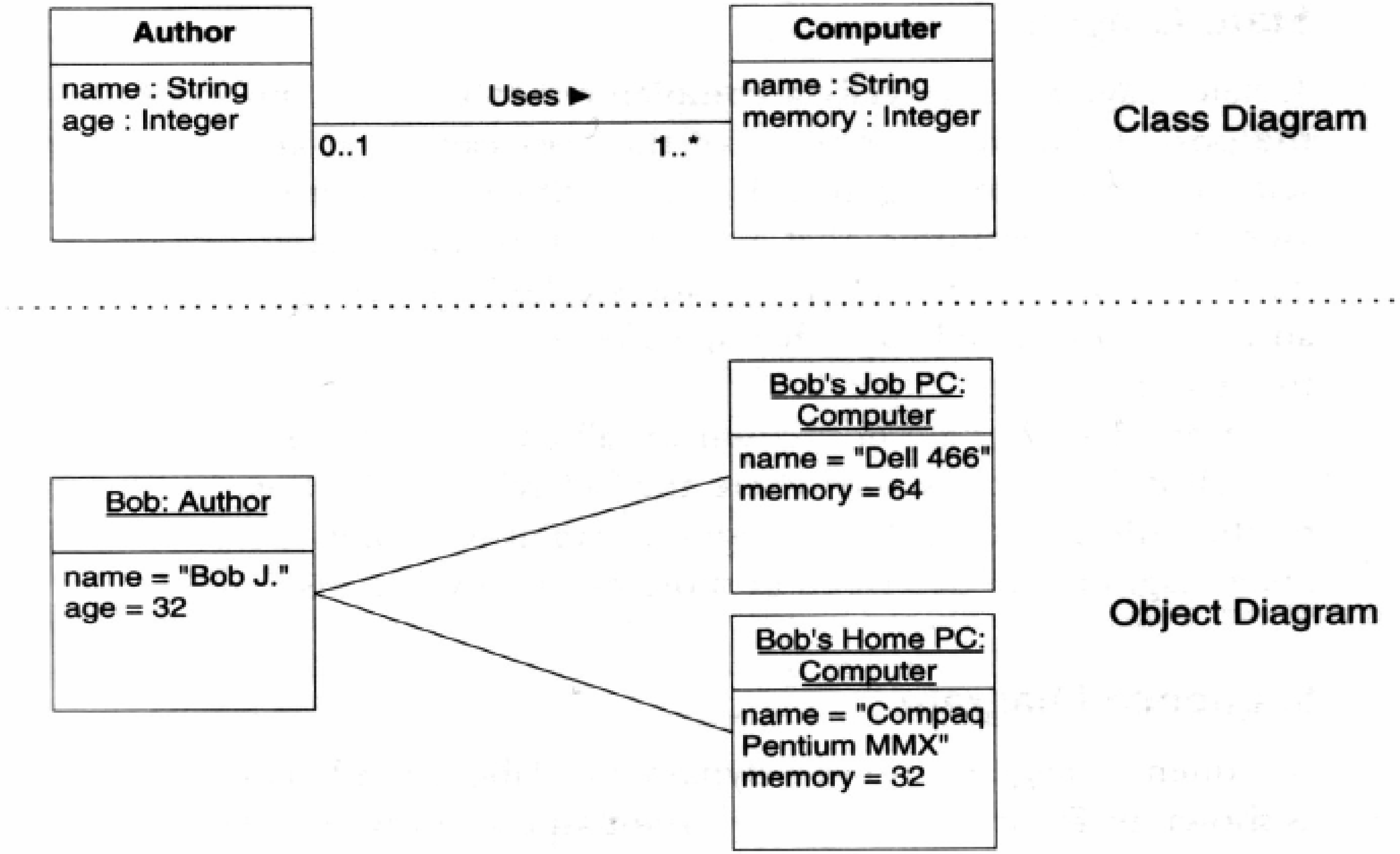
The State Diagram



The Collaboration Diagram



The Object Diagram



Author
name : String
age : Integer

Uses ▶

0..1

1..*

Computer
name : String
memory : Integer

Class Diagram

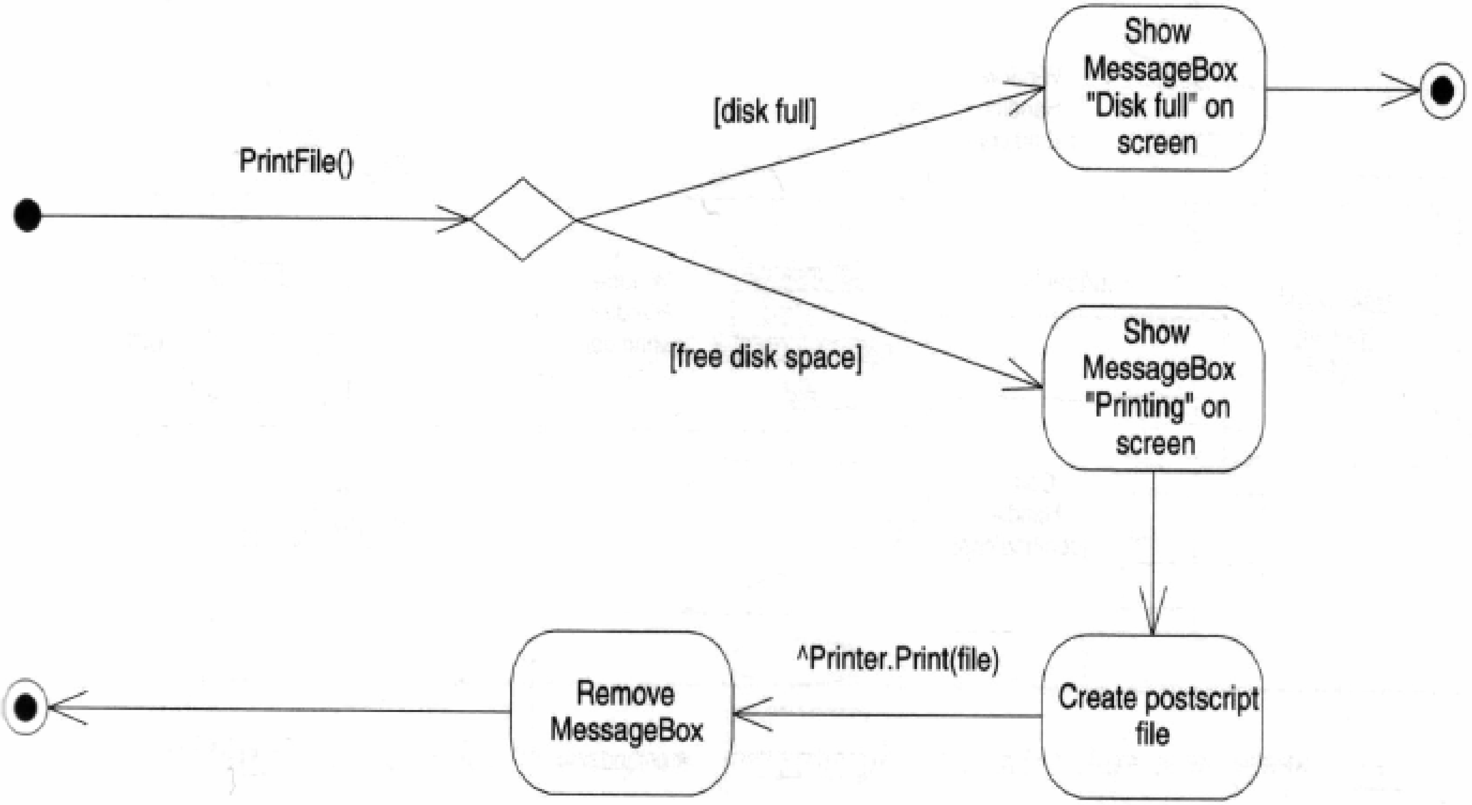
Bob: Author
name = "Bob J."
age = 32

Bob's Job PC: Computer
name = "Dell 466"
memory = 64

Bob's Home PC: Computer
name = "Compaq Pentium MMX"
memory = 32

Object Diagram

The Activity Diagram



PrintFile()

[disk full]

Show MessageBox "Disk full" on screen

[free disk space]

