Introduction to OOAD and UML

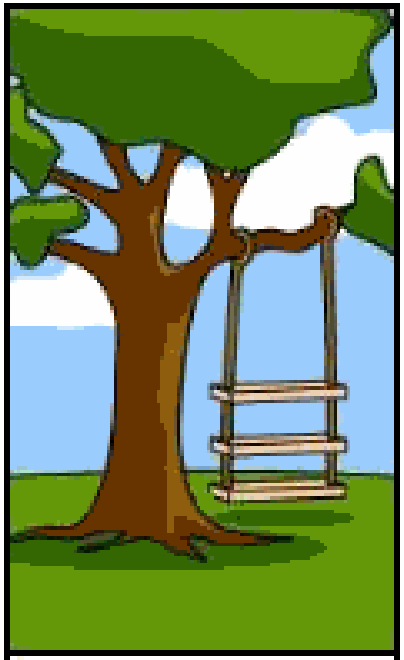
Rajeev Gupta
Java Trainer & consultant

Learning Objectives

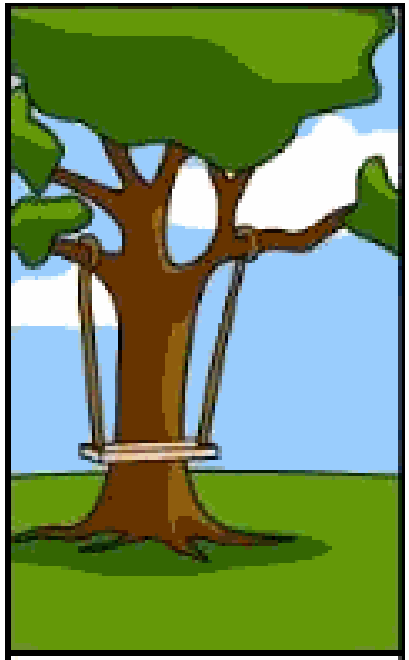
- Key terms
  - Association
  - Class diagram
  - Event
  - Object
  - Object class
  - Operation
  - Sequence diagram
  - State
  - State transition
  - Unified Modeling Language (UML)
  - Use case

The Object-Oriented Modeling Approach

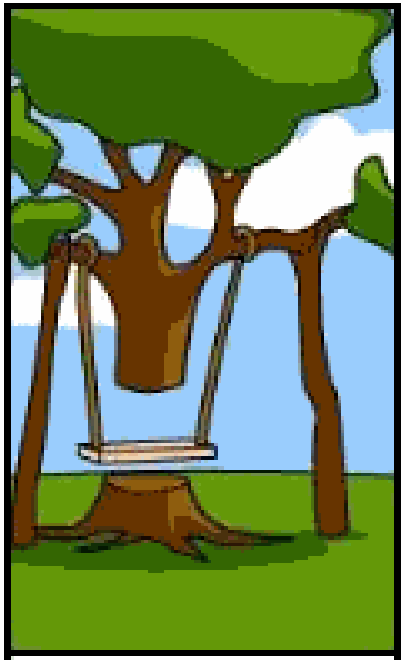
- Benefits
  1. The ability to tackle more challenging problem domains
  2. Improved communication among users, analysts, designers, and programmers
  3. Reusability of analysis, design, and programming results
  4. Increased consistency among the models developed during object-oriented analysis, design, and programming



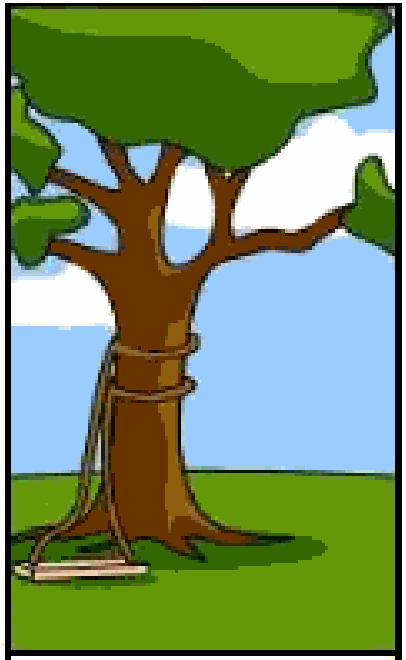
How the customer explained it



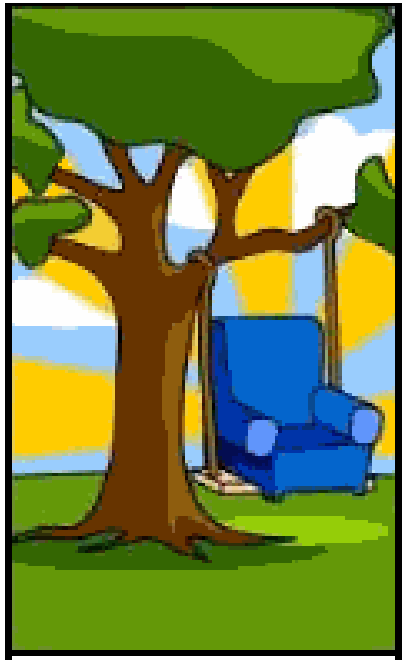
How the Project Leader understood it



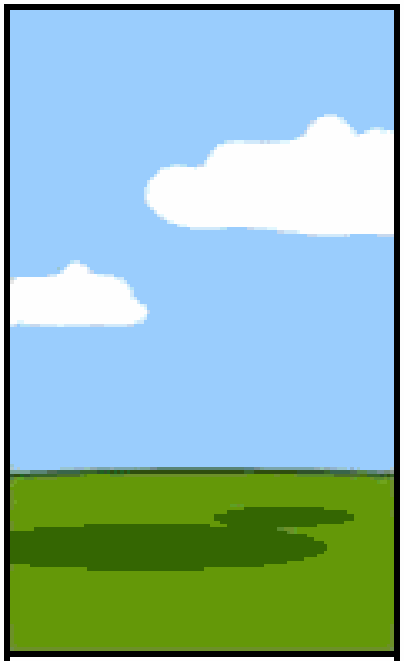
How the Analyst designed it



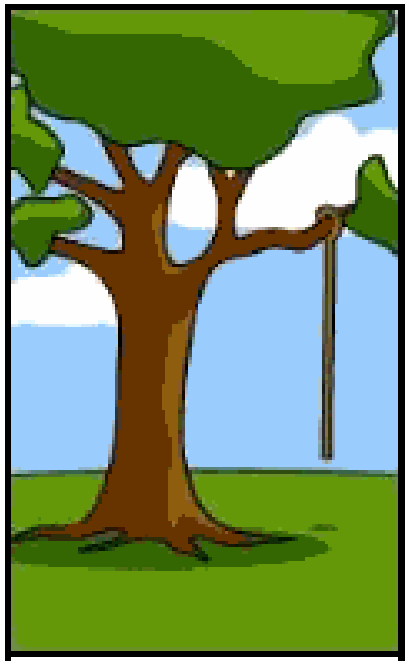
How the Programmer wrote it



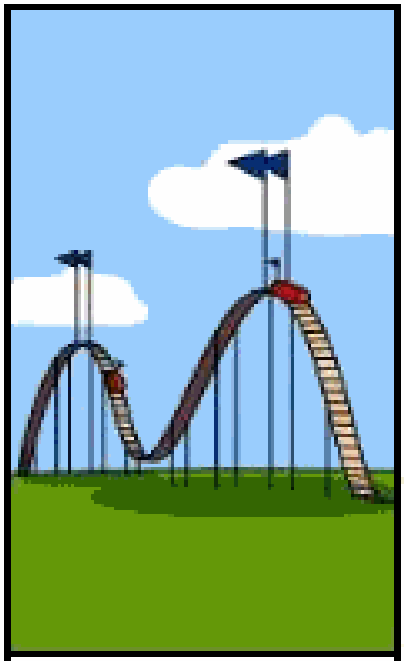
How the Business Consultant described it



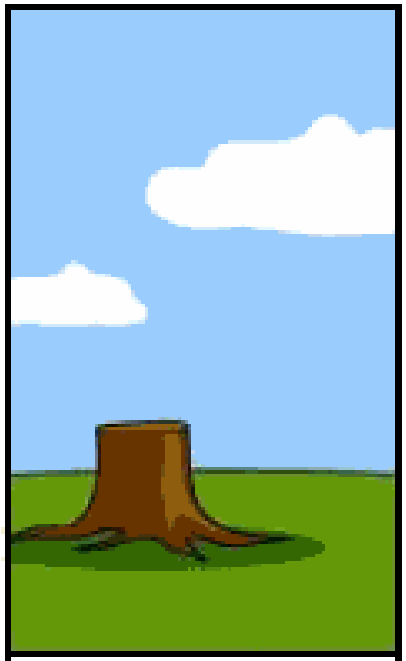
How the project was documented



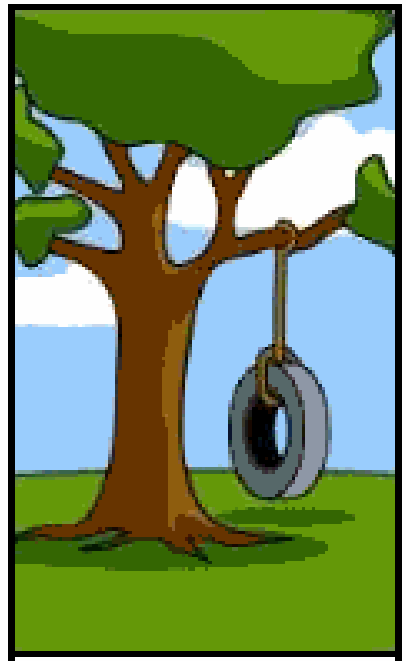
What operations installed



How the customer was billed



How it was supported



What the customer really needed

The Object-Oriented Modeling Approach (continued)

- Object-Oriented Systems Development Life Cycle
  - **Process of progressively developing representation of a system component (or object) through the phases of analysis, design, and implementation**
  - **The model is abstract in the early stages**
  - **As the model evolves, it becomes more and more detailed**

The Object-Oriented Systems Development Life Cycle

- Analysis Phase
  - Model of the real-world application is developed showing its important properties
  - Model specifies the functional behavior of the system independent of implementation details
- Design Phase
  - Analysis model is refined and adapted to the environment
- Implementation Phase
  - Design is implemented using a programming language or database management system

The Object-Oriented Systems Development Life Cycle (continued)

- Unified Modeling Language (UML)
  - **A notation that allows the modeler to specify, visualize and construct the artifacts of software systems, as well as business models**
  - **Techniques and notations**
    - **Use cases**
    - **Class diagrams**
    - **State diagrams**
    - **Sequence diagrams**

Use-Case Modeling

- Applied to analyze functional requirements of the system
- Performed during the analysis phase to help developers understand functional requirements of the system without regard for implementation details
- Use Case
  - A complete sequence of related actions initiated by an actor
- Actor
  - An external entity that interacts with the system

Use-Case Modeling

- Use cases represent complete functionality of the system
- Use cases may participate in relationships with other use cases
- Use cases may also use other use cases

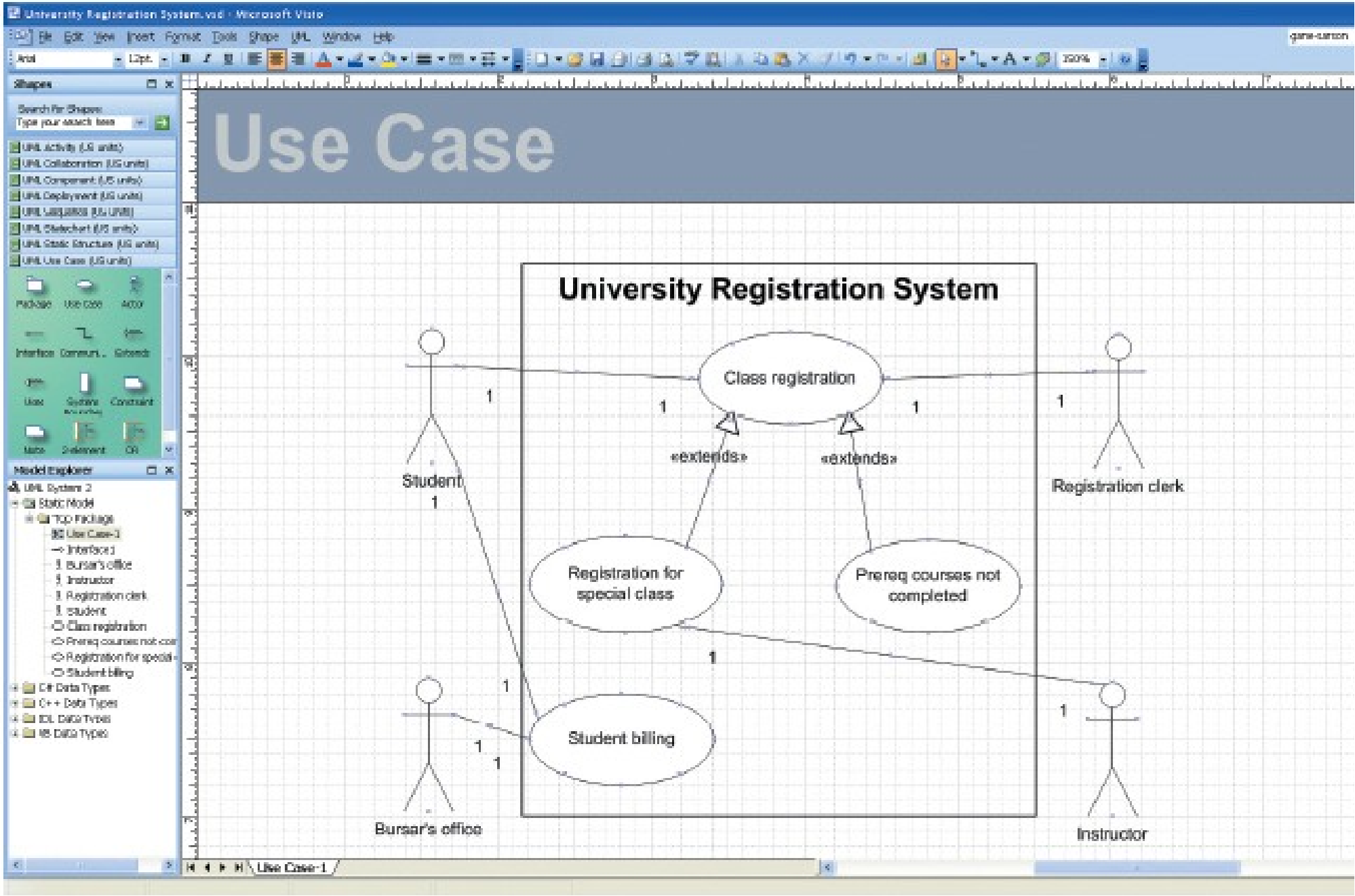


FIGURE A-1

USE-CASE DIAGRAM FOR A UNIVERSITY REGISTRATION SYSTEM DRAWN USING MICROSOFT VISIO

Object Modeling: Class Diagrams

- Object
  - An entity that has a well-defined role in the application domain, and has state, behavior, and identity
- State
  - A condition that encompasses an object's properties and the values those properties have
- Behavior
  - A manner that represents how an object acts and reacts
- Object Class
  - A set of objects that share a common structure and a common behavior

