

OBJECT ORIENTED ANALYSIS AND DESIGN

- Object- Oriented Analysis and design(OOAD) is a popular technical approach for –
 - Analyzing
 - Designing an application , system, or business
 - By applying the object oriented paradigm and
 - Visual modeling throughout the development life cycles for better communication and product quality



Object-oriented

Analysis

and

Design

Do the right thing

Analysis emphasizes an investigation of the problem and requirements, rather than a solution.

Do the thing right

Design emphasizes a conceptual solution that fulfils the requirements, rather than its implementation.

DEFINITION OF OOAD

- ▶ **What is OOAD** :- Object-oriented analysis and design(OOAD) is a software engineering approach that models a system as a group of interacting objects.
- ▶ **OOAD** -> A software development approach that emphasizes a logical solution based on objects
- ▶ **EXAMPLES OF OBJECT ORIENTED SYSTEMS :-**
- ▶ IN OO system , “everything is object”.
- ▶ A spreadsheet cell, bar chart, report,
- ▶ Numbers, arrays , records , fields , files ,forms, an invoice , etc



Difference between Traditional and Object Oriented Approach


TRADITIONAL APPROACH	OBJECT ORIENTED SYSTEM DEVELOPMENT
Collection of procedures(functions)	Combination of data and functionality
Focuses on function and procedures, different styles and methodologies for each step of process	Focuses on object, classes, modules that can be easily replaced, modified and reused.
Moving from one phase to another phase is complex.	Moving from one phase to another phase is easier.
Increases duration of project	decreases duration of project
Increases complexity	Reduces complexity and redundancy

OBJECT-ORIENTED ANALYSIS

- ❖ Object-Oriented Analysis (OOA) is the procedure of identifying software engineering requirements and developing software specifications in terms of a software system's object model, which comprises of interacting objects.
- ❖ The primary tasks in object-oriented analysis (OOA) are –
 1. Identifying objects
 2. Organizing the objects by creating object model diagram.
 3. Defining the internals of the objects, or object attributes.
 4. Defining the behavior of the objects, i.e., object actions
 5. Describing how the objects interact




OBJECT-ORIENTED DESIGN

- ▶ Object-Oriented Design (OOD) involves implementation of the conceptual model produced during object-oriented analysis.
 - ▶ In OOD, concepts in the analysis model, which are technology-independent, are mapped onto implementing classes, constraints are identified and interfaces are designed, resulting in a model for the solution domain, i.e., a detailed description of how the system is to be built on concrete technologies.
 - ▶ OOA focuses on What the system does , OOD on how the system does it.
- 

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.
- 

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

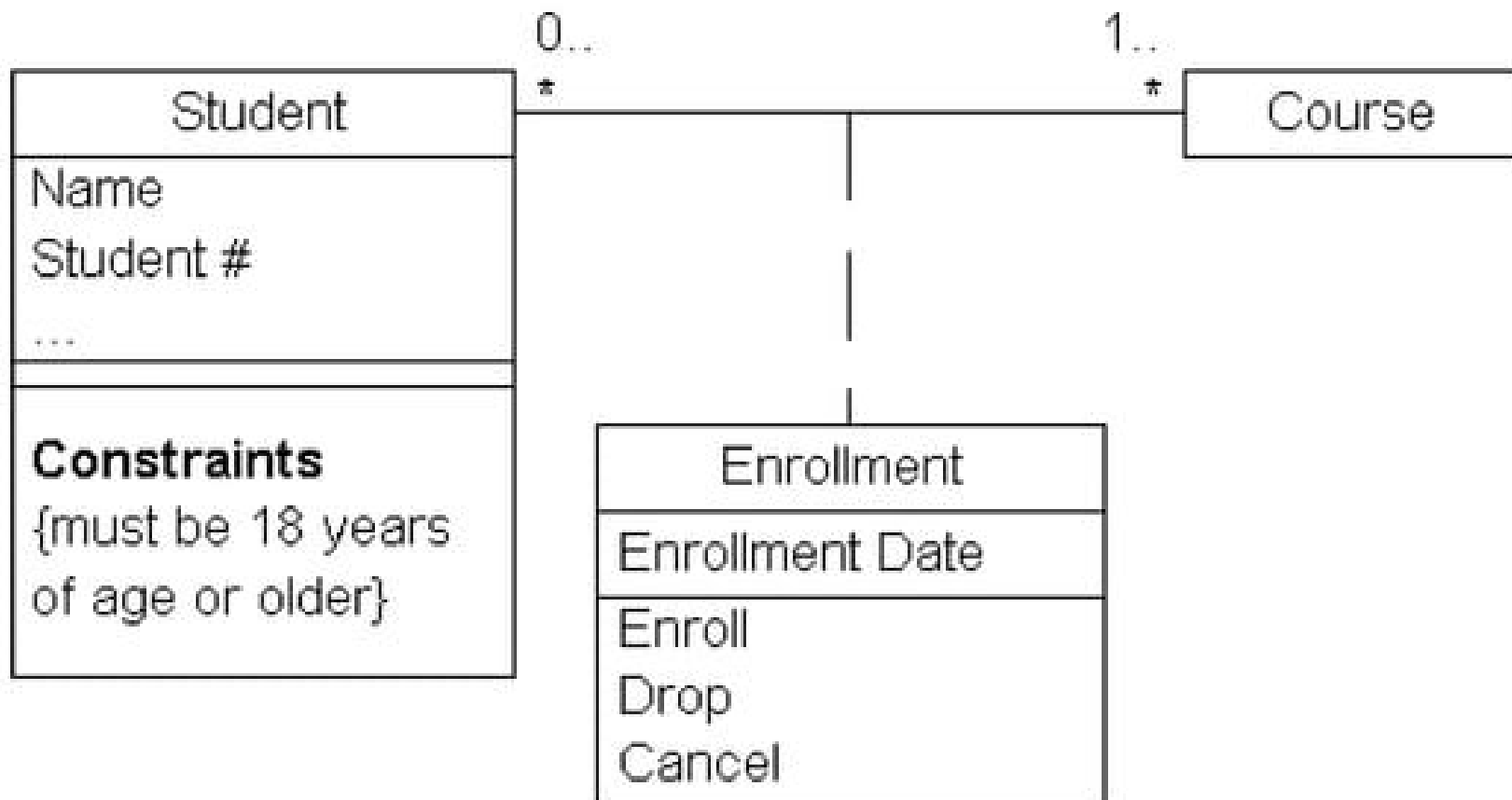
- Communication(collaboration)
- Sequence
- *Interaction overview*
- *Timing*

