Requirement Analysis & Specification

- Object-Oriented Analysis -

Fall, 2024

jehong@chungbuk.ac.kr

Topics Covered

Requirement Analysis Methods

Object-Oriented Analysis

- Overview
- Functional Modeling
- Structural Modeling
- Behavioral Modeling

Balancing the Models

Summary & Discussion

Methods for Requirements Analysis & Spec. (1)

Structured Analysis

- Popular in the late 1970s and 1980s
- Focused on the data and the functional processes
- Representative methodologies
  - SA/SD : Structured Analysis and Structured Design
  - CODART: Concurrent Design Approach for Real-Time system

Object-Oriented Analysis

- Popular in the early 1990s
- Focused on the classes which contain the attributes and operation
- Representative methodologies
  - OMT : Object Modeling Method by J. Rumbaugh
  - COMET: Concurrent Object Modeling mETod by H. Gomma
  - OCTOPUS : by M. Awad
  - ESUML: Embedded Software modeling with UML 2.0

Methods for Requirements Analysis & Spec. (2)

Information Engineering

- Popular in the late 1980s
- Focused on the data and the transactional process
- Representative methodologies
  - Information Engineering by J. Martine

Formal Methods

- Focused on the events, the system states and state transitions
- Representative techniques
  - Petri Net modeling approach by A. Petri
  - Z notation by PRG, Oxford University
  - SDL (Specification and Description Language) by CCITT
  - Statechart by D. Harel

Object-Oriented Analysis

Object-Oriented Analysis : Overview

Methodology

- Formalized techniques to implementing the SDLC/Process models
- Describes "How to develop"

Focus on capturing the structure and behavior of software system in little modules that encompass both data and process