Object Modeling:
Class Diagrams (continued)

- Class Diagram
  - **Class is represented as a rectangle with three compartments**
  - **Objects can participate in relationships with objects of the same class**

Object Modeling: Object Diagrams

- Object Diagram
  - **A graph of instances that are compatible with a given class diagram; also called an instance diagram**
  - **Object is represented as a rectangle with two compartments**
- Operation
  - **A function or service that is provided by all the instances of a class**
- Encapsulation
  - **The technique of hiding the internal implementation details of an object from its external view**

Figure A.3a UML Class and Object Diagrams — Class Diagram Showing Two Classes

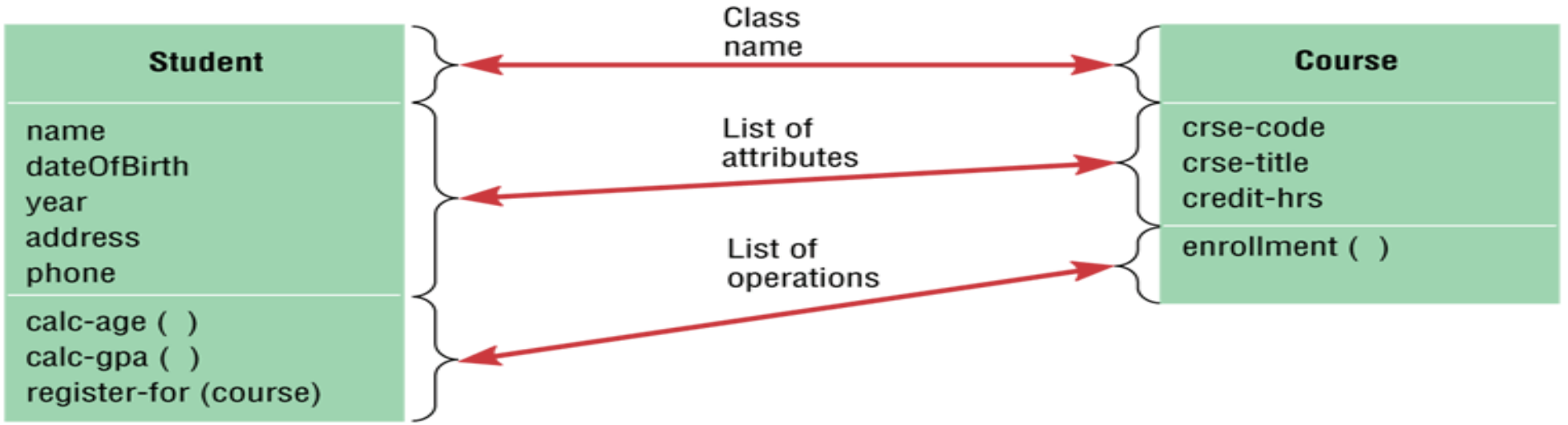


Figure A.3b UML Class and Object Diagrams — Object Diagram with Two Instances

Representing Associations

- Association
  - A relationship between object classes
  - Degree may be unary, binary, ternary or higher
  - Depicted as a solid line between participating classes
- Association Role
  - The end of an association where it connects to a class
  - Each role has multiplicity, which indicates how many objects participate in a given association relationship

Figure A.4a Examples of Association Relationships of Different Degrees — Unary

Figure A.4b Examples of Association Relationships of Different Degrees — Binary

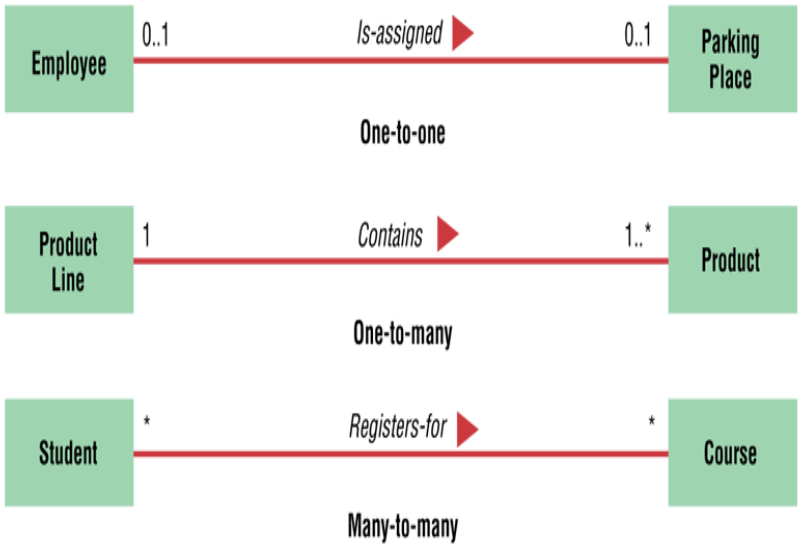
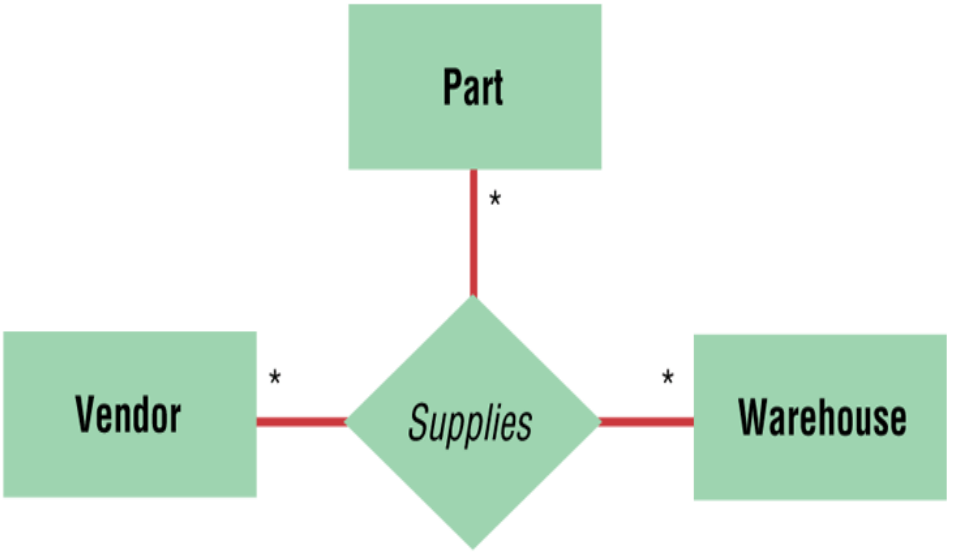


Figure A.4c Examples of Association Relationships of Different Degrees — Ternary



Representing Generalization

- Generalization
  - Abstraction of common features among multiple classes, as well as their relationships, into a more general class
- Subclass
  - A class that has been generalized
- Superclass
  - A class that is composed of several generalized subclasses

Representing Generalization (continued)

- Discriminator
  - Shows which property of an object class is being abstracted by a generalization relationship
- Inheritance
  - A property that a subclass inherits the features from its superclass
- Abstract Class
  - A class that has no direct instances but whose descendents may have direct instances
- Concrete Class
  - A class that can have direct instances

Figure A.6a Examples of Generalization, Inheritance, and Constraints — Employee Superclass with Three Subclasses

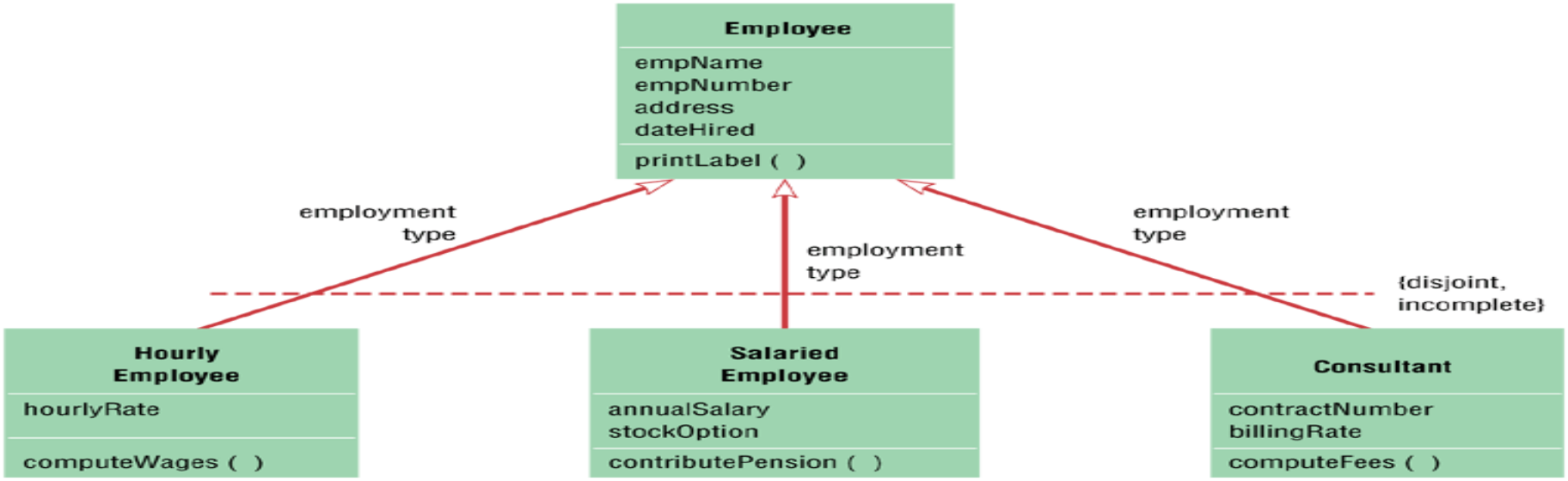
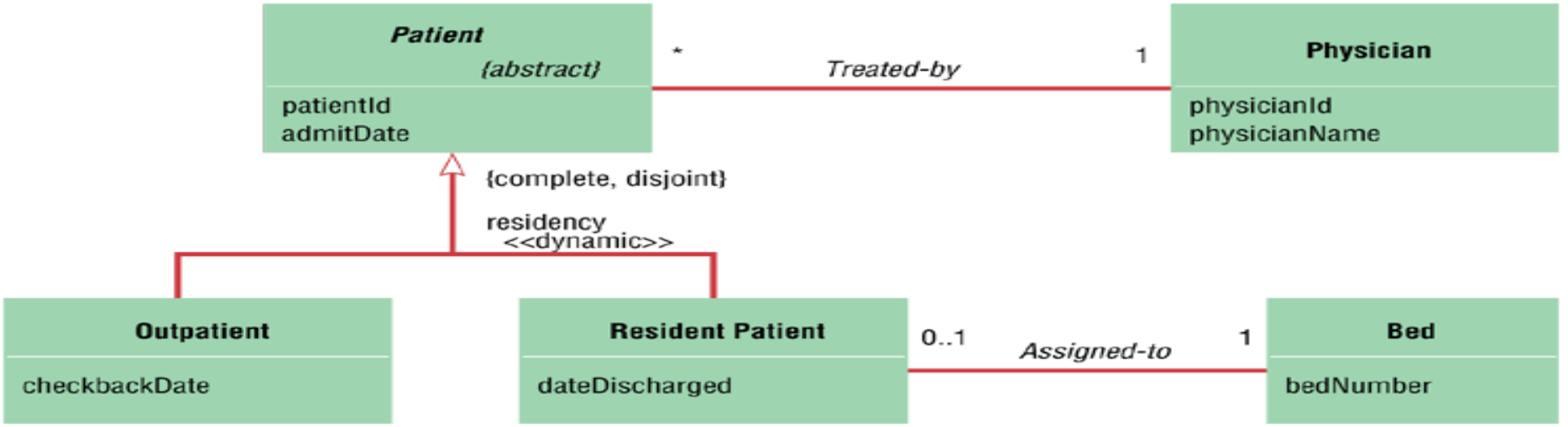


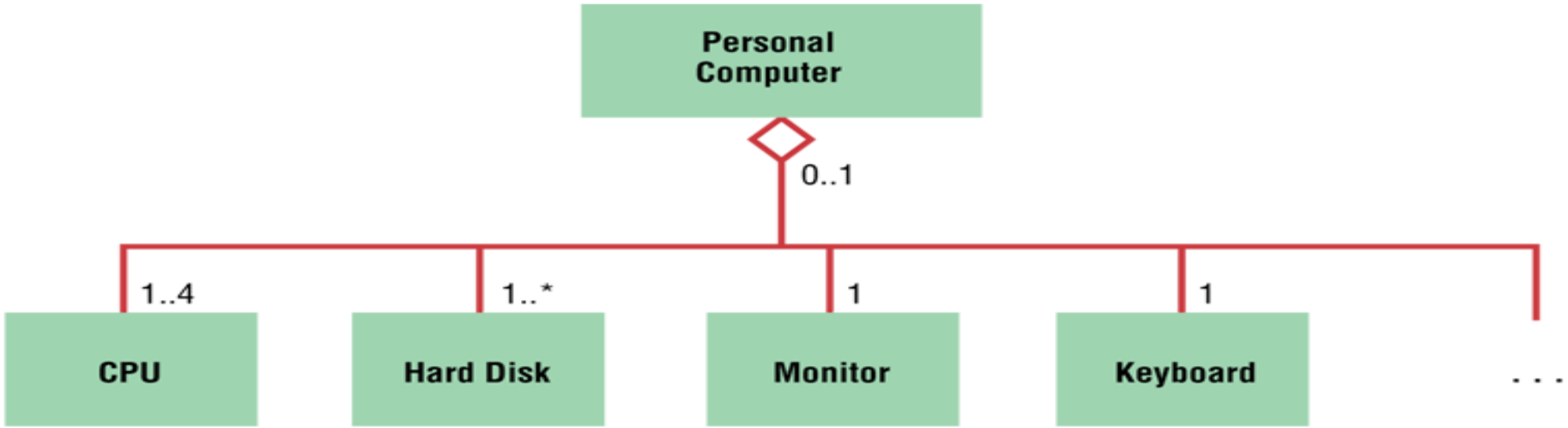
Figure A.6b Examples of Generalization, Inheritance, and Constraints — Abstract Patient Class with Two Concrete Subclasses