Class diagram

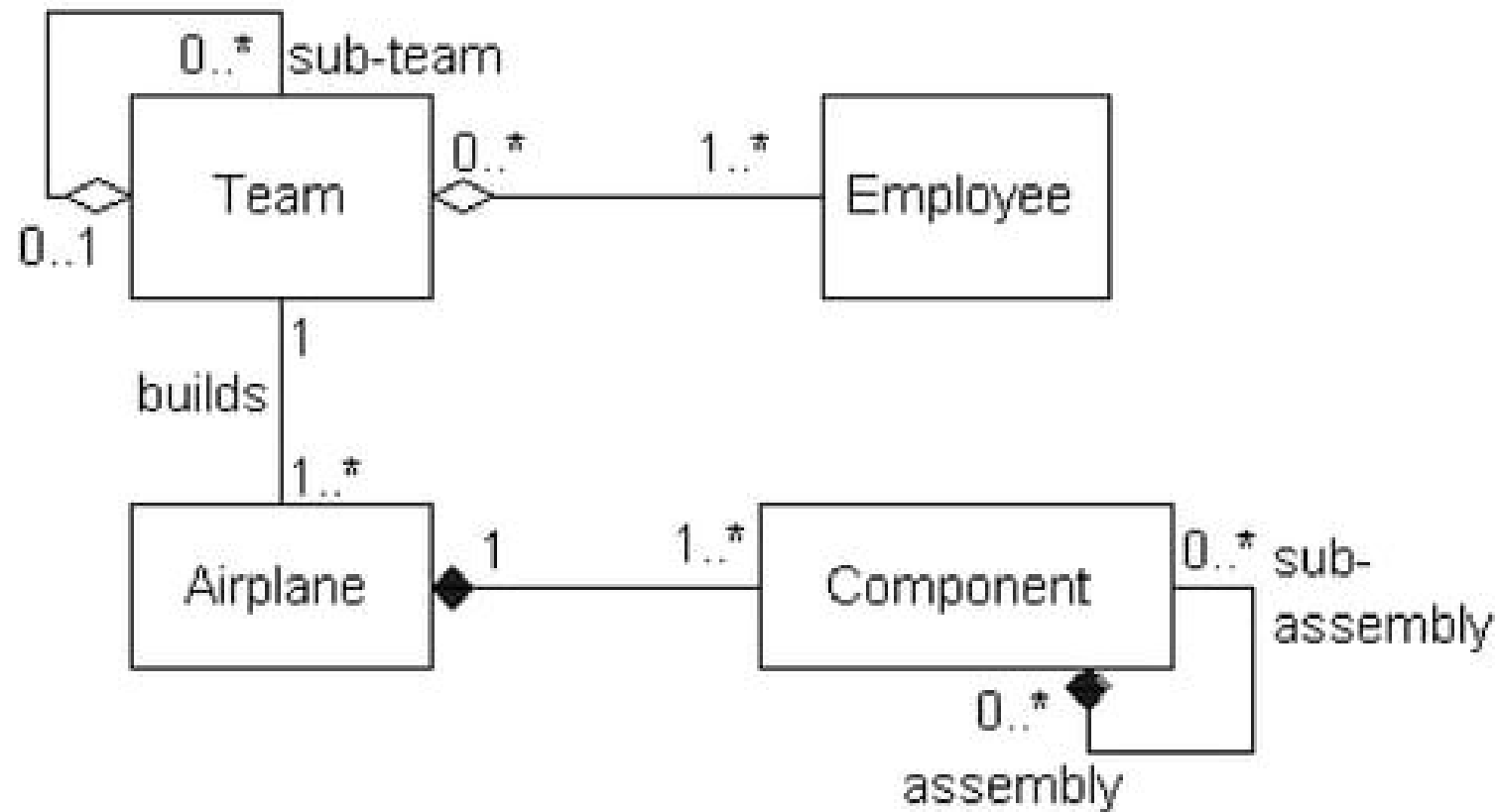
UML class diagrams show the classes of the system, their inter-relationships, and the operations and attributes of the classes