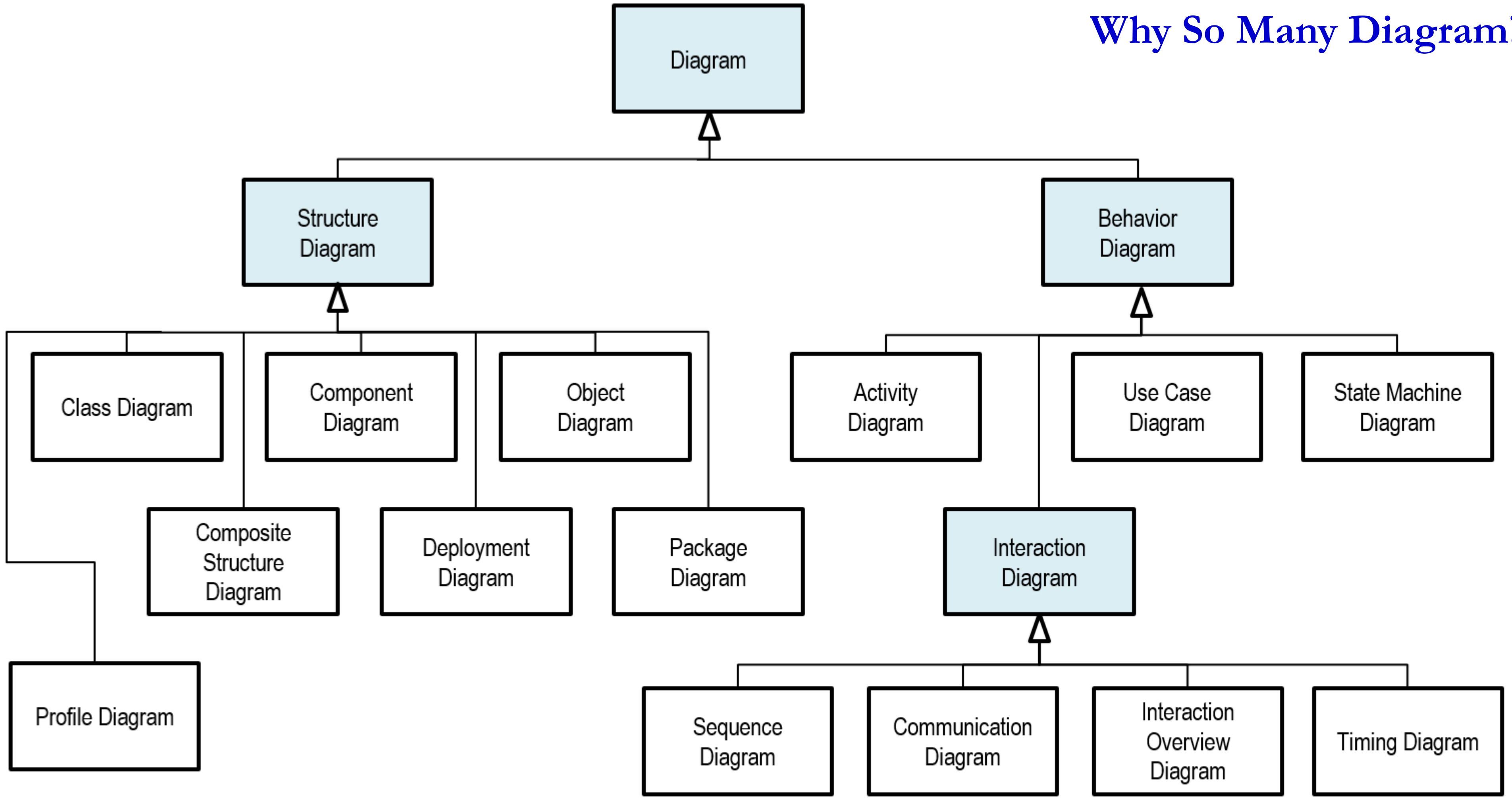
Basic characteristics

- Classes and Objects
- Methods and Message
- Encapsulation and Information hiding
- Inheritance
- Polymorphism and Dynamic Binding