Representing Aggregation

- Aggregation
  - A part-of relationship between a component object and an aggregate object
  - Example: Personal computer
    - Composed of CPU, Monitor, Keyboard, etc

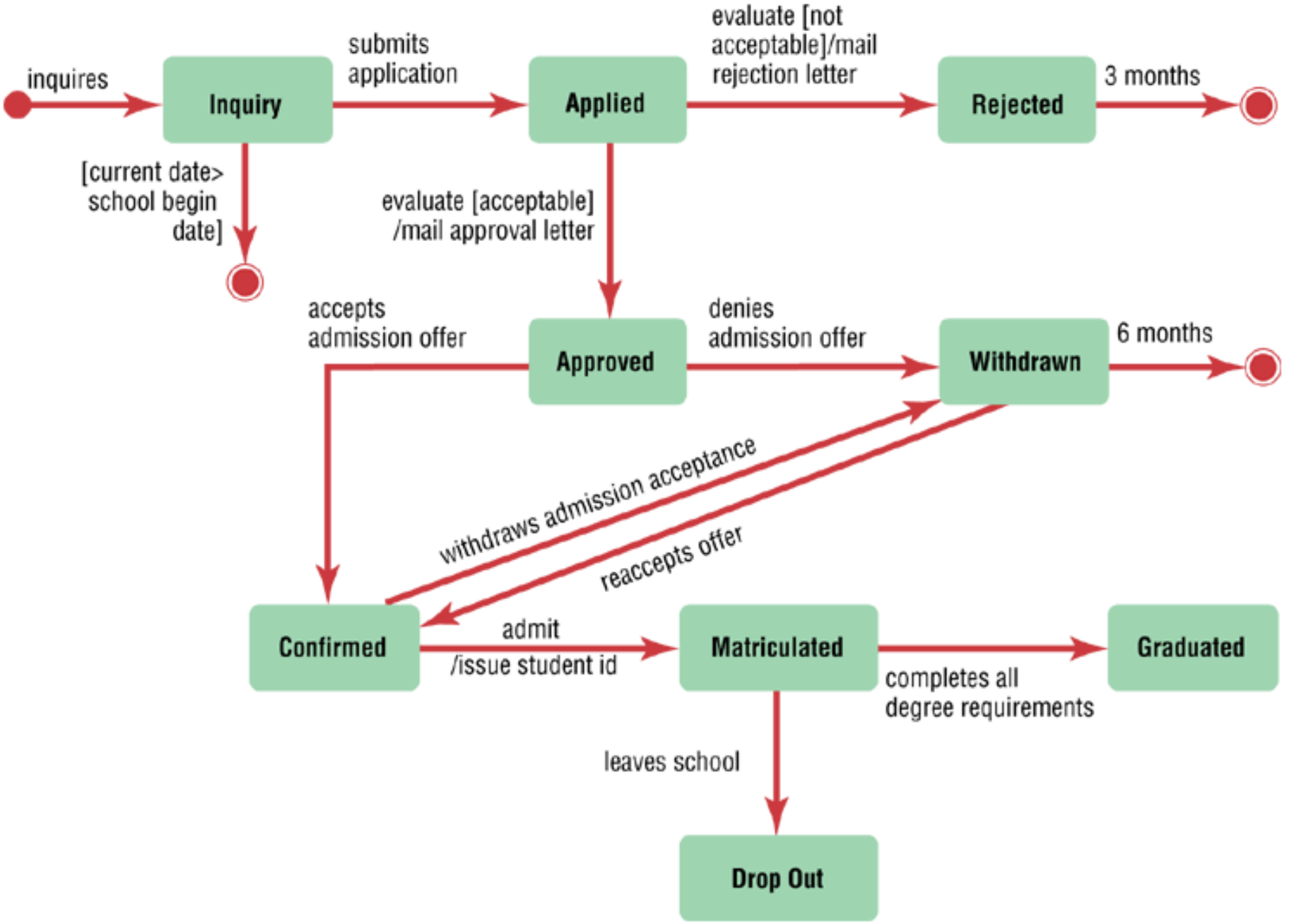
Figure A.7 Example of Aggregation



Dynamic Modeling: State Diagrams

- State
  - A condition during the life of an object during which it satisfies some conditions, performs some actions or waits for some events
  - Shown as a rectangle with rounded corners
- State Transition
  - The changes in the attributes of an object or in the links an object has with other objects
  - Shown as a solid arrow
  - Diagrammed with a guard condition and action
- Event
  - Something that takes place at a certain point in time

Figure A.8 State Diagram for the Student Object



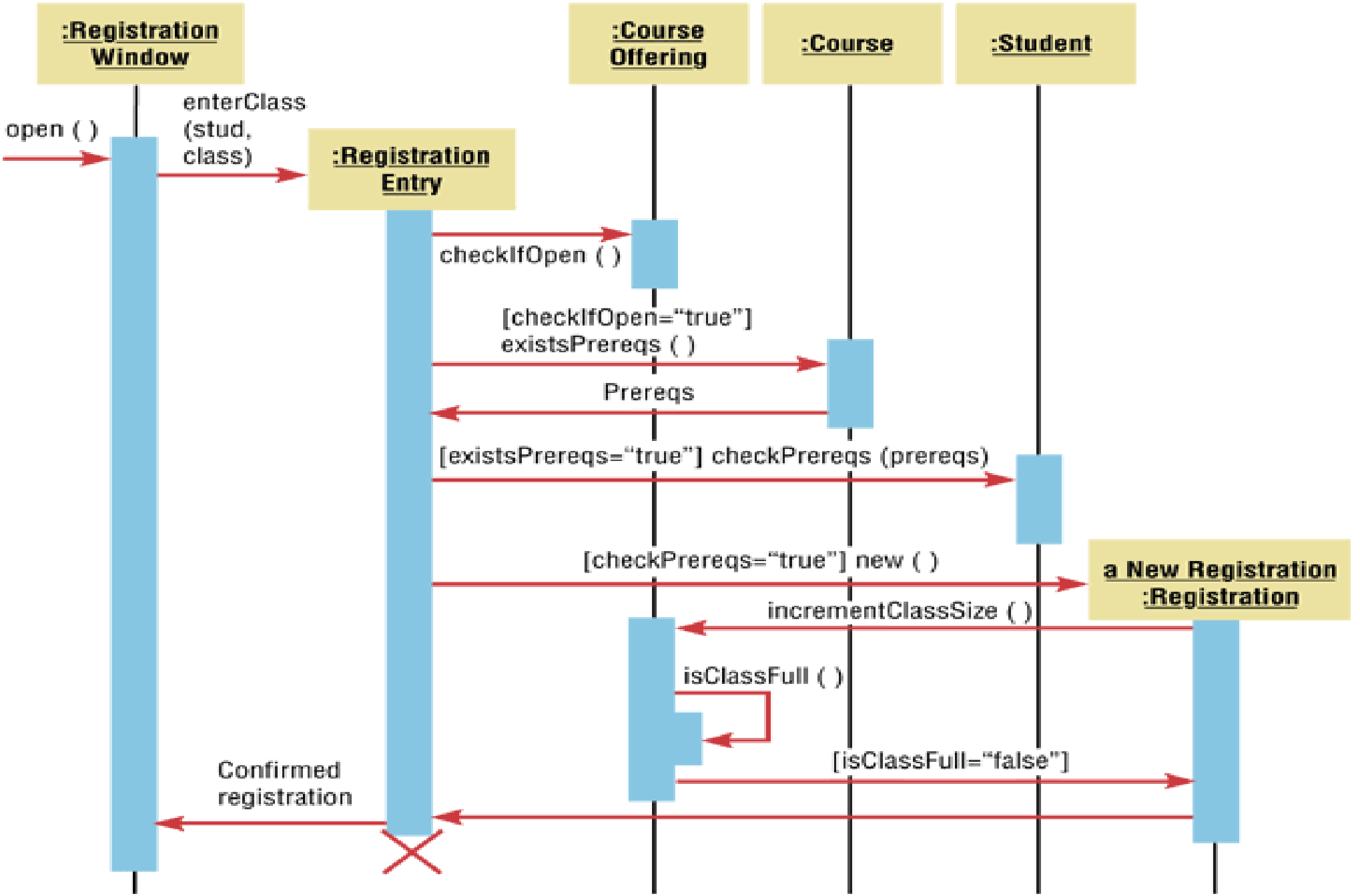
Dynamic Modeling: Sequence Diagrams

- Sequence Diagram
  - A depiction of the interaction among objects during certain periods of time
- Activation
  - The time period during which an object performs an operation
- Messages
  - Means by which objects communicate with each other

Dynamic Modeling: Sequence Diagrams (continued)

- Synchronous Message
  - A type of message in which the caller has to wait for the receiving object to finish executing the called operation before it can resume execution itself
- Simple Message
  - A message that transfers control from the sender to the recipient without describing the details of the communication

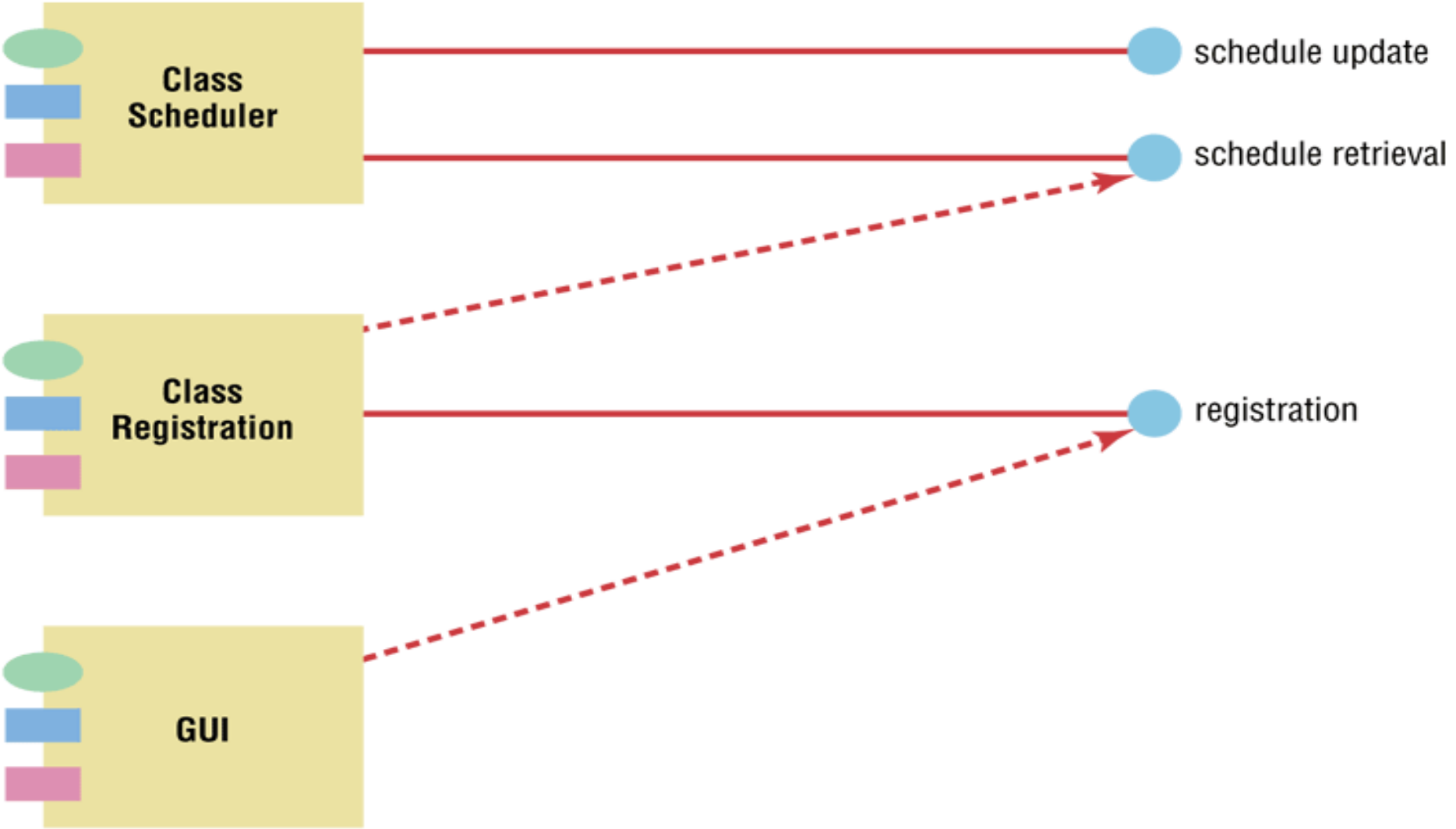
Figure A.9 Sequence Diagram for a Class Registration Scenario with Prerequisites



Moving to Design

- Start with existing set of analysis model
- Progressively add technical details
- Design model must be more detailed than analysis model
- Component Diagram
  - A diagram that shows the software components or modules and their dependencies
- Deployment Diagram
  - A diagram that shows how the software components, processes and objects are deployed into the physical architecture of the system

Figure A.11 A Component Diagram for Class Registration



Summary

- Object-Oriented Modeling Approach
  - Benefits
  - Unified Modeling Language
    - Use cases
    - Class diagrams
    - State diagrams
    - Sequence diagrams
- Use Case Modeling

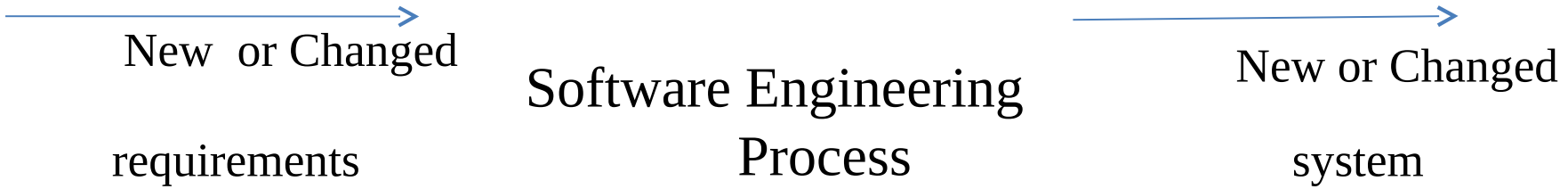
Summary (continued)

- Object Modeling: Class Diagrams
  - Associations
  - Generalizations
  - Aggregation
- Dynamic Modeling: State Diagrams
- Dynamic Modeling: Sequence Diagrams
- Moving to Design

Unified Process

What is Process ???