- 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 diagram



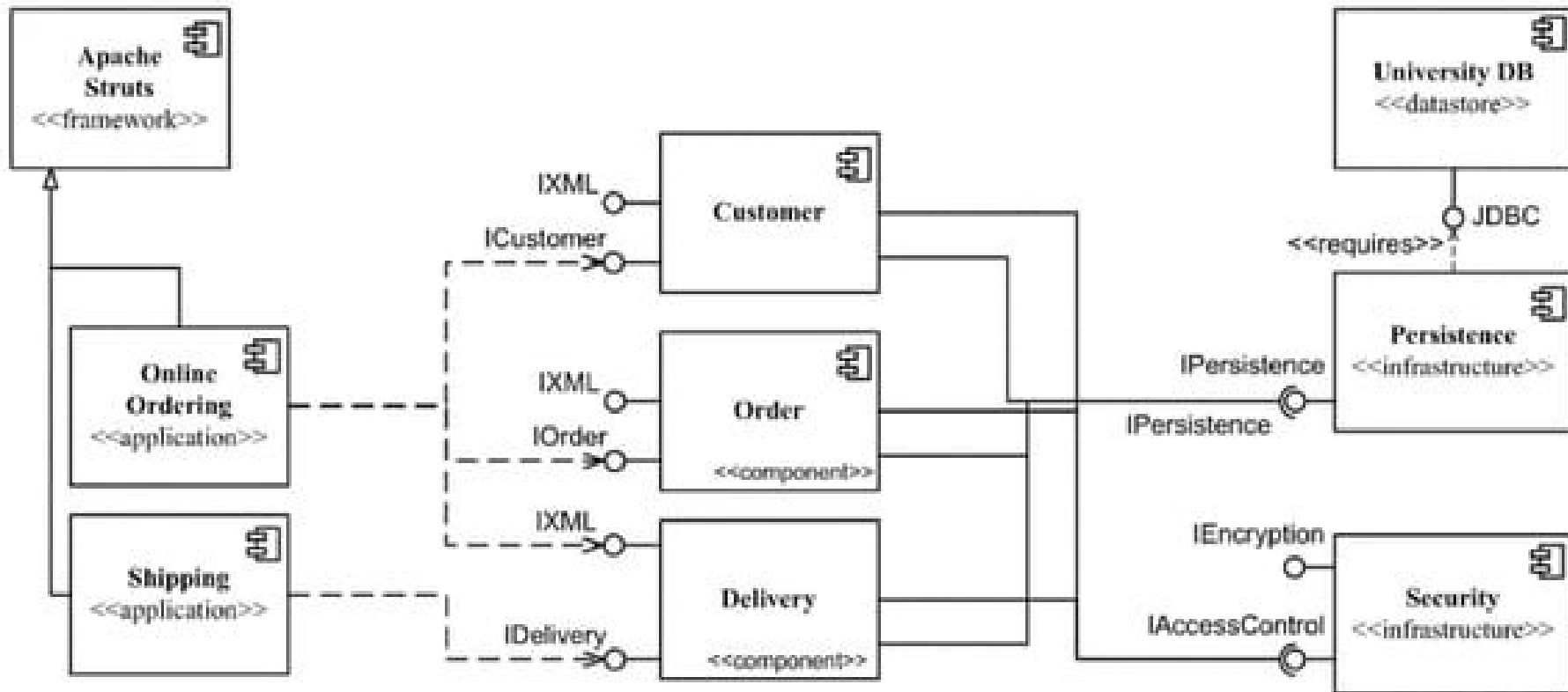
Class diagram



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

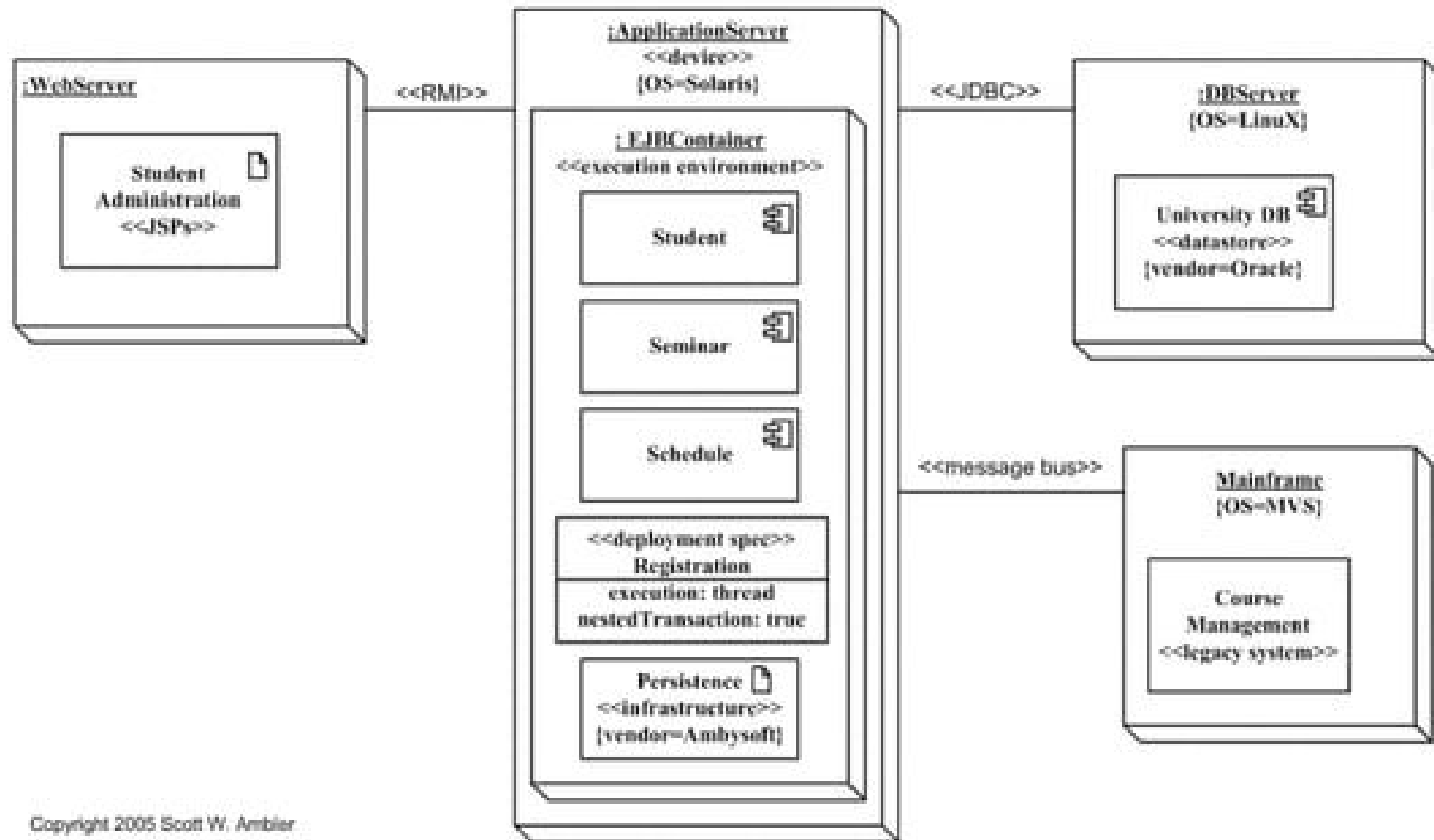


Copyright 2005 Scott W. Ambler

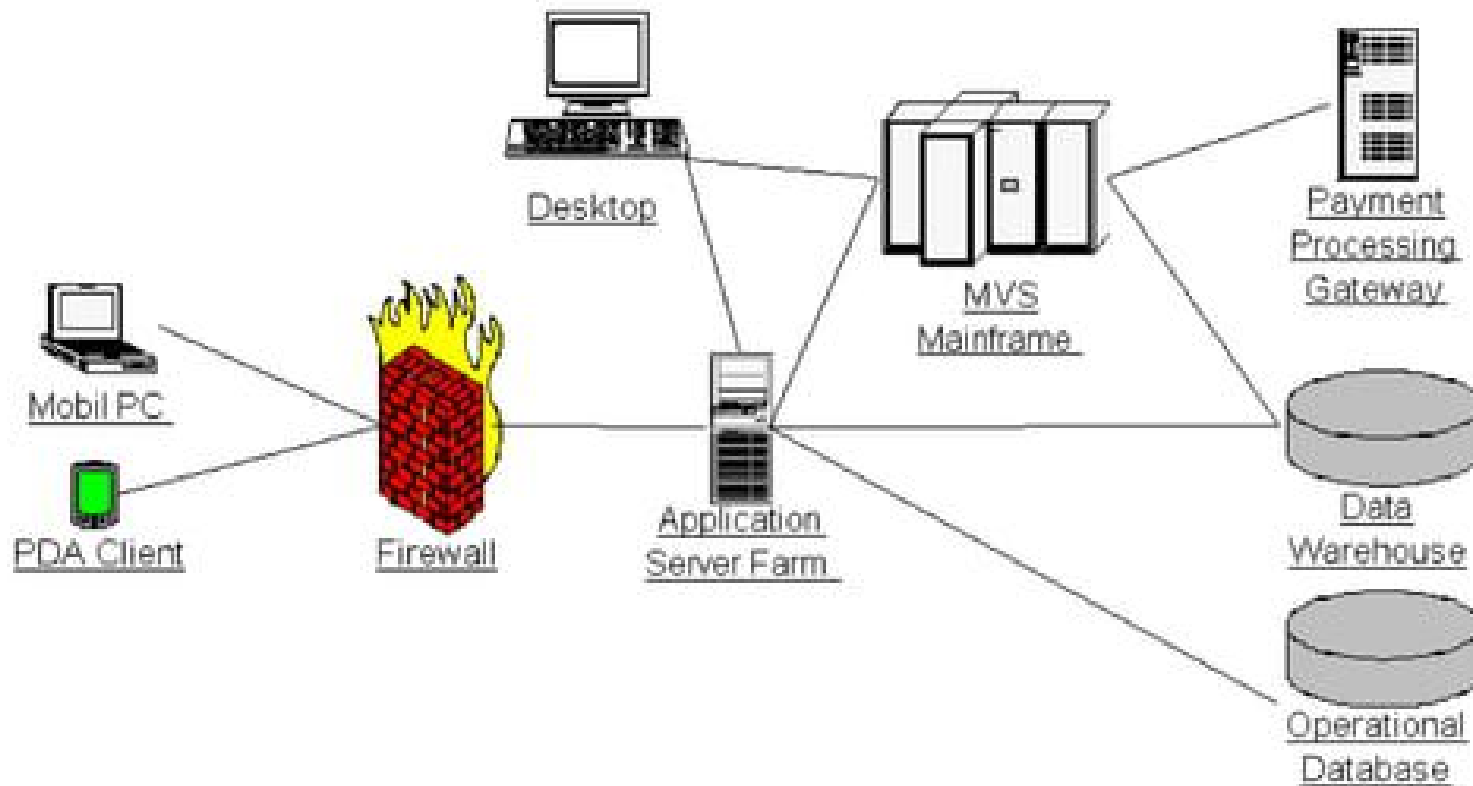
Deployment diagram

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



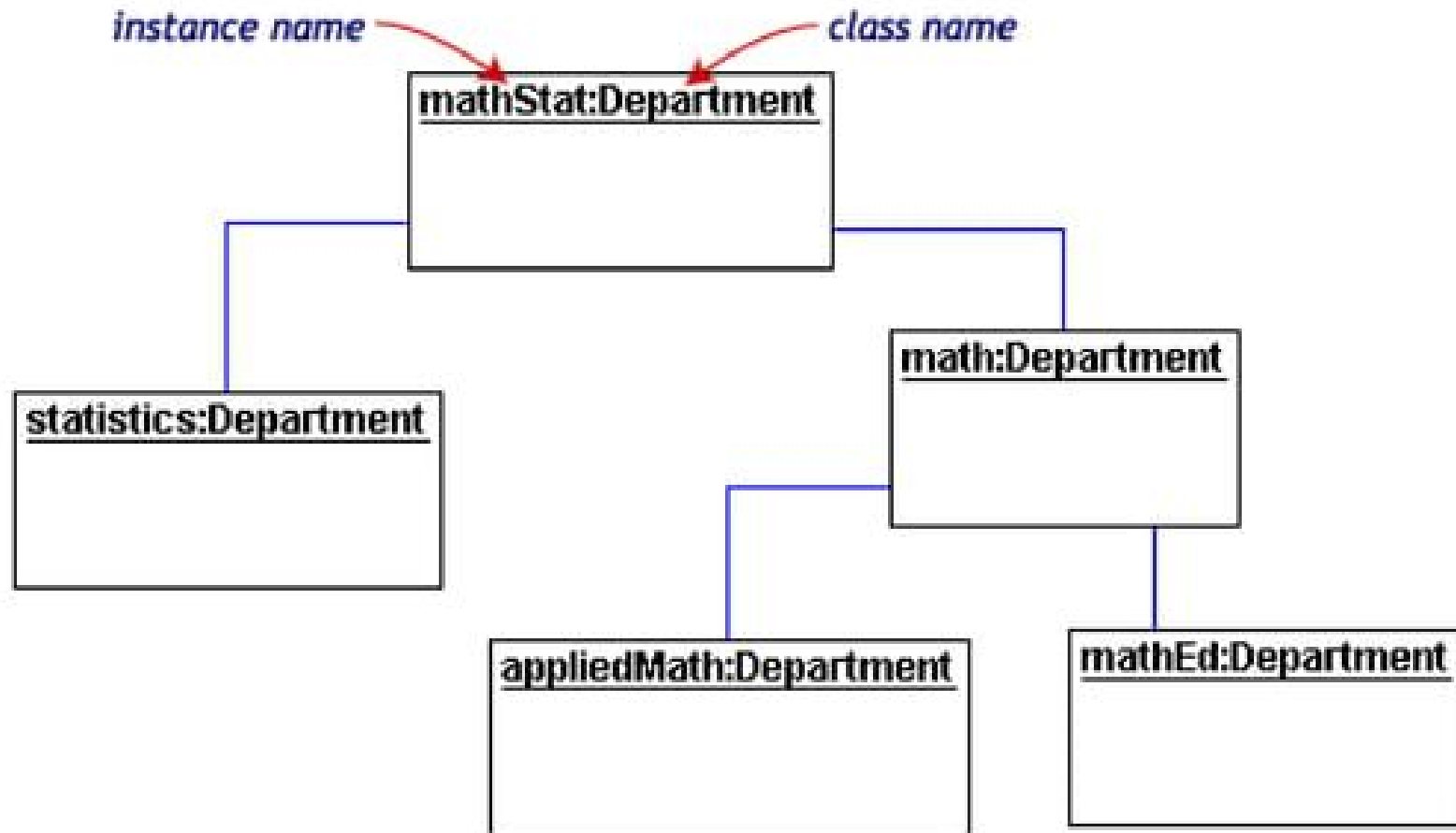
Deployment diagram



Object diagram

- **UML 2 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.