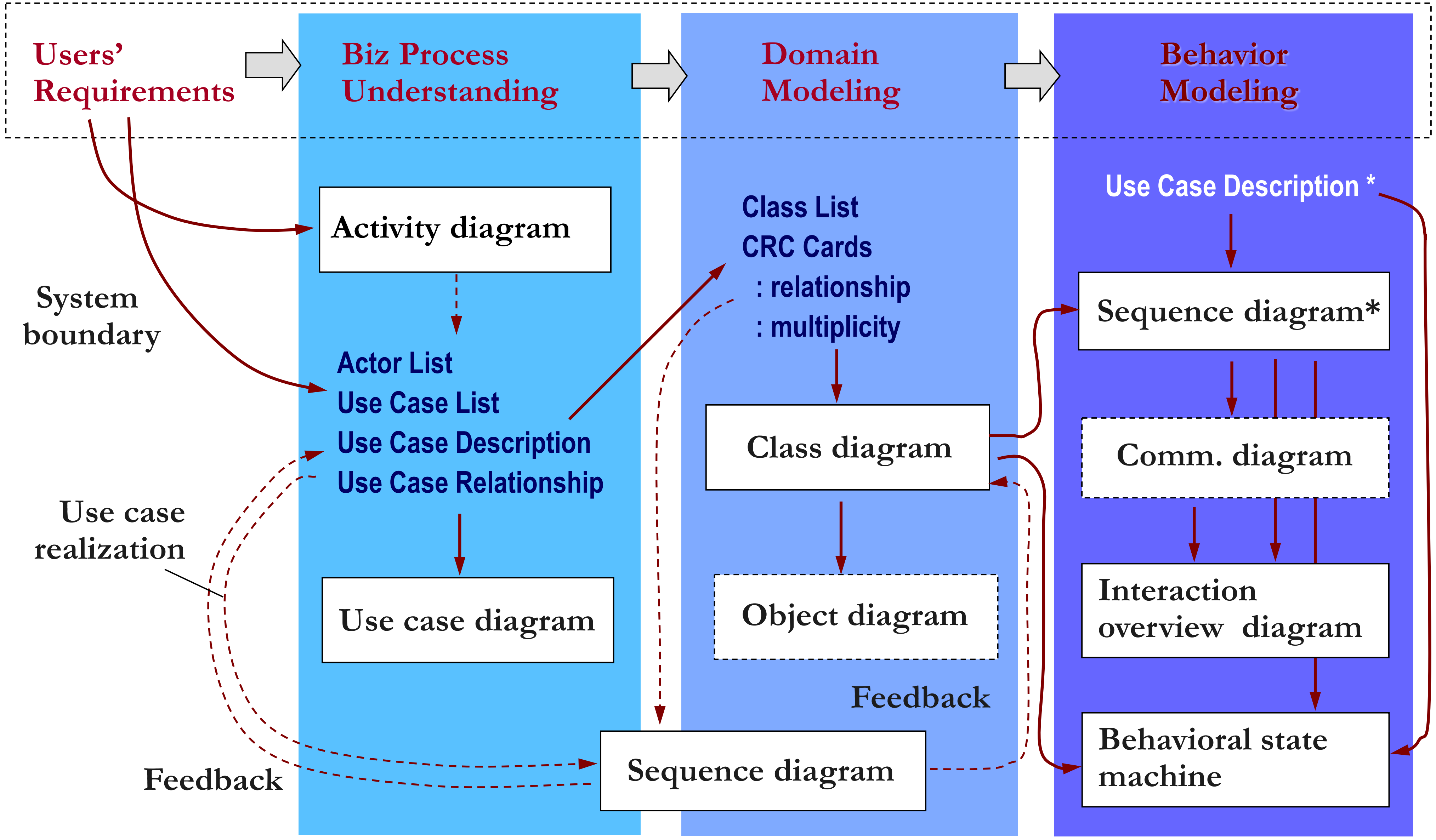
Object-Oriented Analysis : UML Overview

Unified Modeling Language