- Defines *who is doing, what and when to do it,* and *how* to reach a certain goal.



What is Unified Process ??

- Unified process (UP) is an architecture-centric, use-case driven, iterative and incremental development process that leverages unified modeling language and is compliant with the system process engineering metamodel.

- A popular iterative modern process model (framework) derived from the work on the UML and associated process.

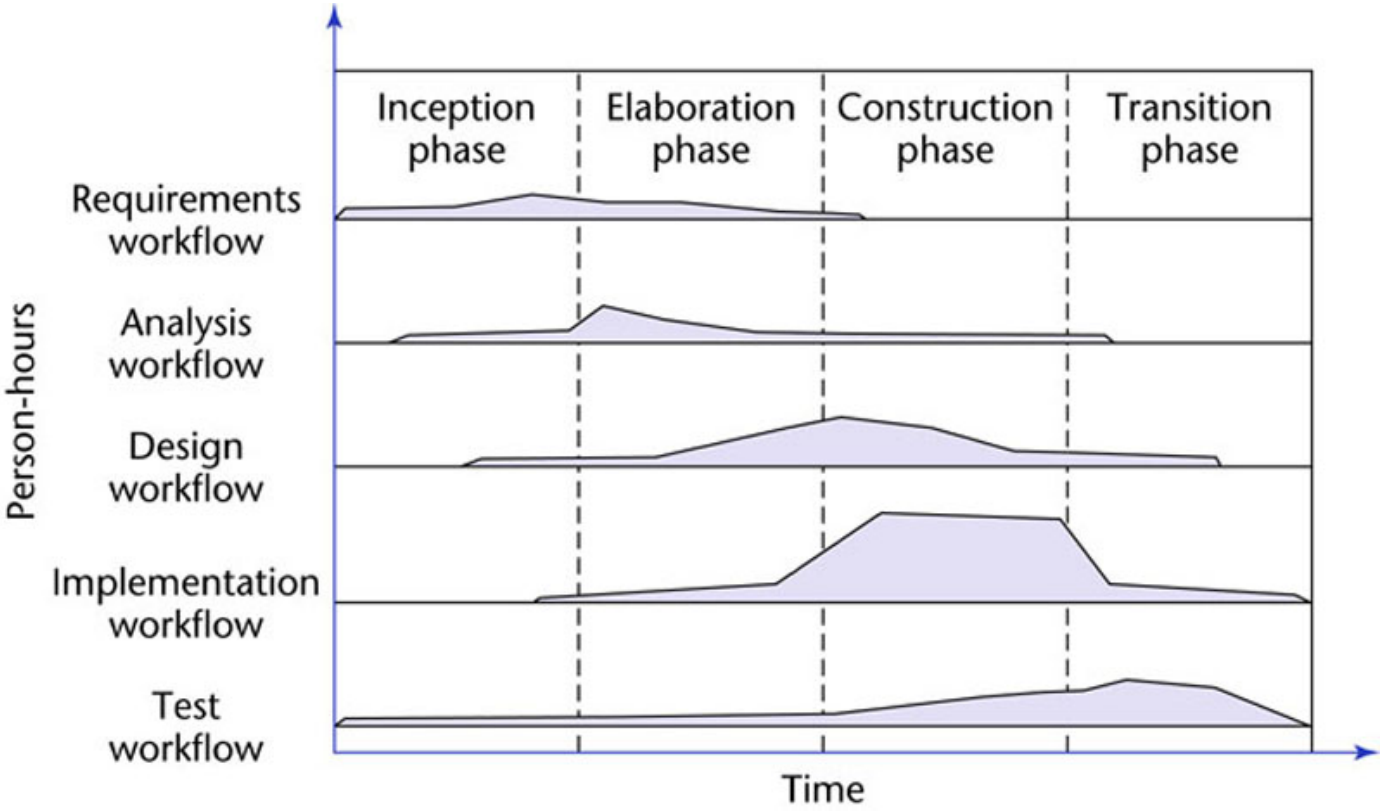
Unified Process Phases

- Inception
  - Establish the business case for the system, define risks, obtain 10% of the requirements, estimate next phase effort.
- Elaboration
  - Develop an understanding of the problem domain and the system architecture, risk significant portions may be coded/tested, 80% major requirements identified.
- Construction
  - System design, programming and testing. Building the remaining system in short iterations.
- Transition
  - Starts when beta testing is completed, Deploy the system in its operating environment. Deliver releases for feedback and deployment

The Phases/Workflows Of Unified Process

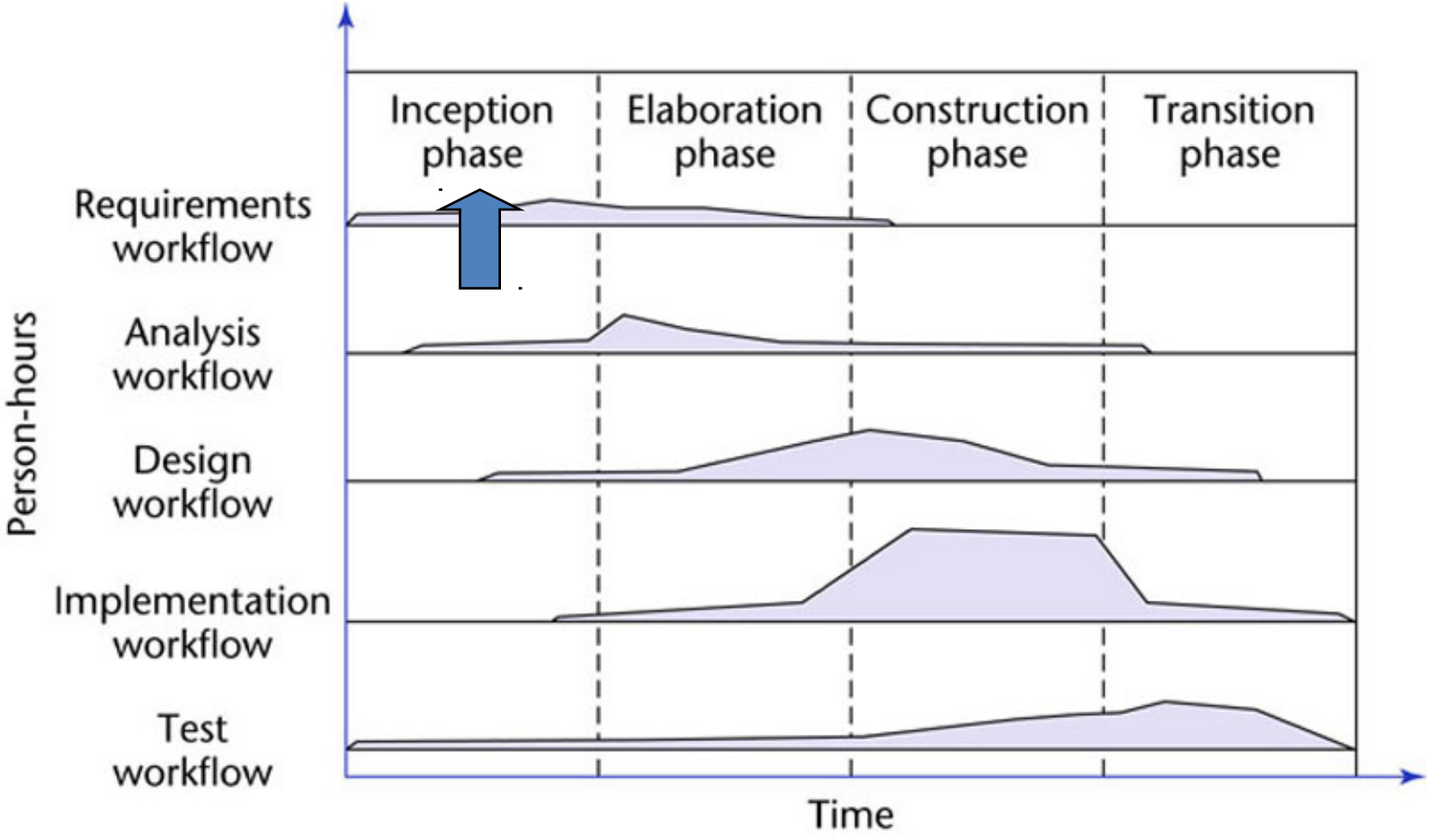
- **Phase** is Business context of a step

Workflow is Technical context of a step



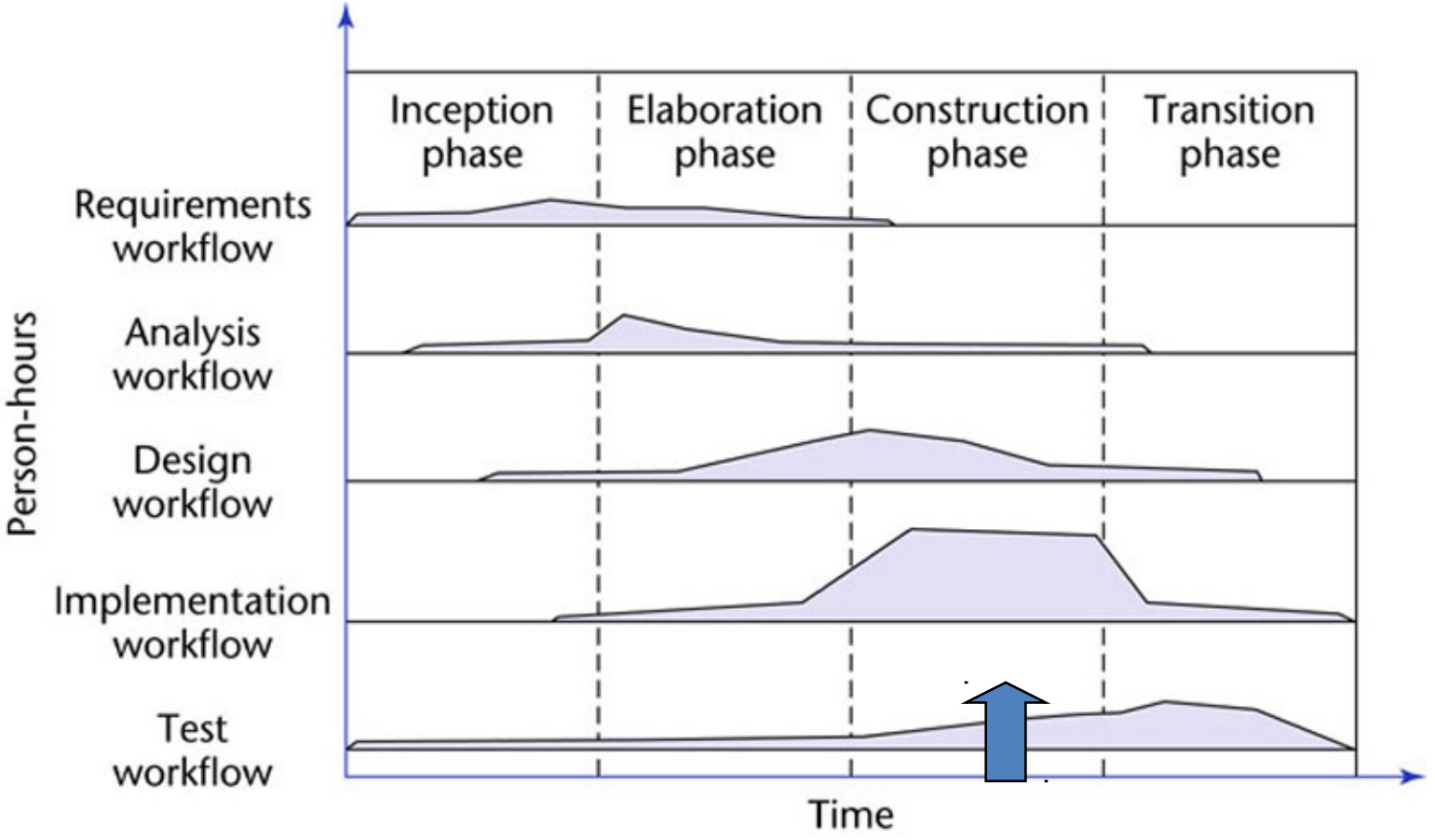
The Phases/Workflows Of Unified Process

- **NOTE:** Most of the requirements work or workflow is done in the inception phase.
- However some is done later.



The Phases/Workflows Of Unified Process

- NOTE: Most of the implementation work or workflow is done in construction
- However some is done earlier and some later.



Example roles in UP

- *Stake Holder:* customer, product manager, etc.
- *Software Architect:* established and maintains architectural vision
- *Process Engineer:* leads definition and refinement of Development Case
- *Graphic Artist:* assists in user interface design, etc.

UML Diagrams

What is UML?

- Standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, business modeling and other non-software systems.
- The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

- The UML is a very important part of developing object oriented software and the software development process.

- The UML uses mostly graphical notations to express the design of software projects.

- Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

Overview of UML Diagrams

Structural

: element of spec. irrespective of time

- Class
- Component
- Deployment
- Object
- *Composite structure*
- *Package*

Behavioral

: behavioral features of a system / business process

- Activity
- State machine
- Use case
- *Interaction*

Interaction

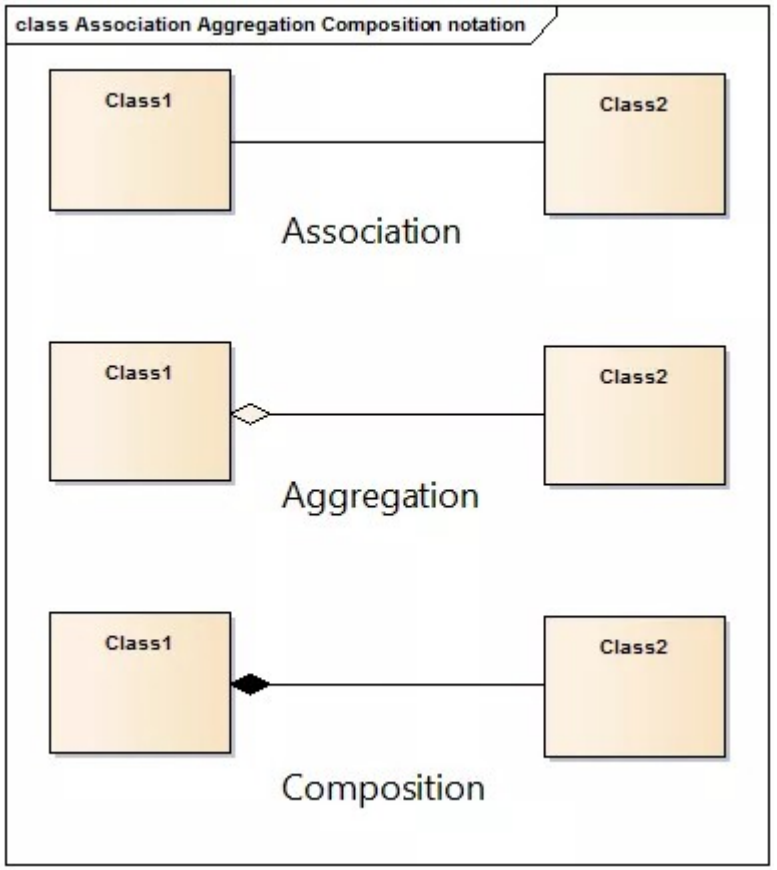
: emphasize object interaction

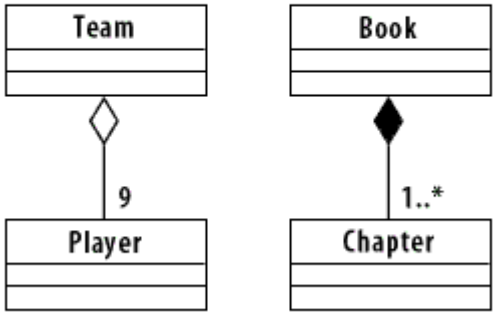
- Communication(collaboration)
- Sequence
- *Interaction overview*
- *Timing*

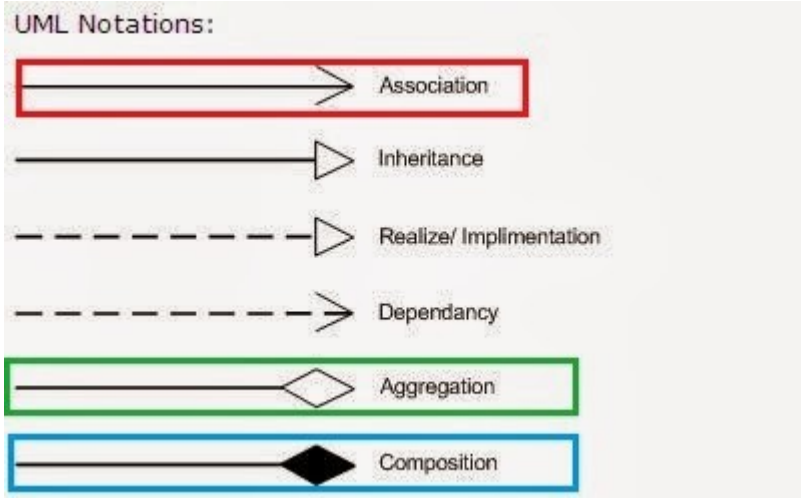
Class diagram

UML class diagrams show the classes of the system, their inter-relationships, and the operations and attributes of the classes

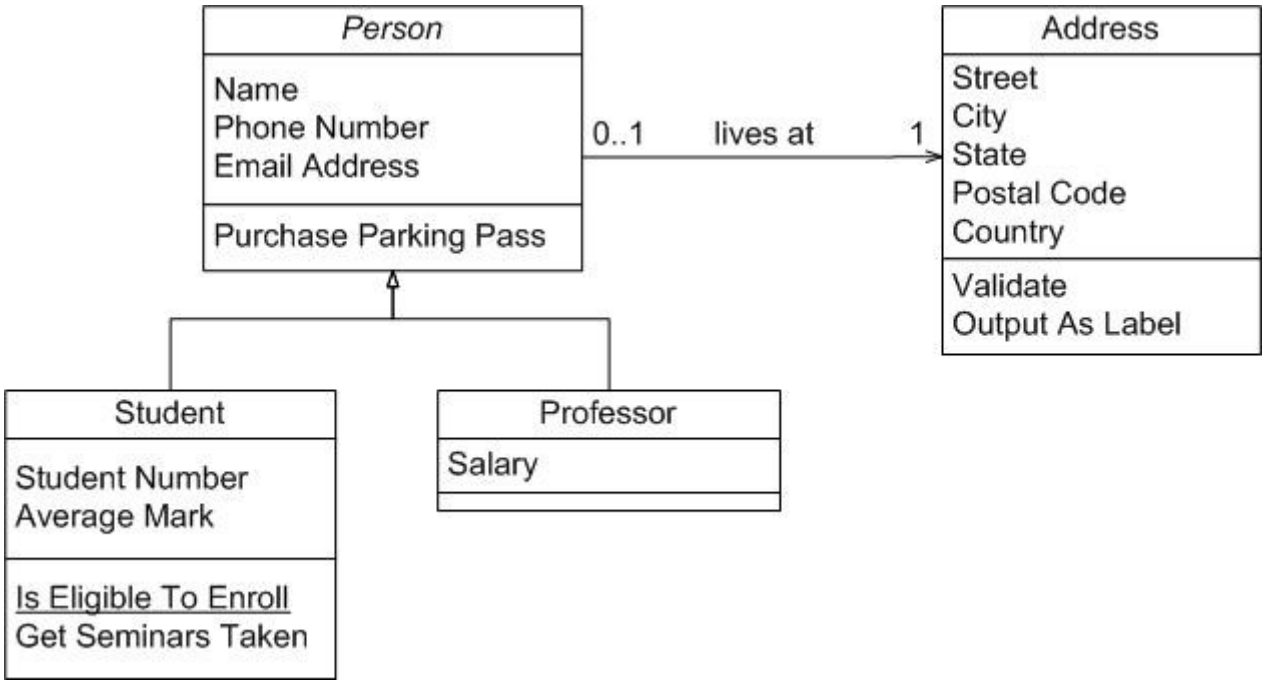
- ✓ Explore domain concepts in the form of a domain model
- ✓ Analyze requirements in the form of a conceptual/analysis model
- ✓ Depict the detailed design of object-oriented or object-based software


class Association Aggregation Composition notation
Class1
Class2
Association
Class1
Class2
Aggregation
Class1
Class2
Composition

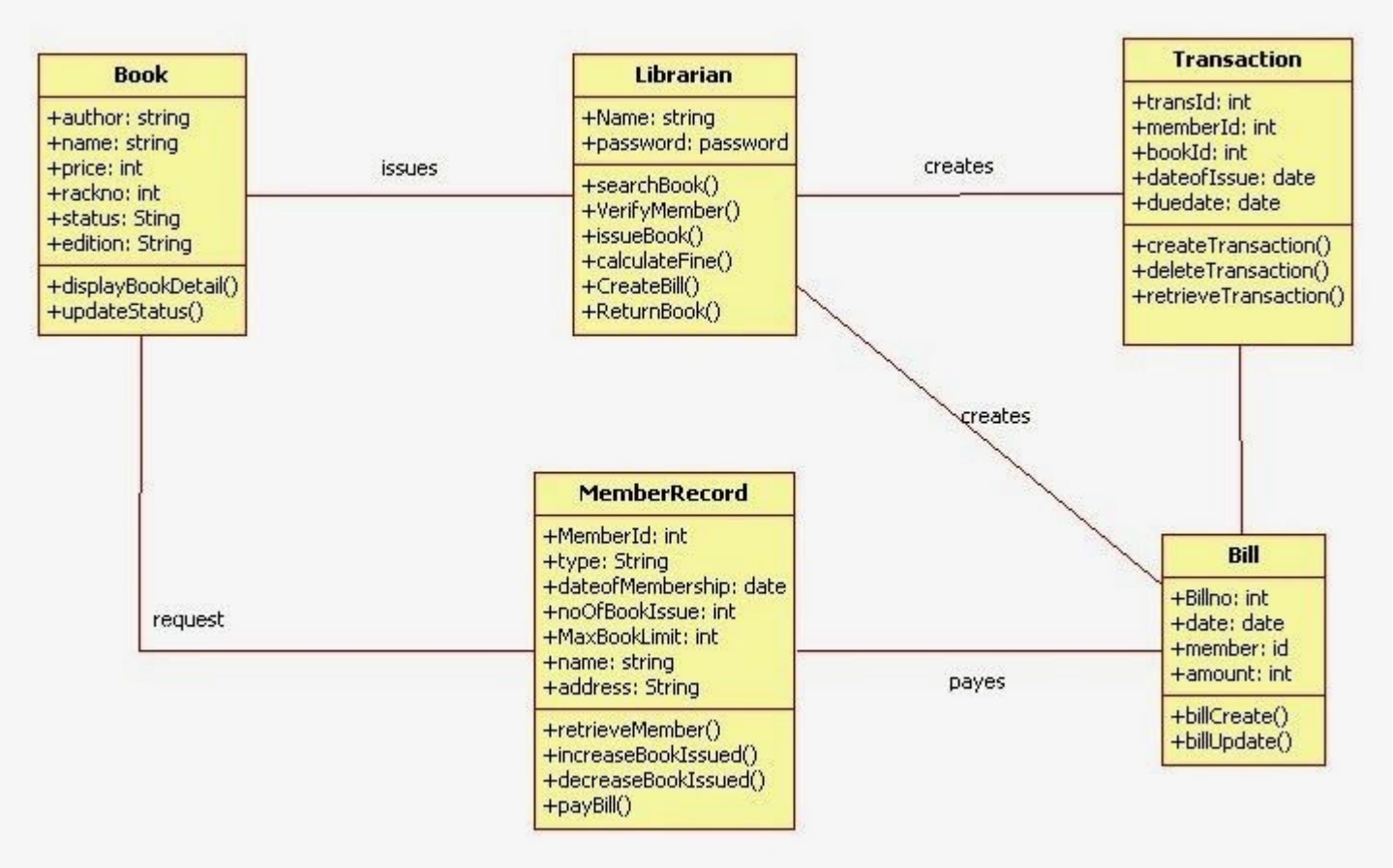

Aggregation and Composition
Team
Book
9
1..*
Player
Chapter


UML Notations:
Association
Inheritance
Realize/ Implimentation
Dependancy
Aggregation
Composition

Class diagram



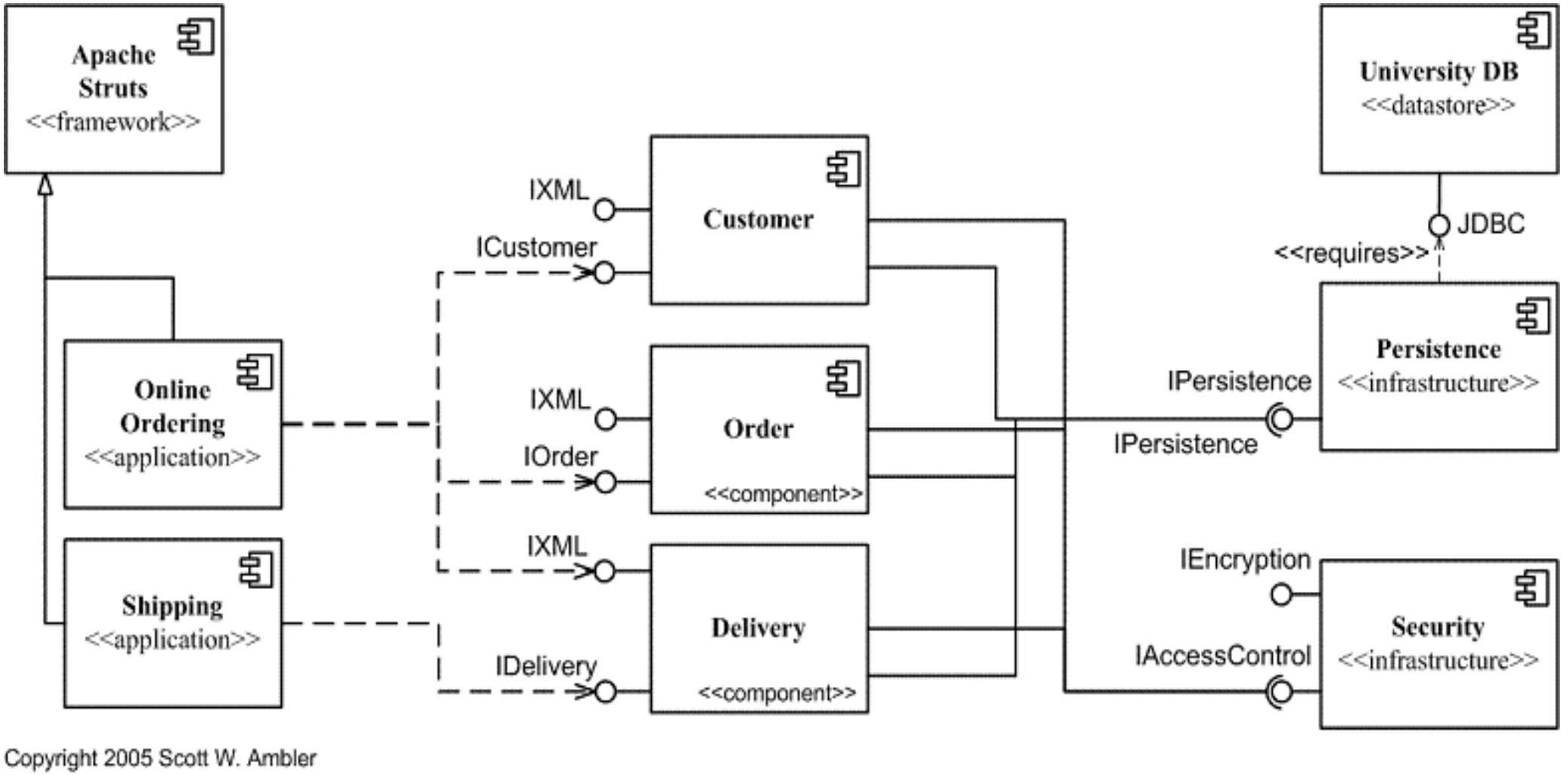
Class diagram



Component diagram

- ✓ UML component diagrams shows the dependencies among software components, including the classifiers that specify them (for example implementation classes) and the artifacts that implement them; such as source code files, binary code files, executable files, scripts and tables.