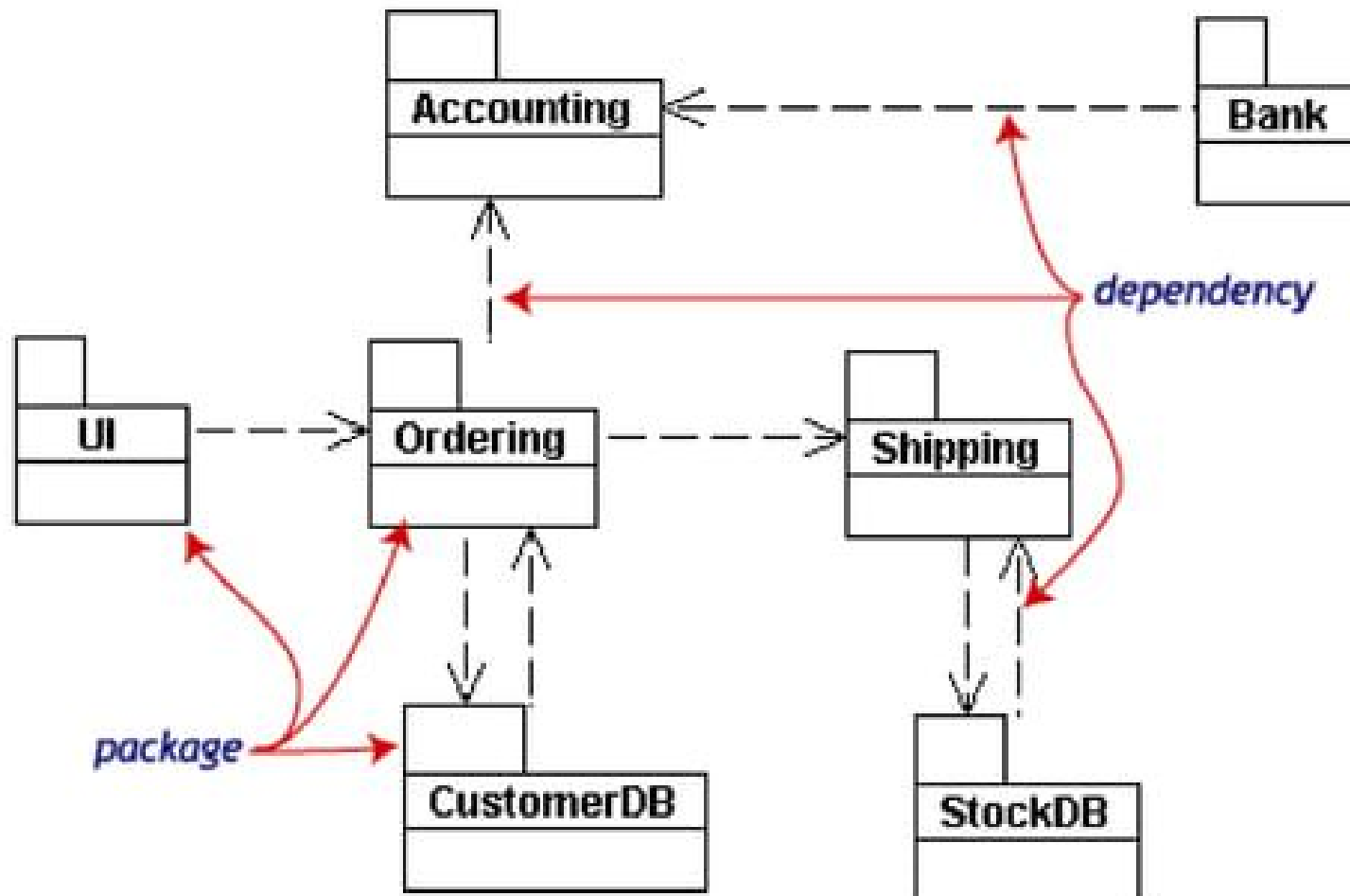
Object diagram



Package diagram

- **UML 2 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.

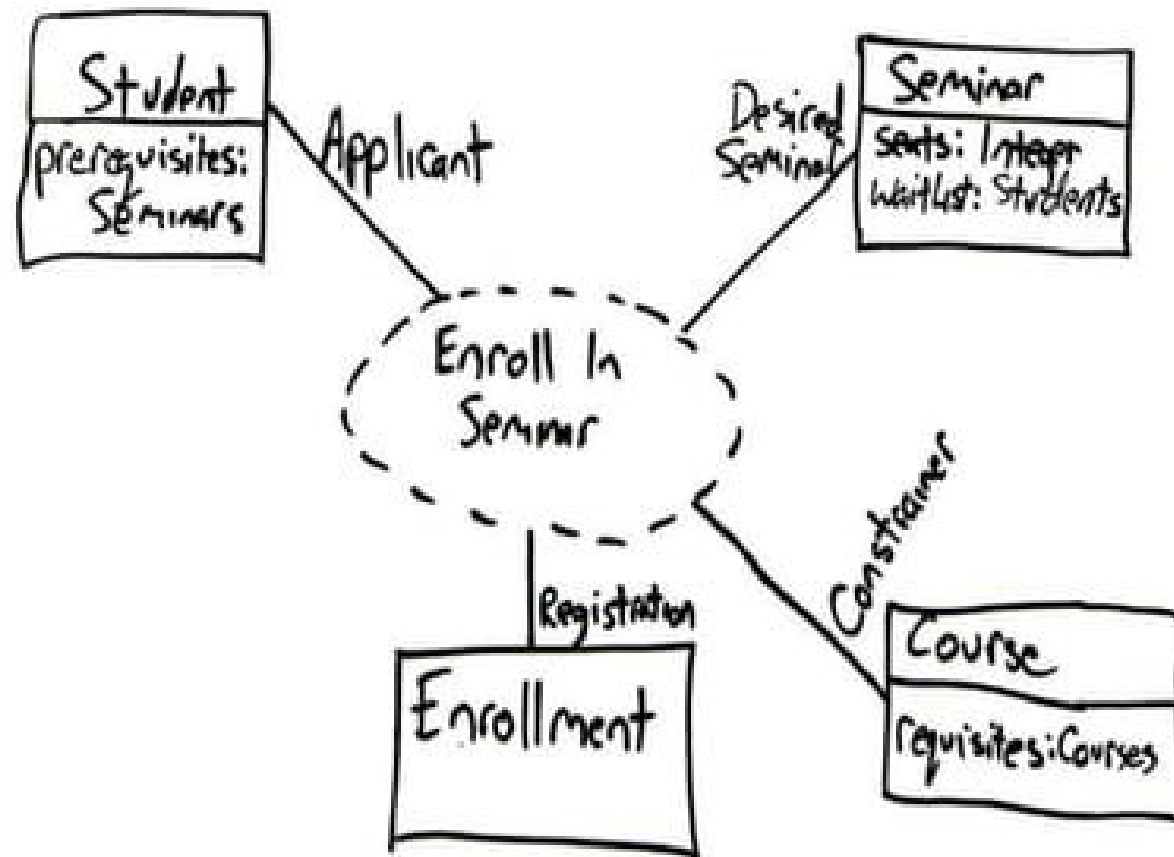
Package diagram



Composite structure diagram

- UML 2 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.