Why So Many Diagram?



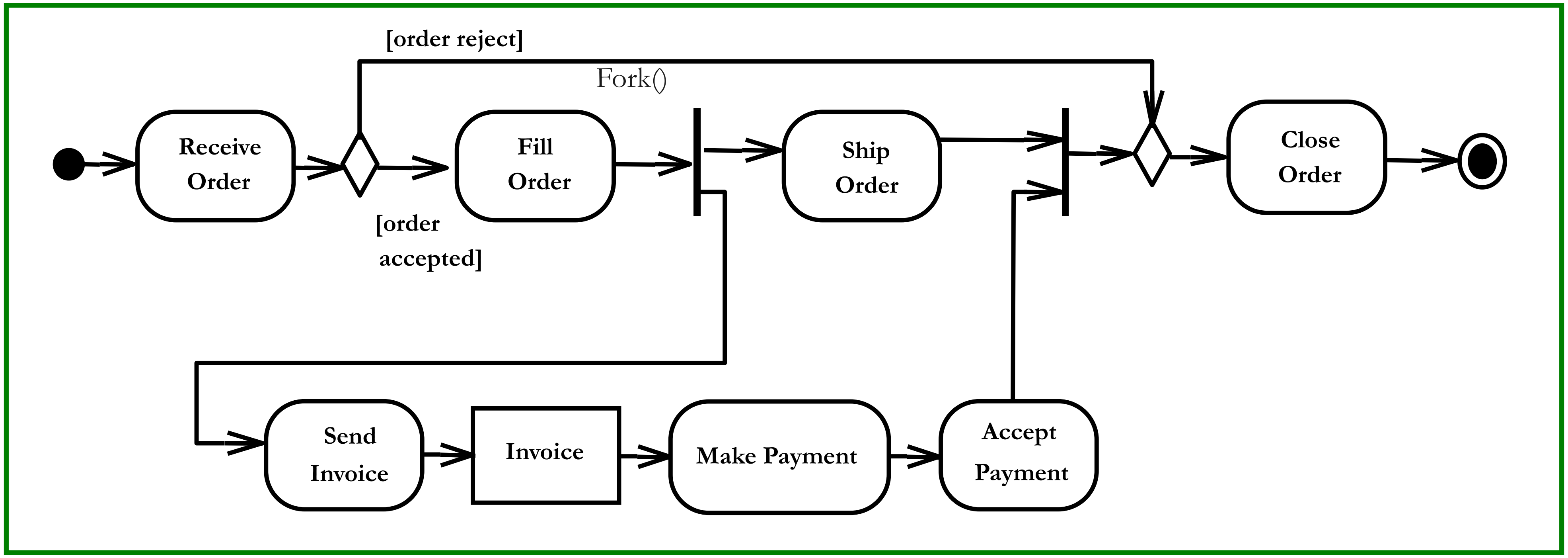
Object-Oriented Analysis : Overall Process