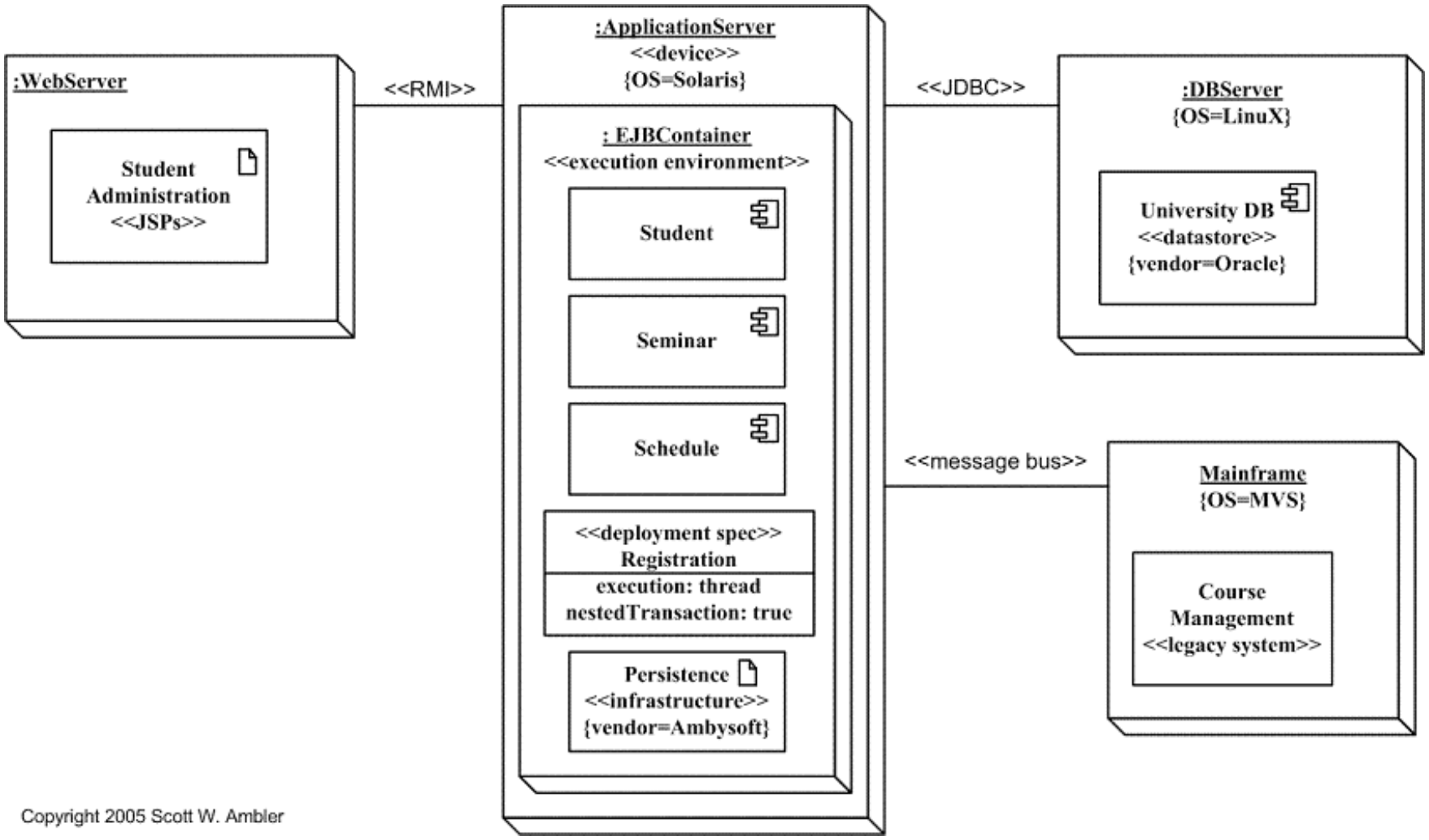
Component diagram



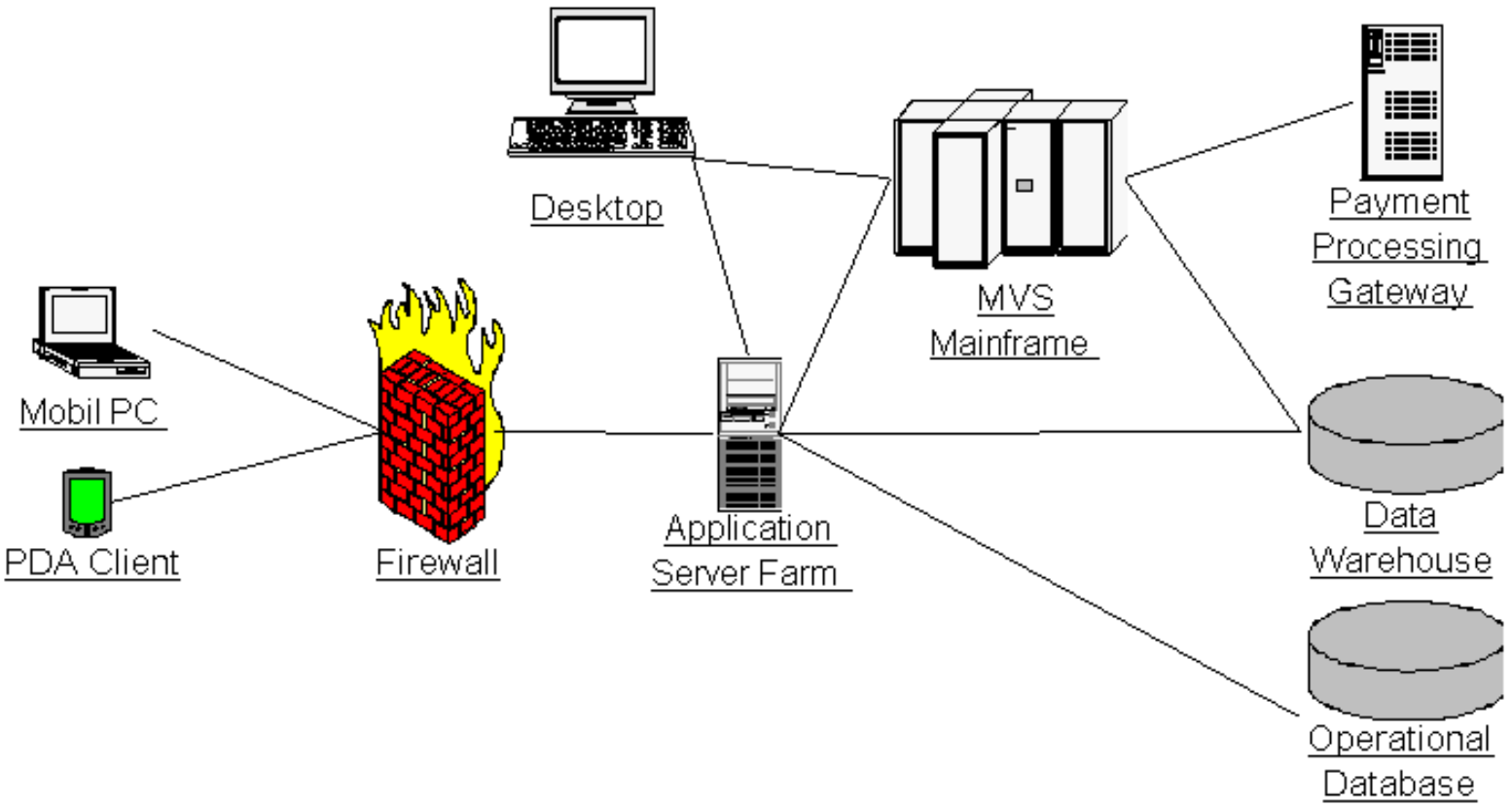
Deployment diagram

- ✓ UML deployment diagram depicts a static view of the run-time configuration of hardware nodes and the software components that run on those nodes. Deployment diagrams show the hardware for your system, the software that is installed on that hardware, and the middleware used to connect the disparate machines to one another.

Deployment diagram



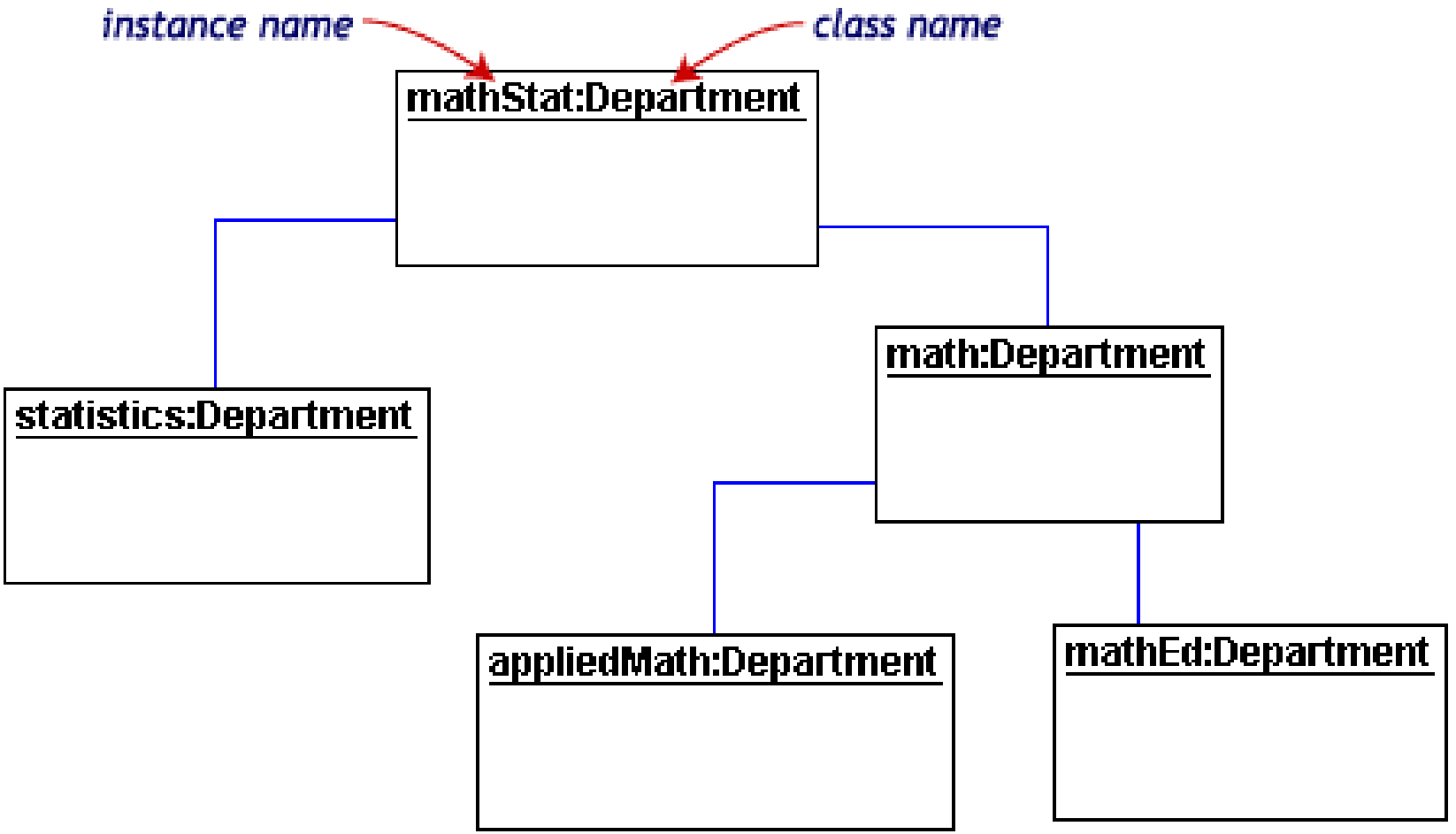
Deployment diagram



Object diagram

- ✓ UML Object diagrams (instance diagrams), are useful for exploring real world examples of objects and the relationships between them. It shows instances instead of classes. They are useful for explaining small pieces with complicated relationships, especially recursive relationships.

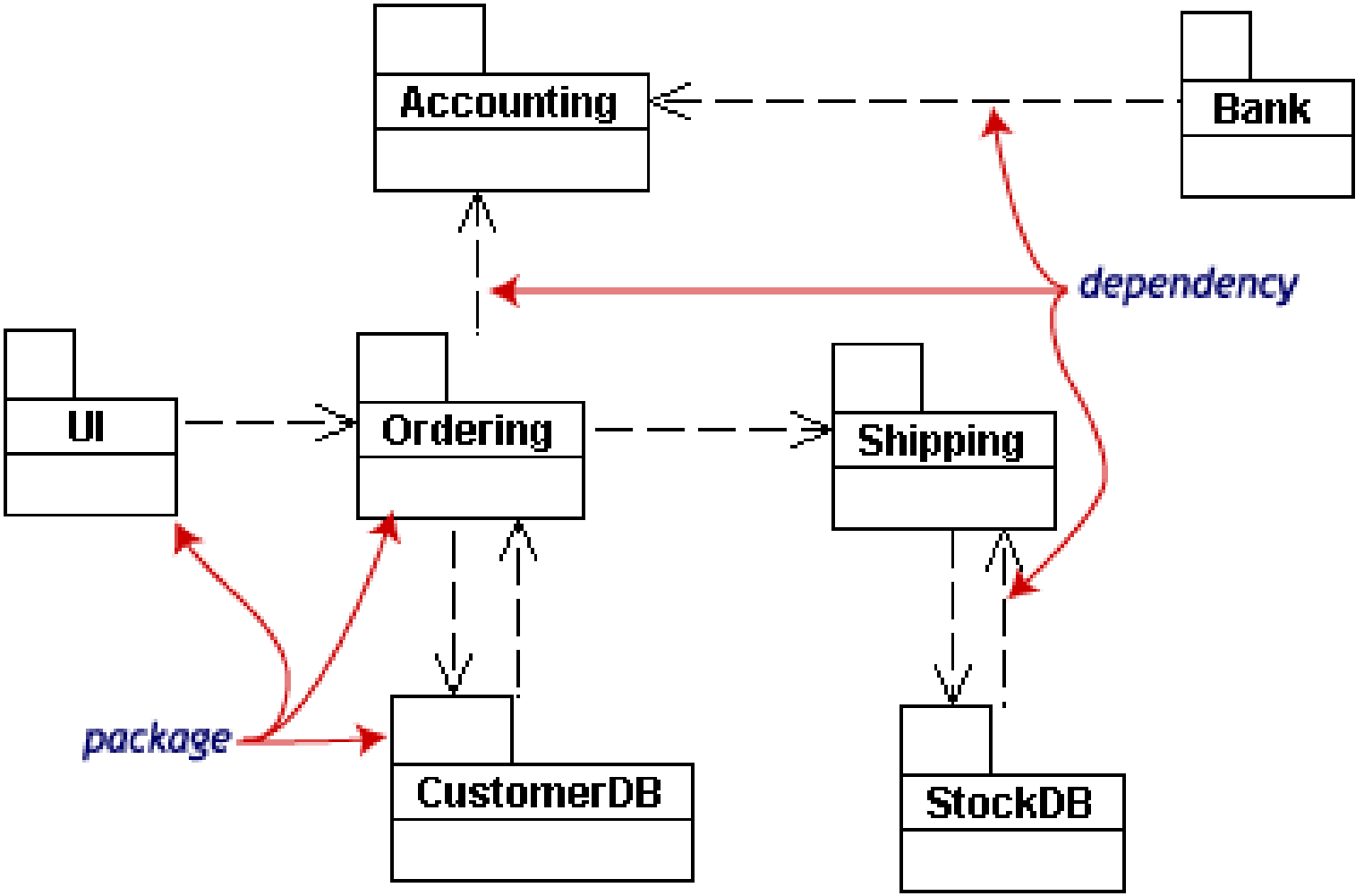
Object diagram



Package diagram

- ✓ UML Package diagrams simplify complex class diagrams, it can group classes into **packages**. A package is a collection of logically related UML elements. Packages are depicted as file folders and can be used on any of the UML diagrams.