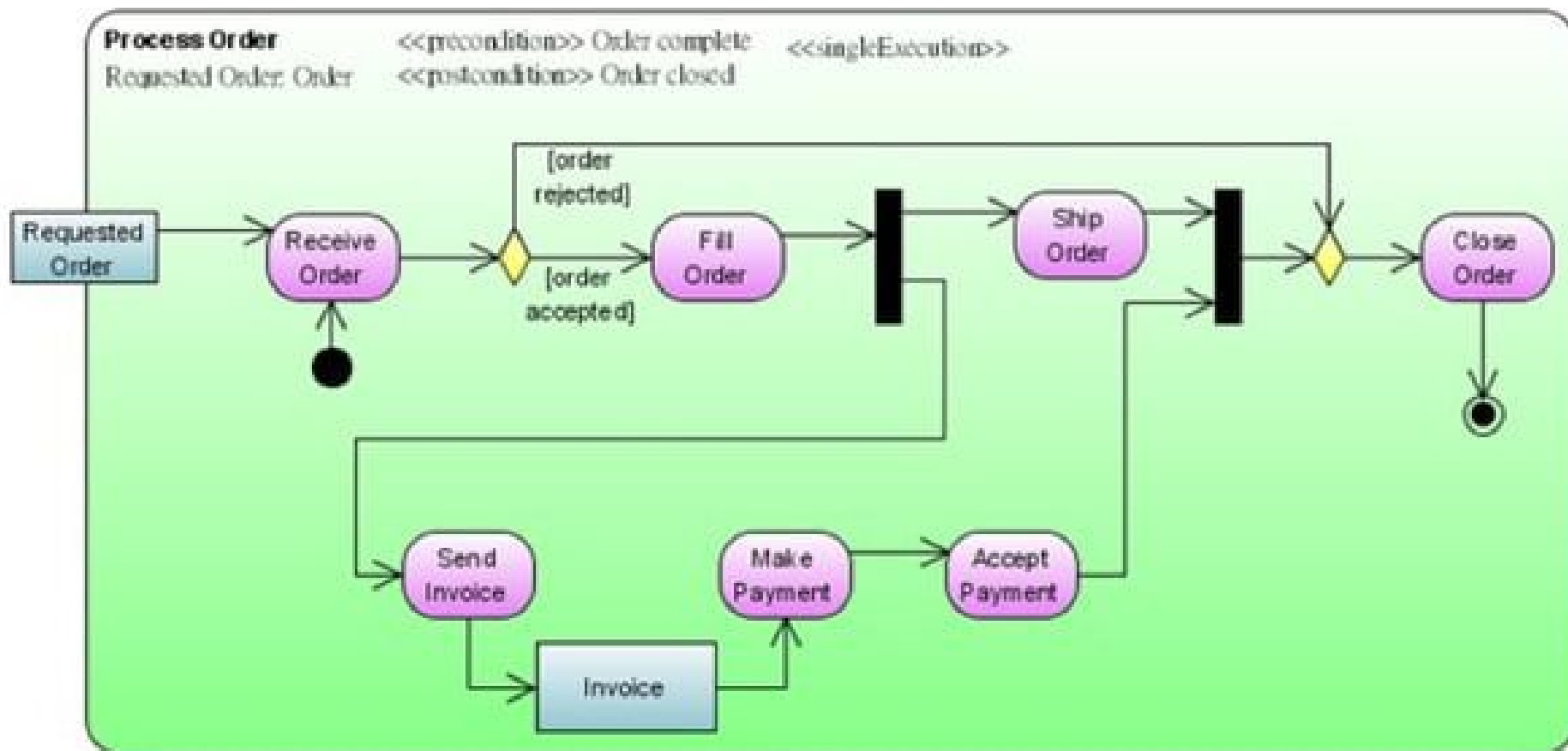
Composite structure diagram



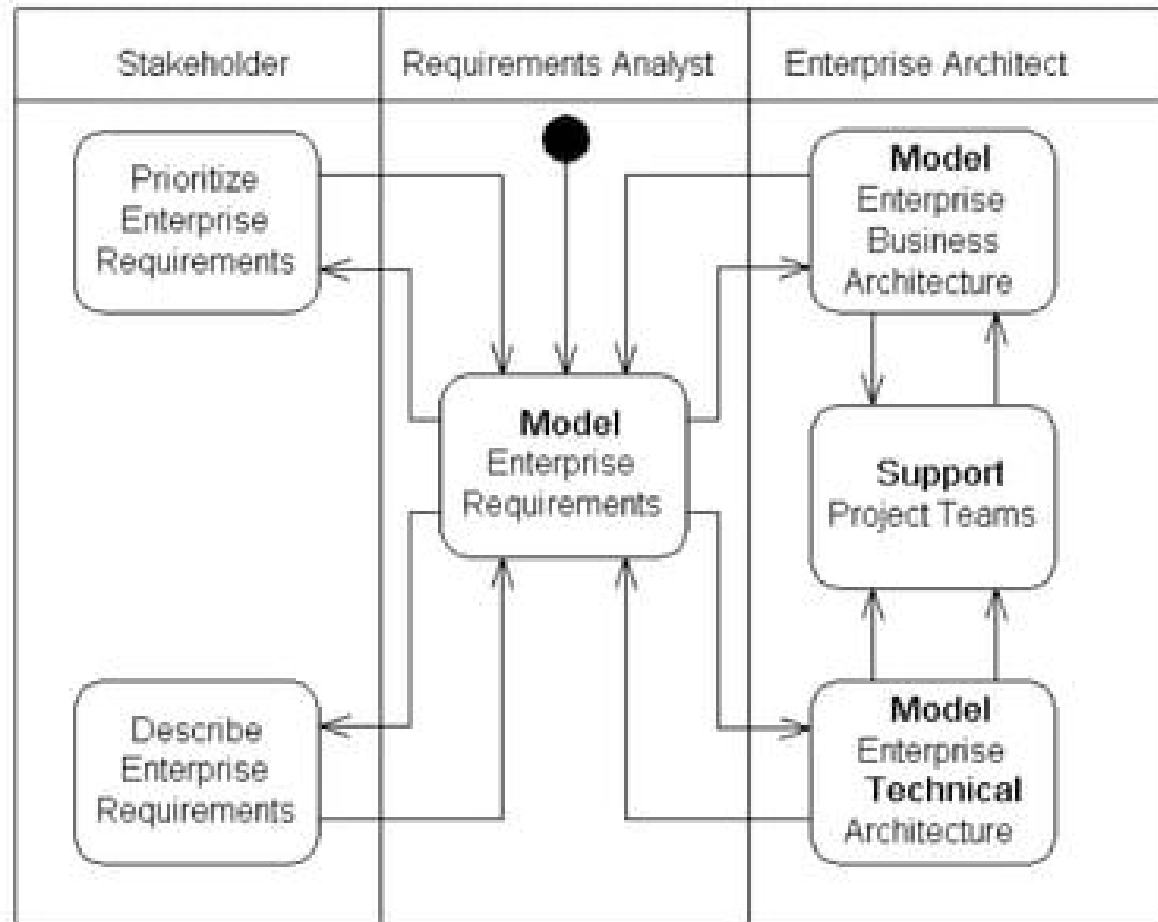
Activity diagram

- UML 2 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



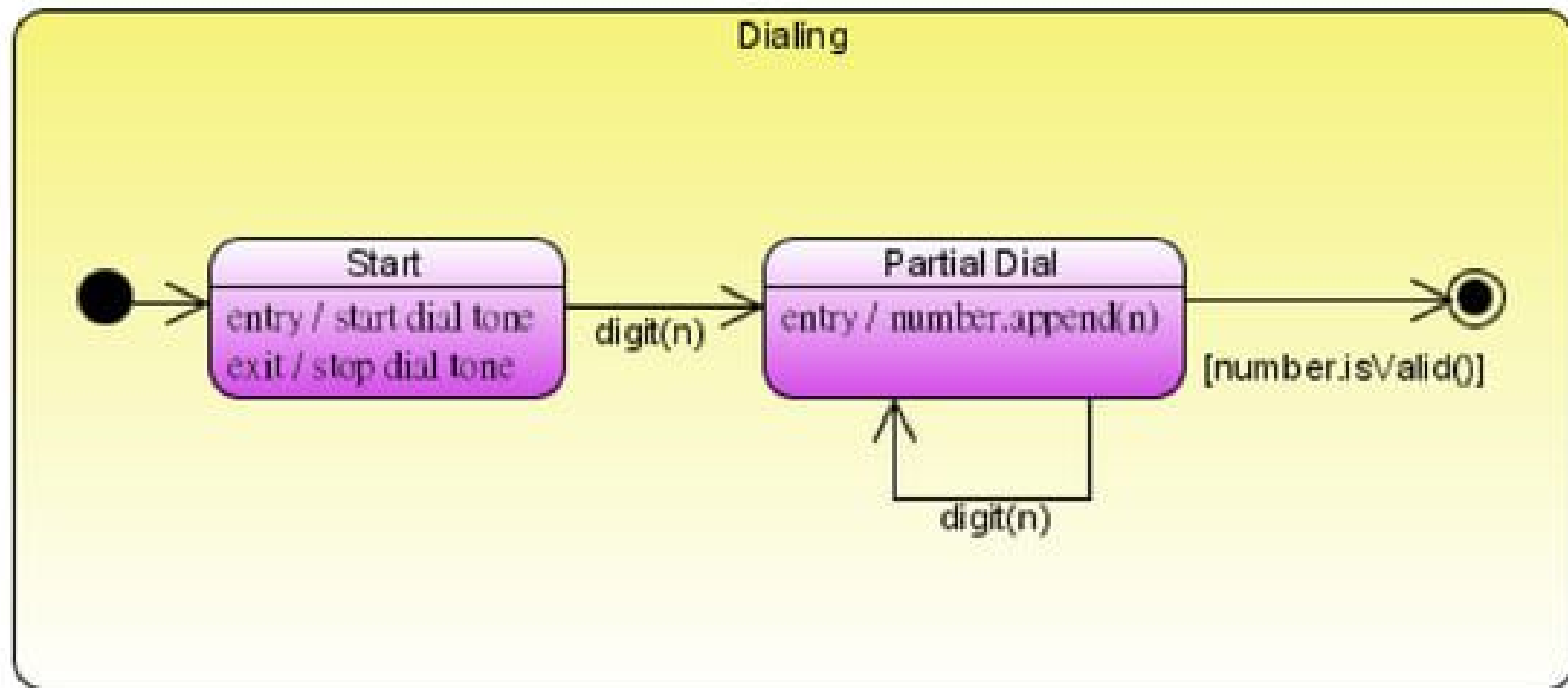
Activity diagram



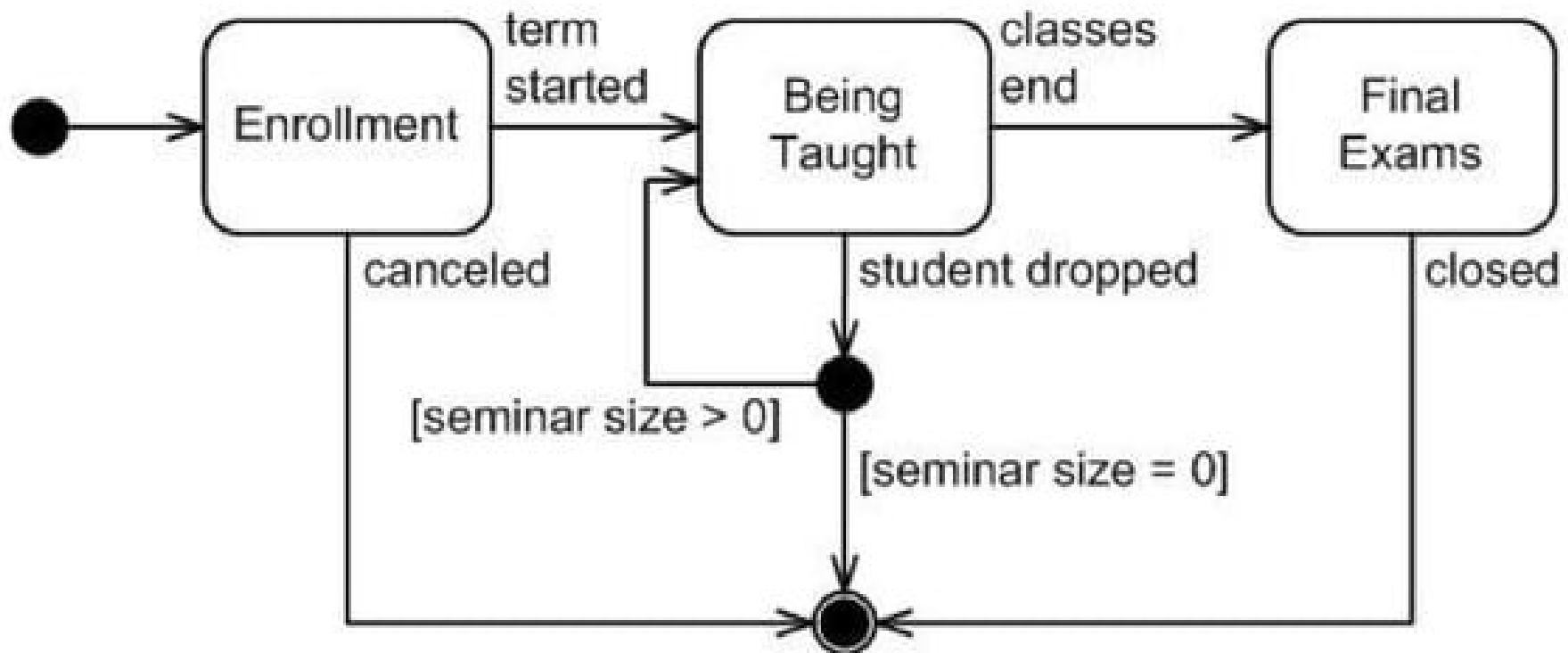
State machine diagram