Understanding Biz Process : Activity Diagram

Useful to specify software or hardware system behaviour

Based on data flow models – a graphical representation (with a Directed Graph) of how data move around an information system



Source: Debenedetti Emanuele, Activity diagrams in UML 2.0

Functional Modeling : Use Case Diagram

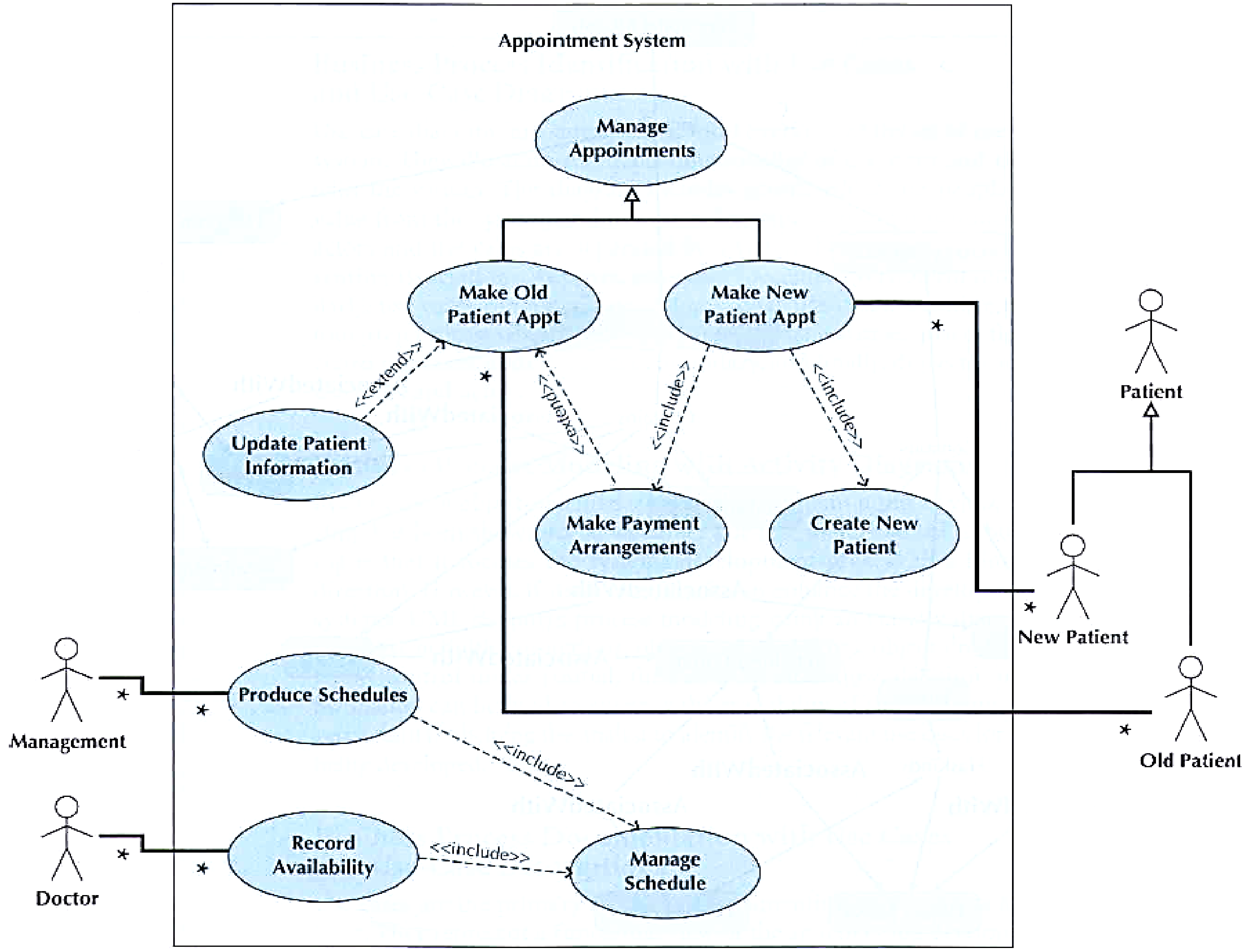
Actor List

Actor Name	Description	Remark
Patient	Person who needs to care or treat by medical doctor	Primary

Use Case List

Use Case Name	Description	Remark
Make appointment	call to create, change, or delete of medical appointment	Primary

Functional Modeling : Use Case Diagram



Functional Modeling : Use Case Description

Example of UC Description : **Make Old Patient Appointment (1/3)**

Use Case Name	**Make Appointment for Old Patient**	ID	UC-01	Importance Level	High
Primary Actor	Old Patient	Use Case Type		Detail, Essential	
Stakeholders	Old Patient – want to make, cjange, or cancel an appoinment Doctor – wants to endure that patient's needs are met in a timely manner				
Brief Description	This use case describes how we make an appoinment as well as changing or canceling an appointment for a previously seen patient.				
Trigger	Patient calls and asks for new appointment, or asks to cancel or change an existing appointment				
Relationships	Association: Old Patient Generalization: Manage appointment Include: - Extend: Update Patient Information				

Functional Modeling : Use Case Description

Example of UC Description : **Make Old Patient Appointment (2/3)**

Normal Flow of Events (Normal Scenario)
1. The patient contracts the office regarding an appointment. 2. The patient provides the receptionist with his or her name and address. 3. If the patient's information has changed Execute the Update_Patient_Information use case. 4. If the patient's payment arrangement has changed Execute the Make_Payment_Arrangement use case. 5. The receptionist asks to patient if he or she would like to make a new appointment, cancel or change an existing appointment. If the patient wants to make a new appointment, the S-1: new appoinment subflow is performed. If the patient wants to cancle an existing appointment, the S-2: cancel appoinment subflow is performed. If the patient wants to change an existing appointment, the S-3: change appoinment subflow is performed. 6. The receptionist provides the results of the y\transaction to the patient.