Package diagram



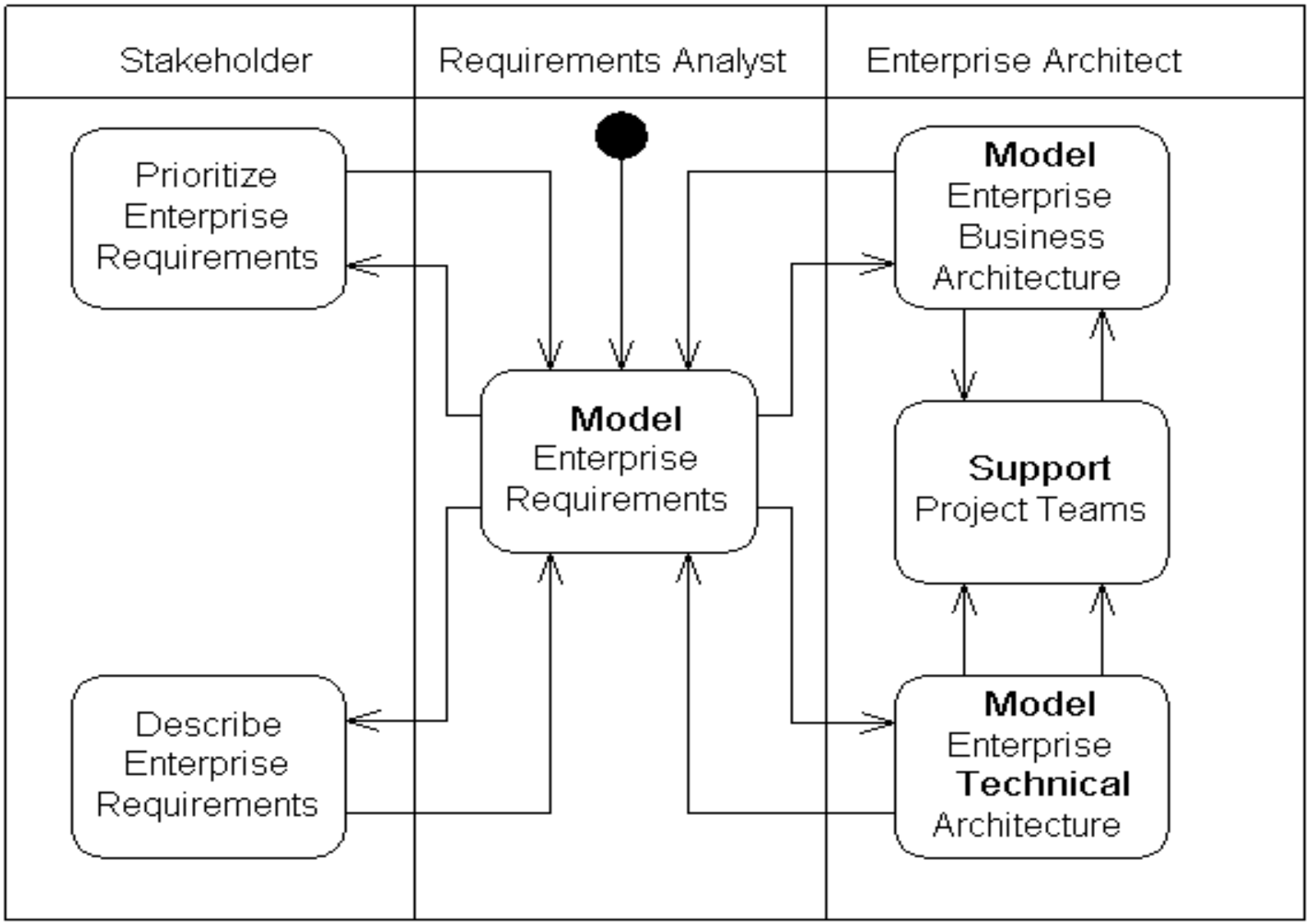
Composite structure diagram

- ✓ UML Composite structure diagrams used to explore run-time instances of interconnected instances collaborating over communications links.
- ✓ It shows the internal structure (including parts and connectors) of a structured classifier or collaboration.

Activity diagram

- ✓ UML Activity diagrams helps to describe the flow of control of the target system.
- ✓ such as the exploring complex business rules and operations, describing the use case also the business process.
- ✓ It is object-oriented equivalent of flow charts and data-flow diagrams (DFDs).

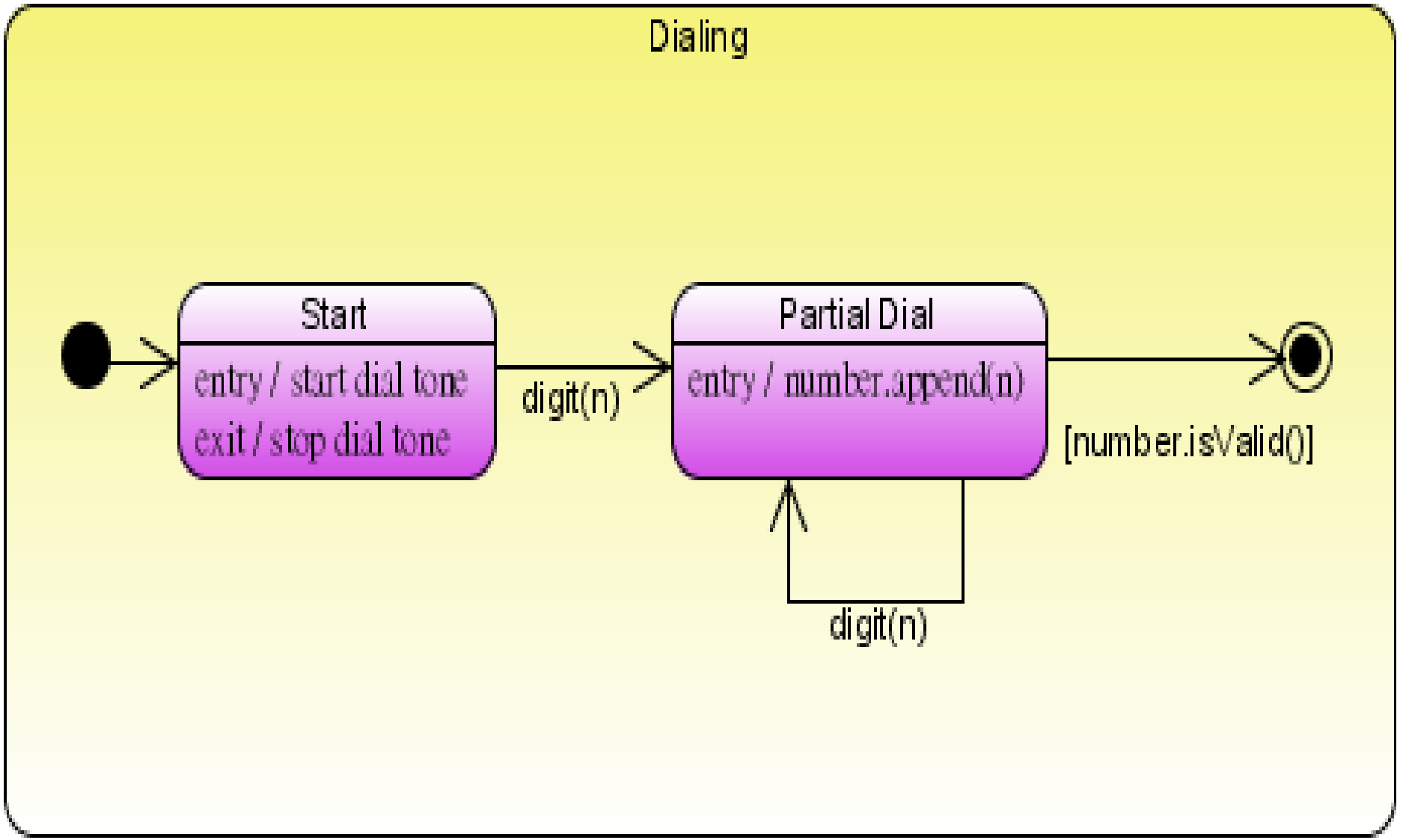
Activity diagram



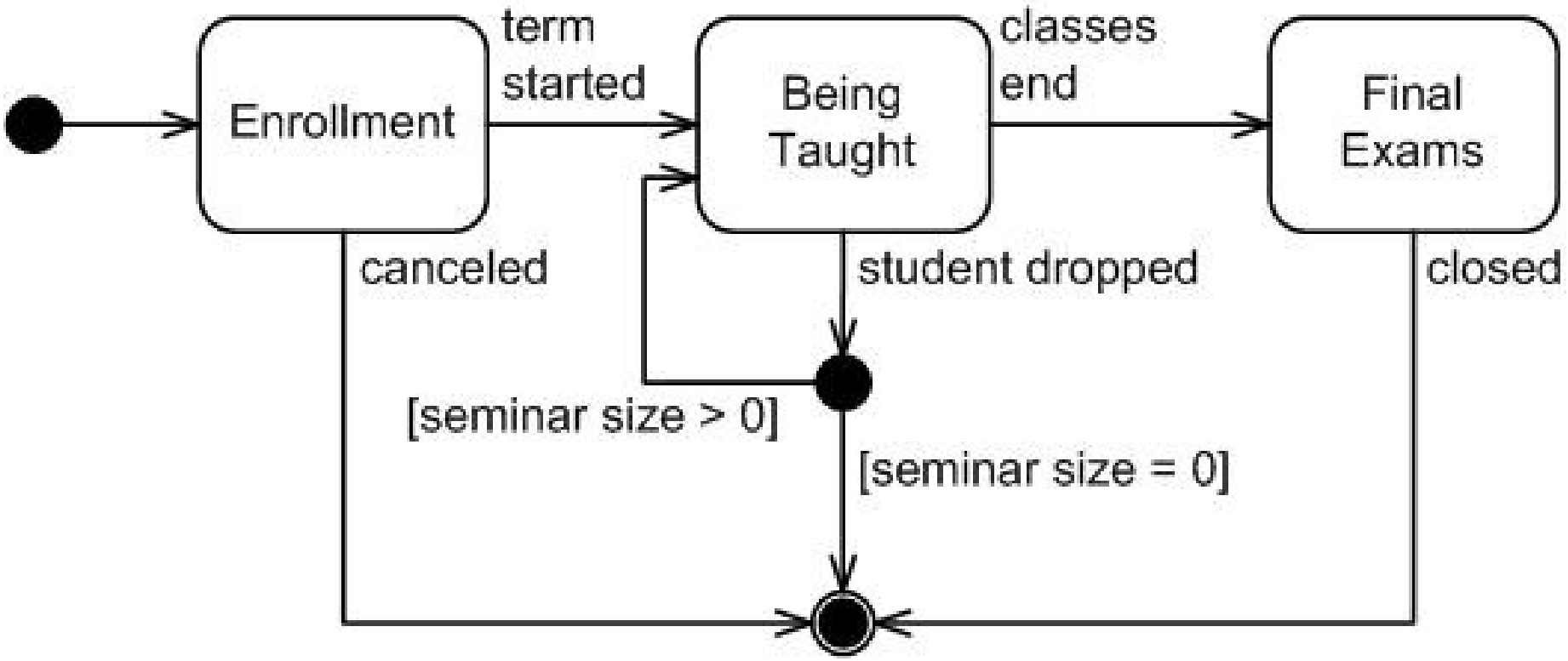
State diagram

- ✓ UML State diagrams can show the different states of an entity also how an entity responds to various events by changing from one state to another.
- ✓ The history of an entity can best be modeled by a finite state diagram.

State diagram



State diagram



Use cases diagram

Use cases diagrams describes the behavior of the target system from an external point of view. Use cases describe "the meat" of the actual requirements.