- UML 2 State machine 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 machine diagram



State machine diagram

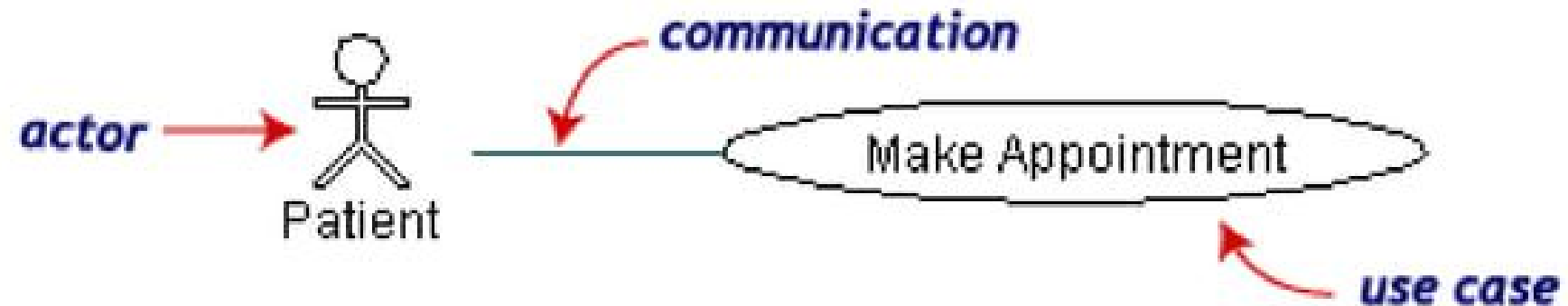


Use cases diagram

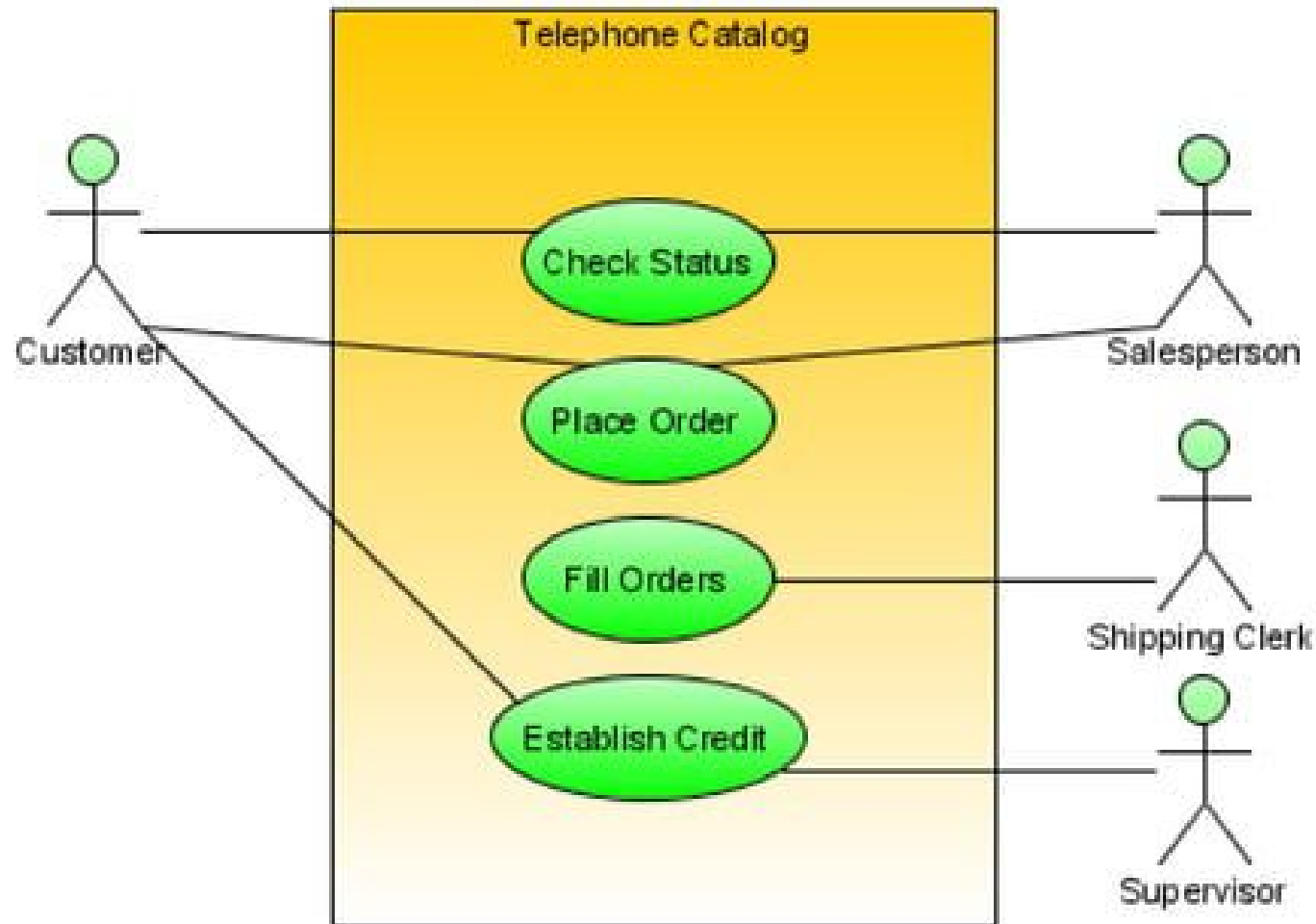
UML 2 Use cases diagrams describes the behavior of the target system from an external point of view. Use cases describe "the meat" of the actual requirements.