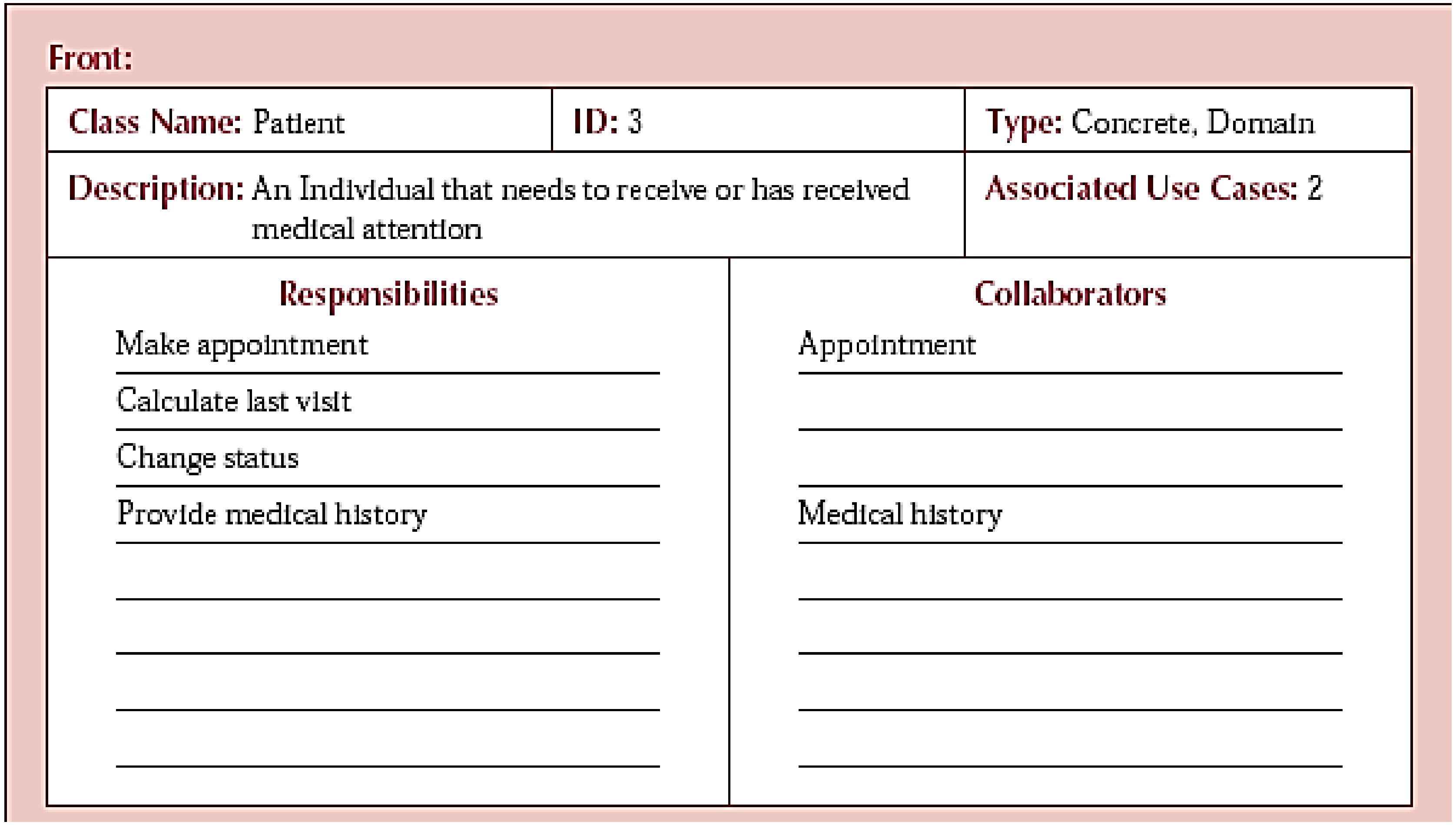
Functional Modeling : Use Case Description

Example of UC Description : **Make Old Patient Appointment (3/3)**

Sub Flow
S-1: New appointment 1. The receptionist asks for the patient for possible appointment tume. 2. The receptionist matches the patient's desired appointment time with available date and time, and schedules thre new appointment. S-2 Cancel appointment 1. The receptionist asks for the patient for the old appointment tume. 2. The receptionist finds the current appointment in the appointment file and cancels it. S-3: Change appointment 1. The receptionist performs the S-2: Cancel appointment subflow. 2. The receptionist performs the S-1: new appoinrment subflow.
Alternative / Exceptional Flow
S-1, 2a1: The receptionist proposes some alternative time based on what is available in the appointment schedule. S-1, 2a2: The patient chooses one of the proposed time or devides not to make an appointment..