- ***Use cases:*** A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

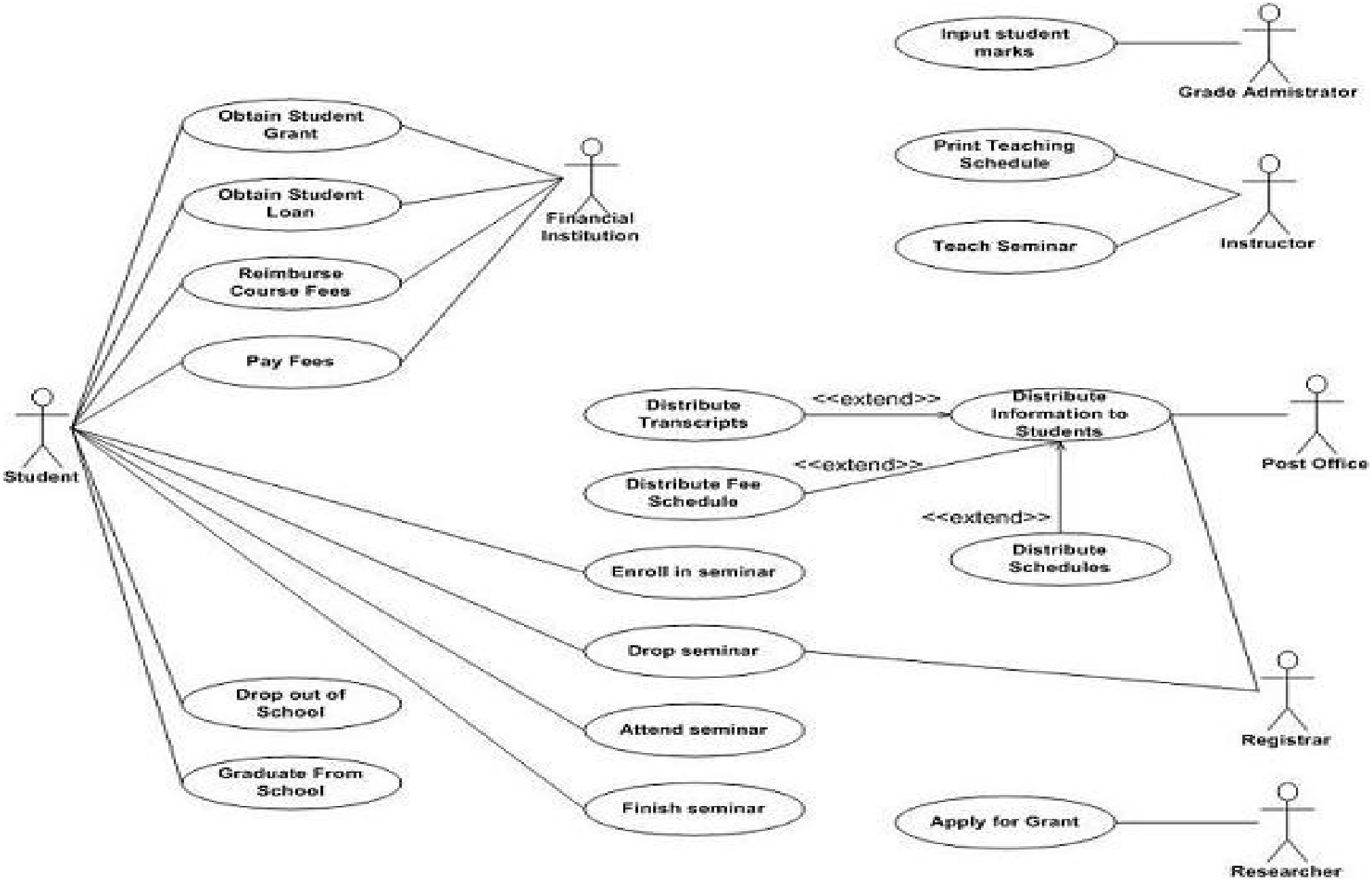
Use cases diagram (cont…)

- ***Actors:*** An actor is a person, organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures.

- ***Associations:*** Associations between actors and use cases are indicated by solid lines. An association exists whenever an actor is involved with an interaction described by a use case.

Use cases diagram

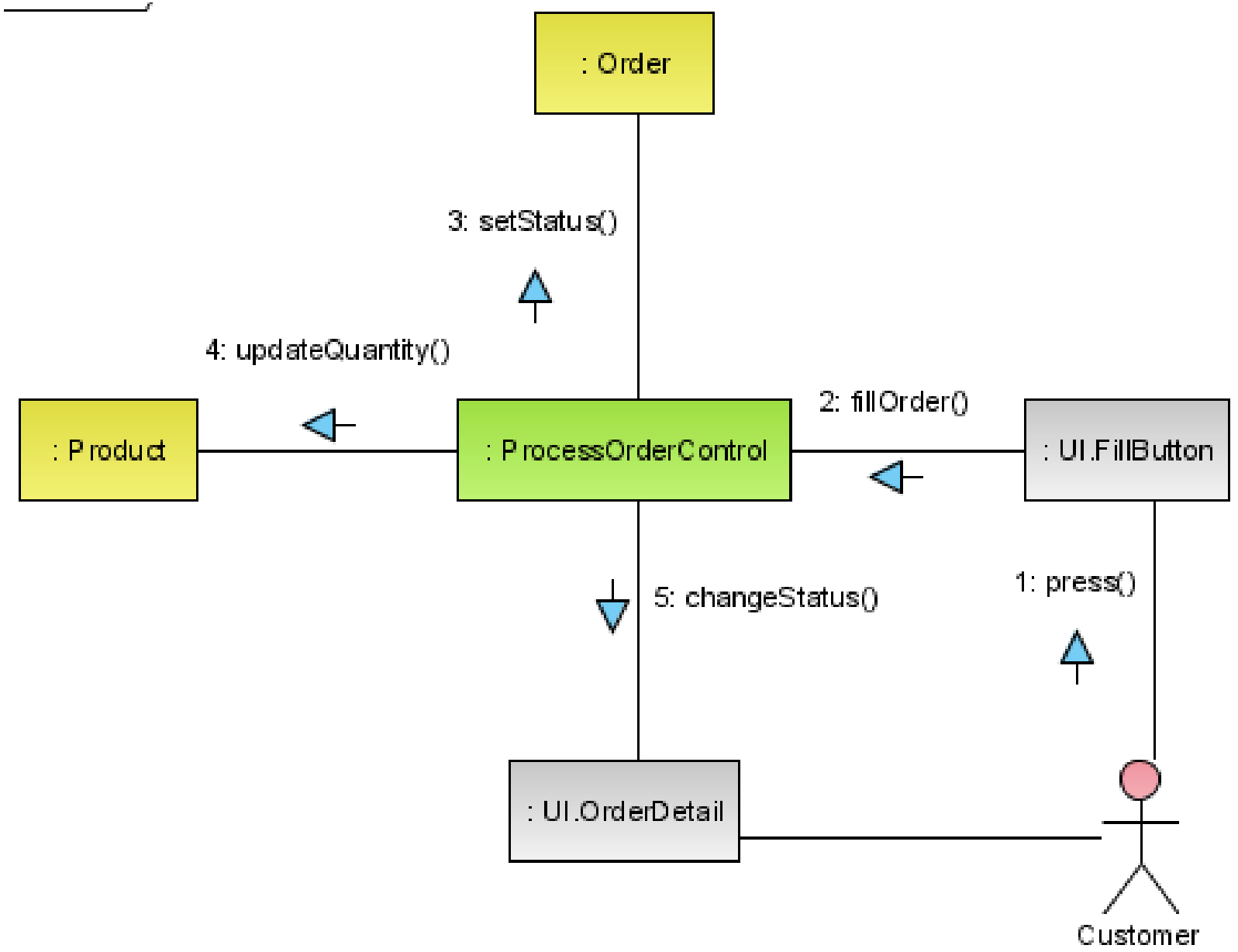
Use cases diagram



Communication diagram

- Communication diagrams used to model the dynamic behavior of the use case.
- When compare to Sequence Diagram, the Communication Diagram is more focused on showing the collaboration of objects rather than the time sequence.

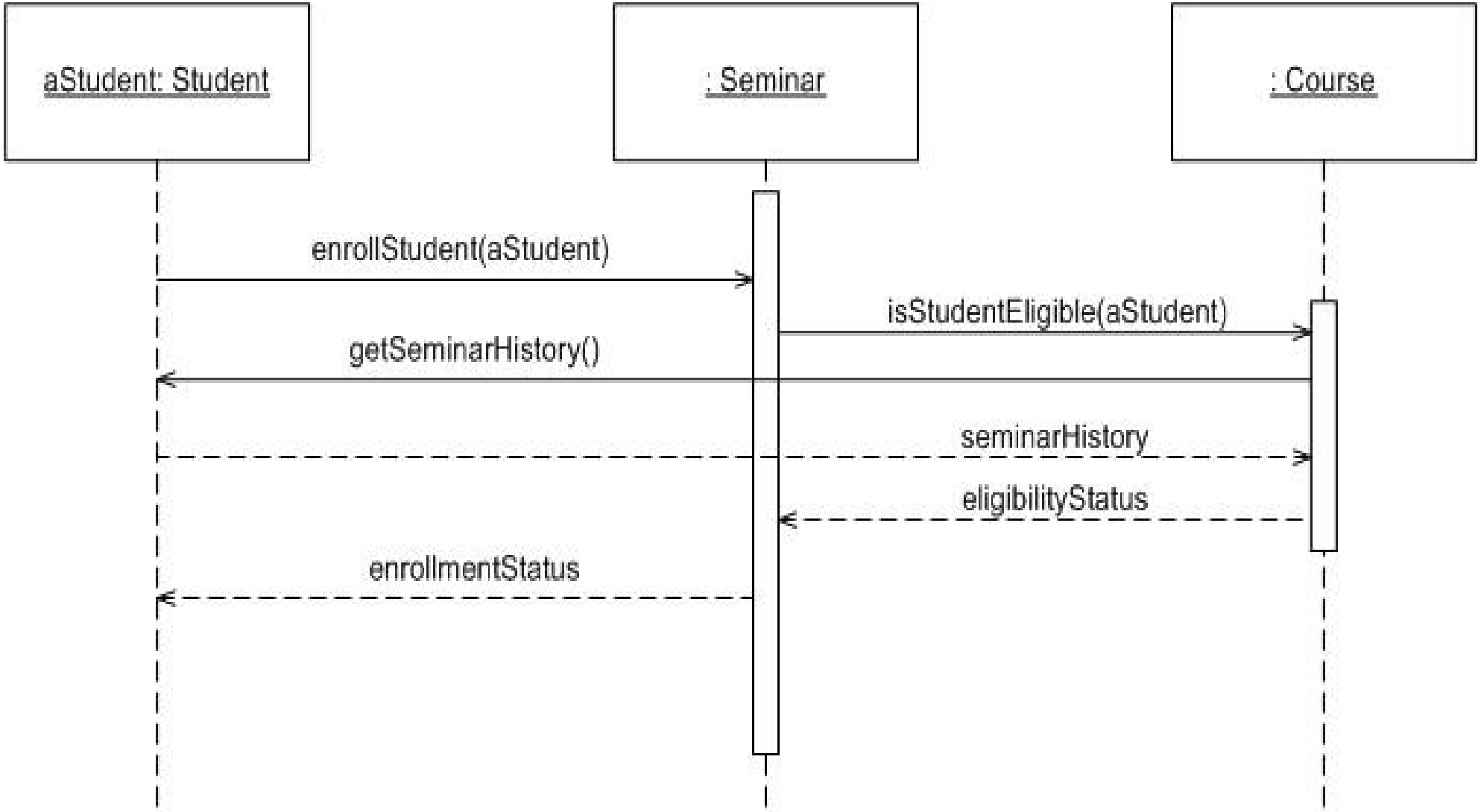
Communication diagram



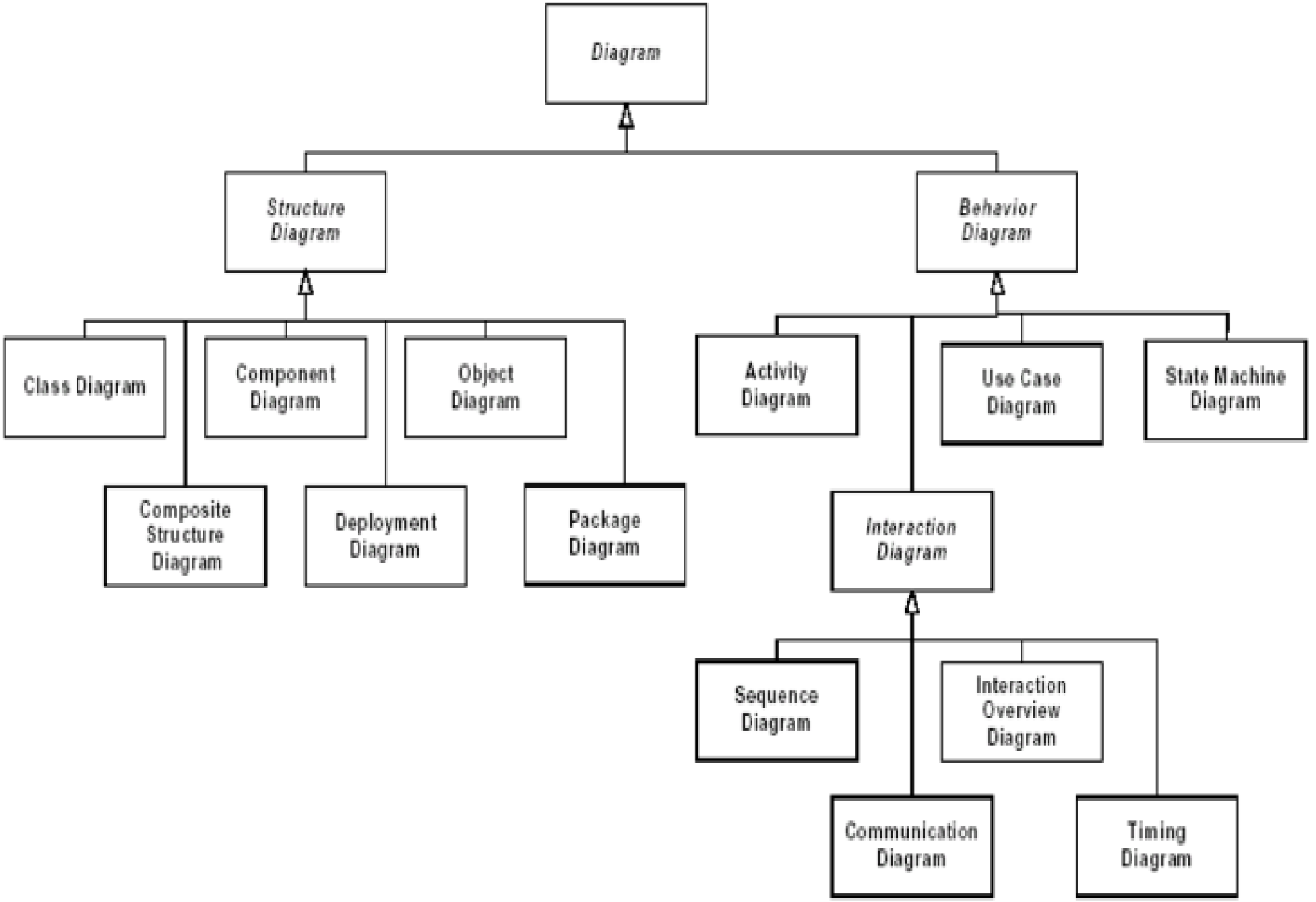
Sequence diagram

- ✓ UML Sequence diagrams models the collaboration of objects based on a time sequence.
- ✓ It shows how the objects interact with others in a particular scenario of a use case.

Sequence diagram



UML Diagram Hierarchy



References

- http://www.agilemodeling.com/
- http://www.visual-paradigm.com/VPGallery/diagrams/index.html
- http://bdn.borland.com/article/0,1410,31863,00.html
- http://en.wikipedia.org/wiki/Unified_Modeling_Language
- http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/index.htm