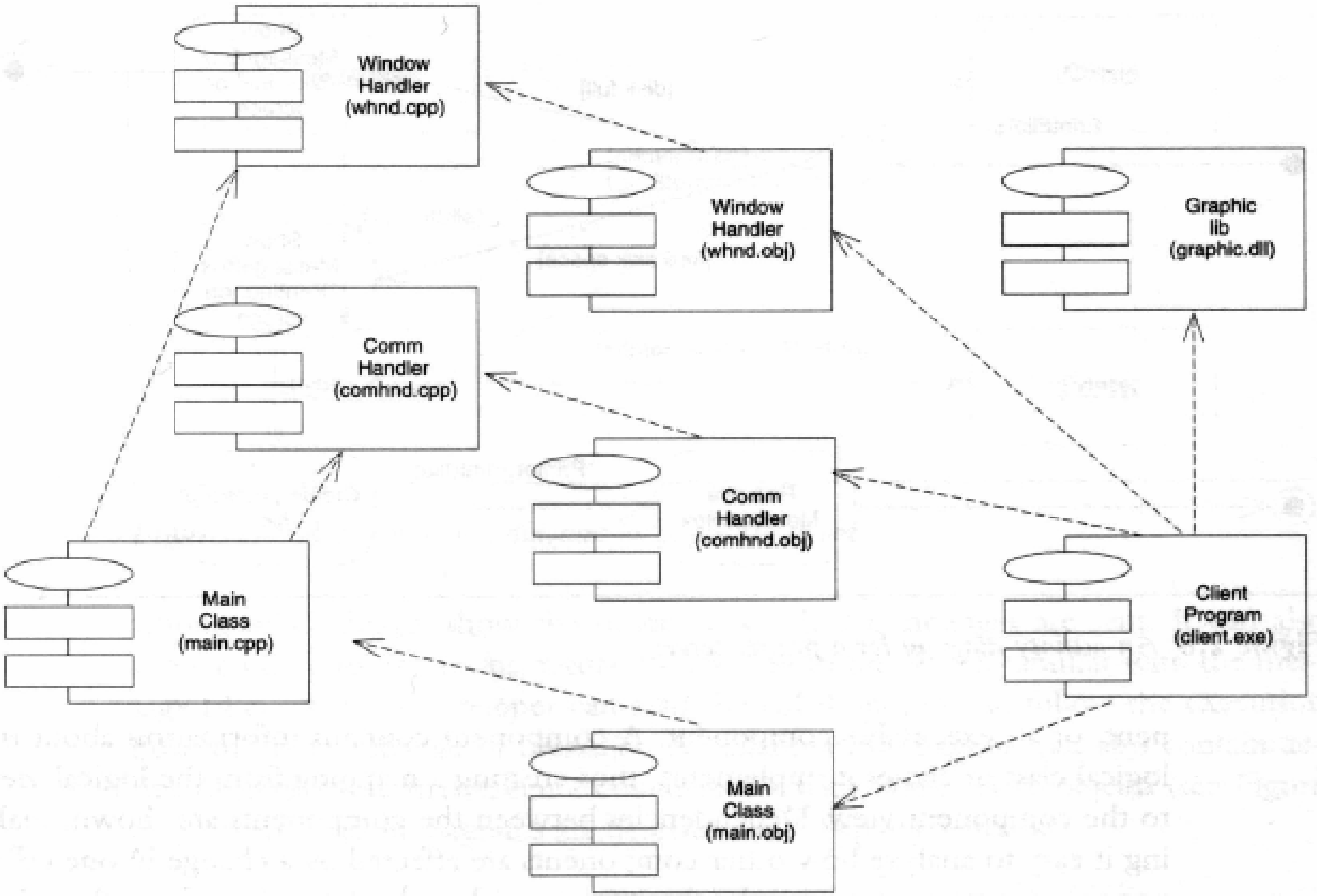
Show MessageBox "Printing" on screen

Create postscript file

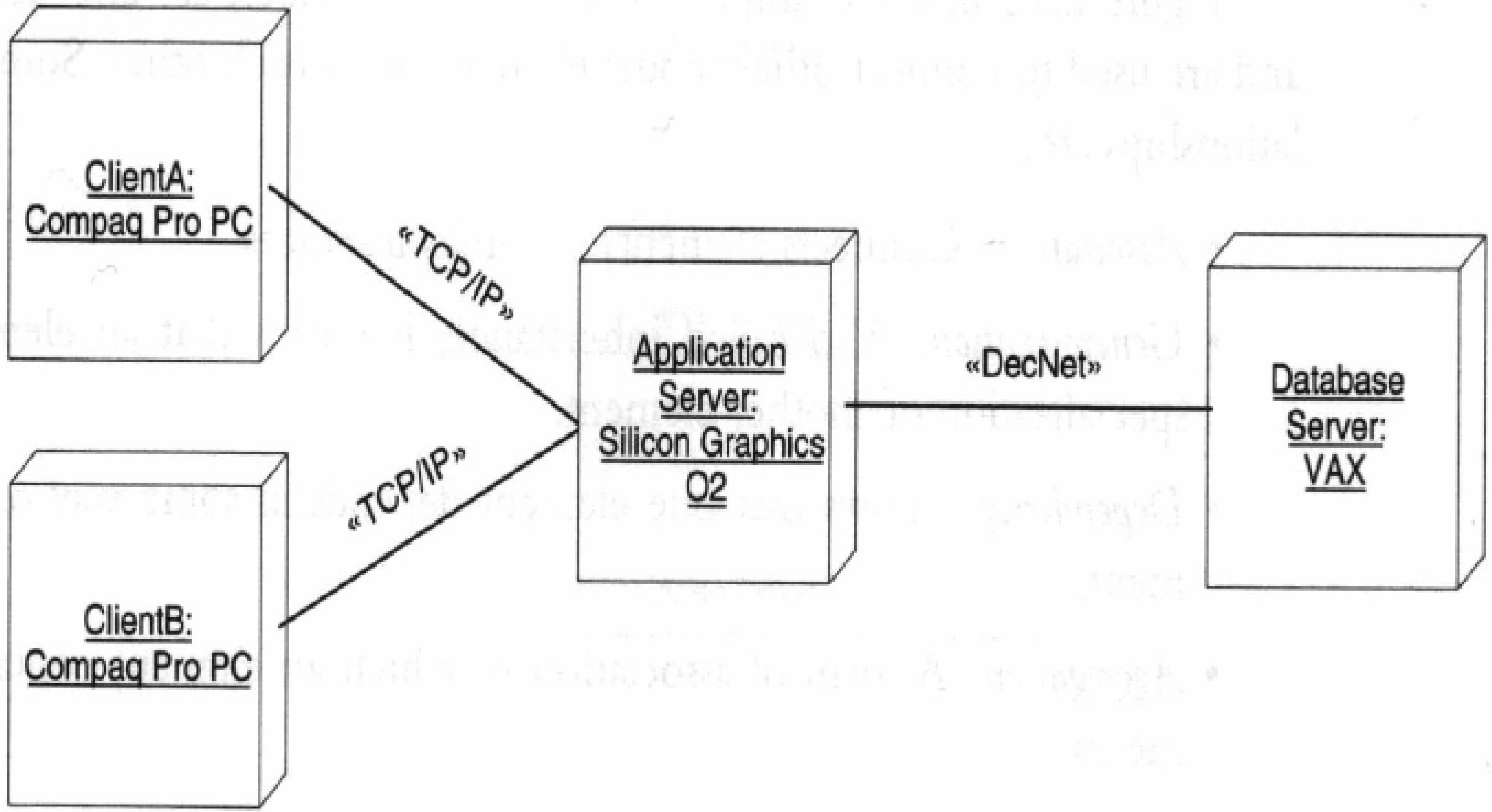
^Printer.Print(file)

Remove MessageBox

The Component Diagram


Window Handler (whnd.cpp)
Window Handler (whnd.obj)
Graphic lib (graphic.dll)
Comm Handler (comhnd.cpp)
Comm Handler (comhnd.obj)
Main Class (main.cpp)
Client Program (client.exe)
Main Class (main.obj)

The Deployment Diagram



Structural Concepts

- Actor
- Attribute
- Class
- Components
  - Include files
  - Header files
  - Link libraries
  - Modules
  - Executables
- Interface
- Object
- Package

Structural Concepts

- Activity
- Event
- Message
- Method
- Operation
- State
- Use case (goal)

Relationship Concepts

- Aggregation
- Association
- Composition
- Dependency
- Generalization
- Multiplicity
- Navigability
- Realization
- Stereotype