- **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.
- **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.

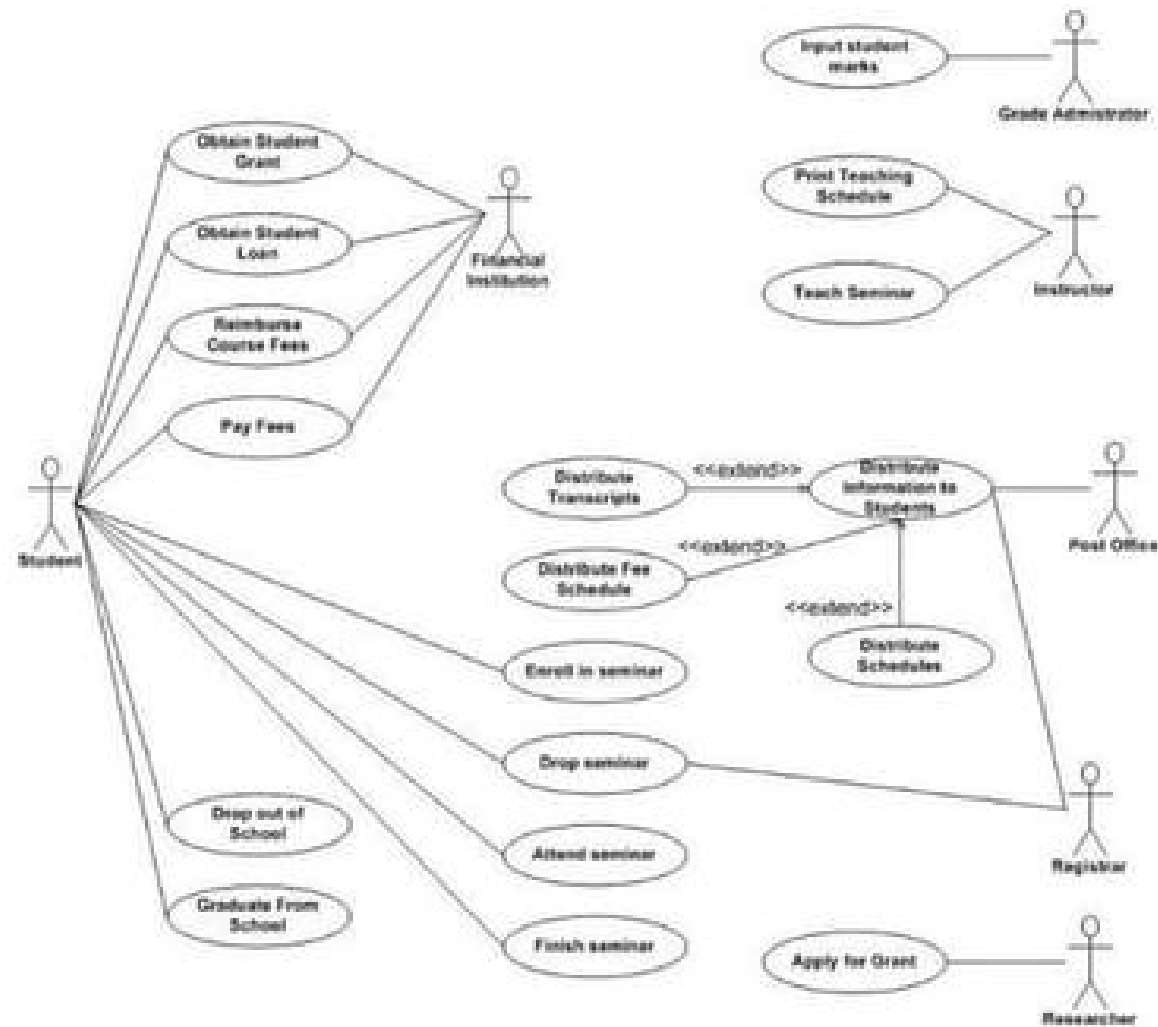
Use cases diagram



Use cases diagram



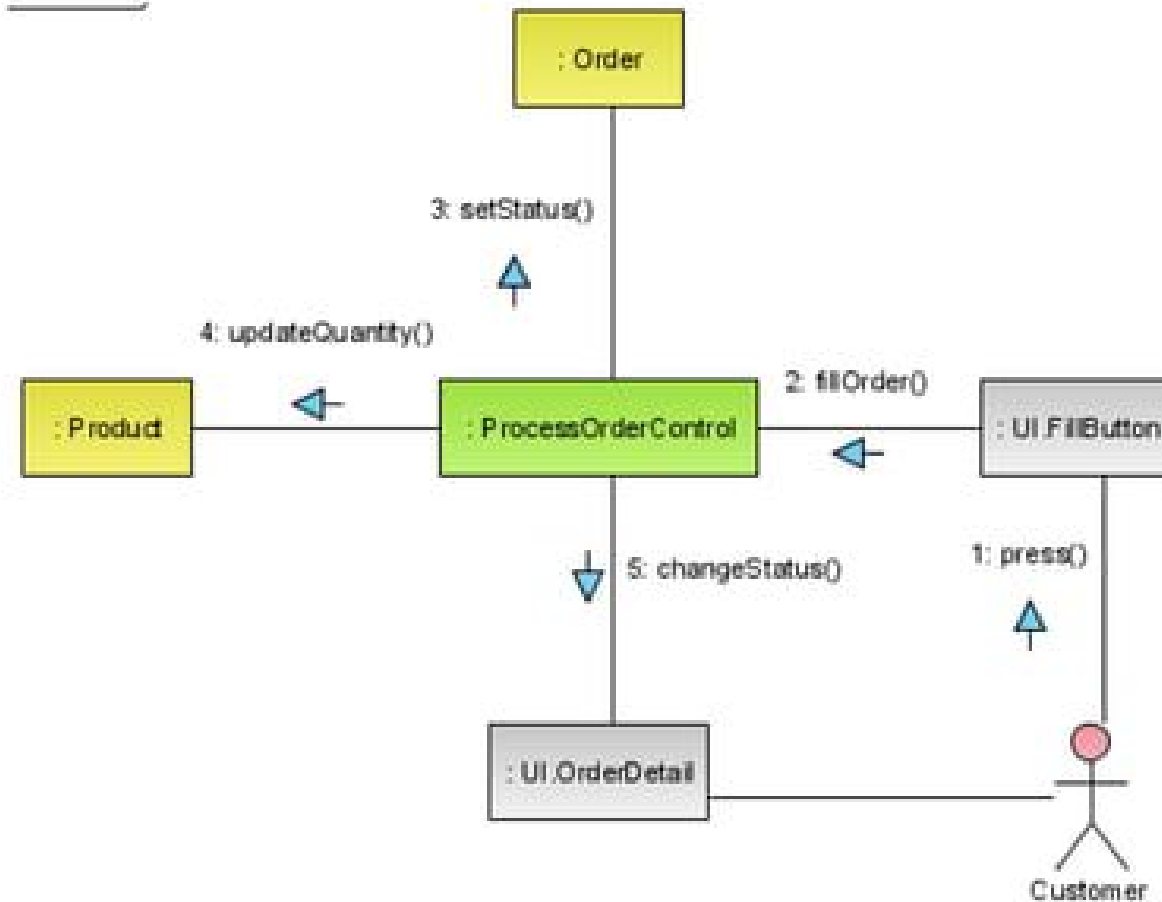
Use cases diagram



Communication diagram

- UML 2 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.

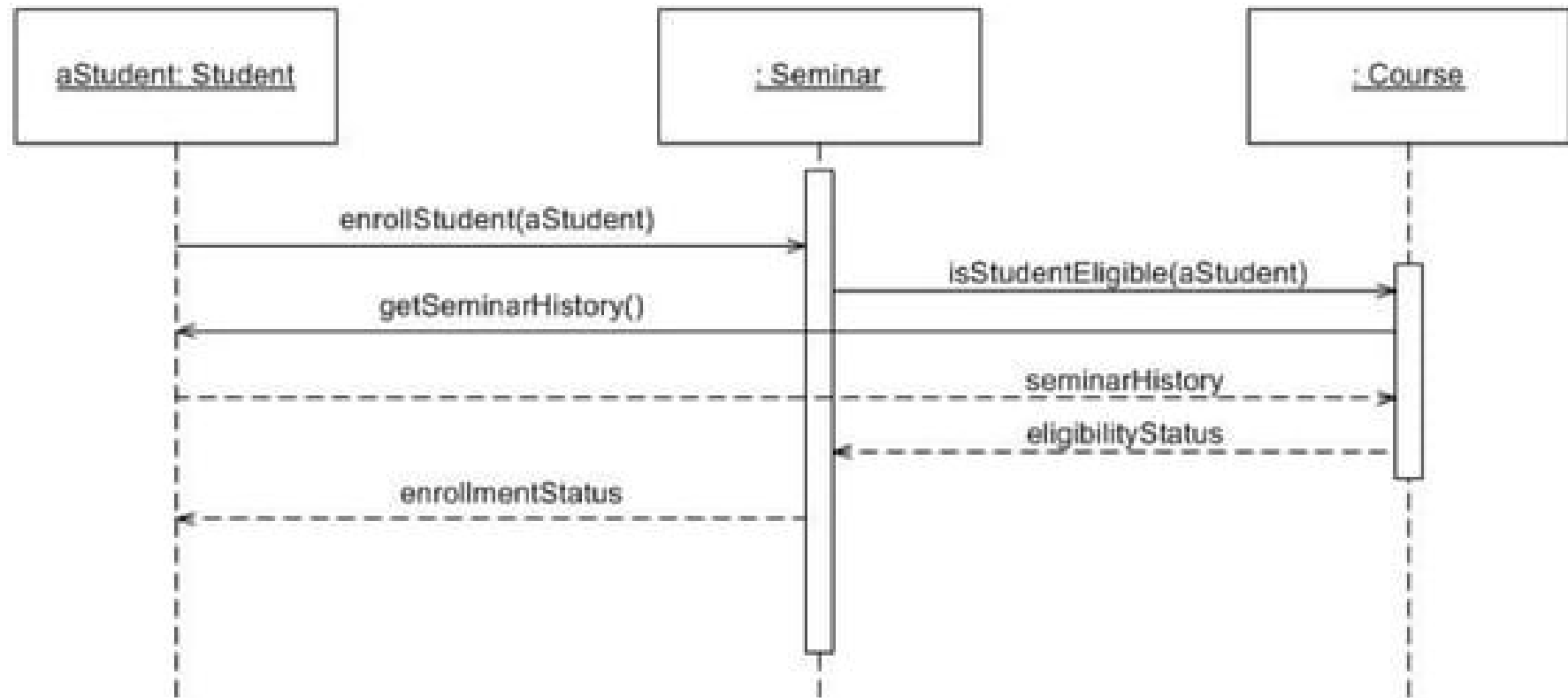
Communication diagram



Sequence diagram

- UML 2 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.

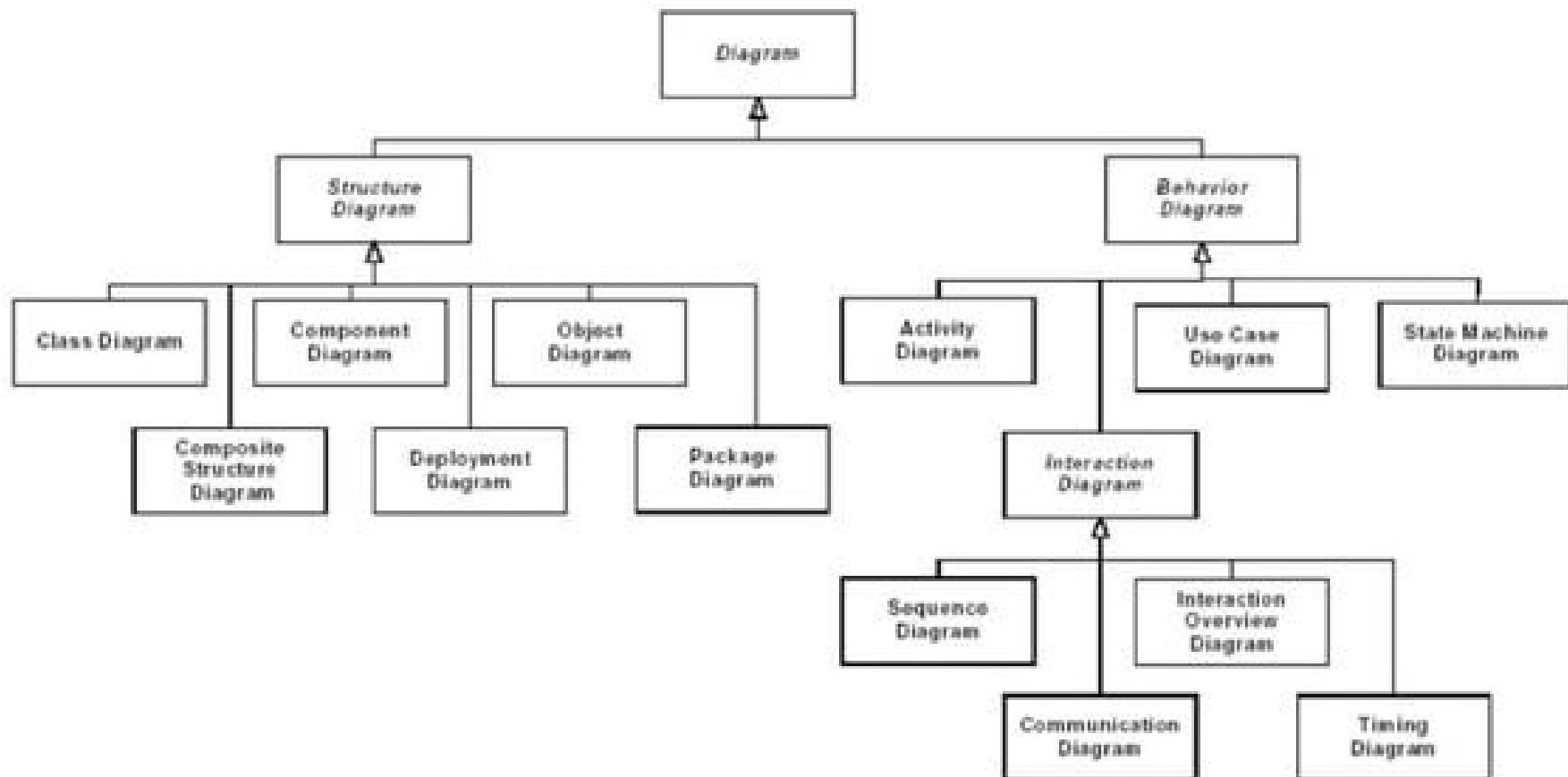
Sequence diagram



Timing diagram

- UML 2 Timing diagrams shows the behavior of the objects in a given period of time. Timing diagram is a special form of a sequence diagram. The differences between timing diagram and sequence diagram are the axes are reversed so that the time are increase from left to right and the lifelines are shown in separate compartments arranged vertically.

UML diagram hierarchy



Interaction overview diagram

- UML 2 Interaction overview diagrams focuses on the overview of the flow of control of the interactions. It is a variant of the Activity Diagram where the nodes are the interactions or interaction occurrences. It describes the interactions where messages and lifelines are hidden.