Structural Modeling : Class Descrption

CRC (Class-Responsibility-Collaboration) Card (1/2)



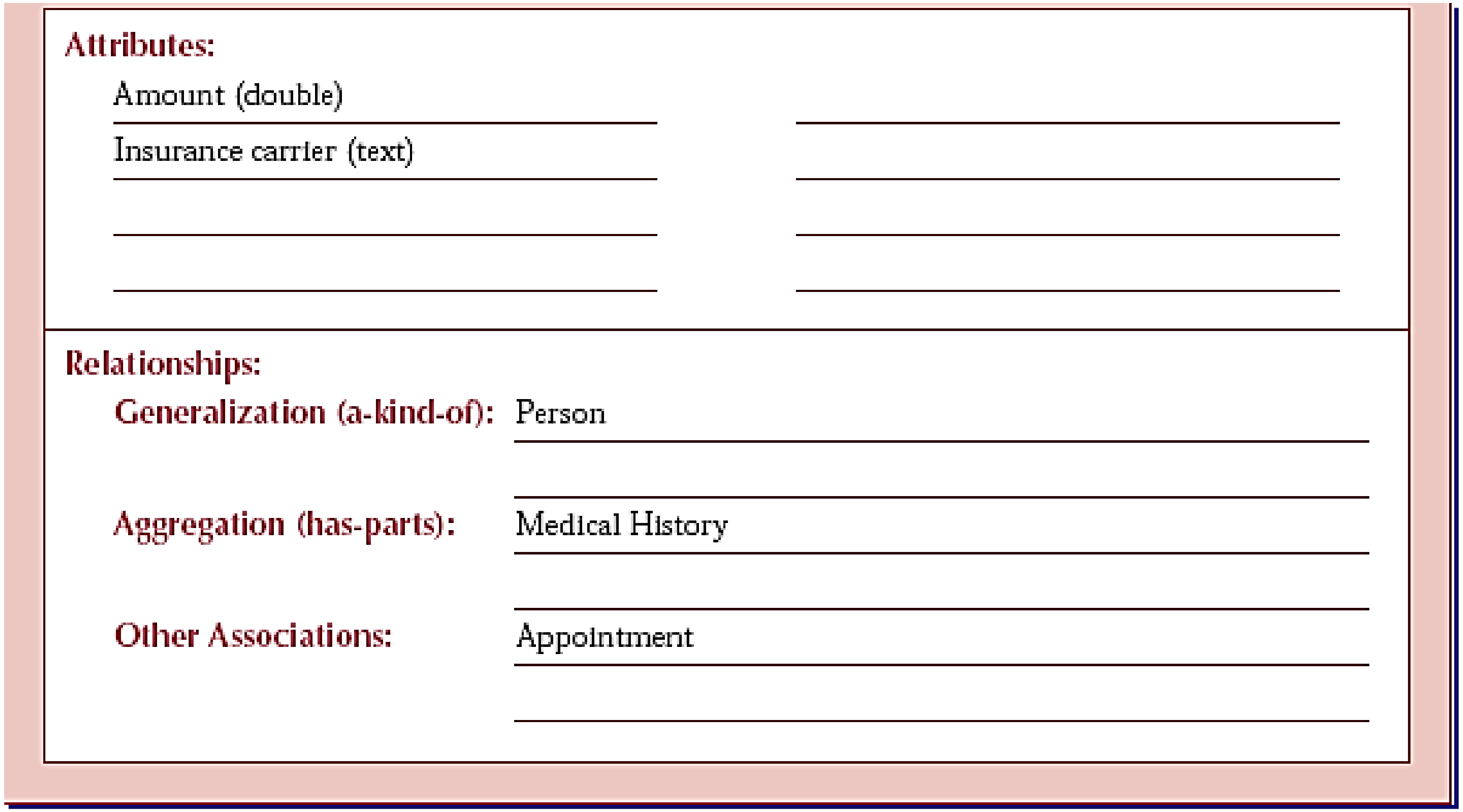
Front:

Class Name: Patient	**ID:** 3	**Type:** Concrete, Domain
Description: An Individual that needs to receive or has received medical attention		**Associated Use Cases:** 2

Responsibilities	Collaborators
Make appointment	Appointment
Calculate last visit	
Change status	
Provide medical history	Medical history

Structural Modeling : Class Descrption

CRC (Class-Responsibility-Collaboration) Card (2/2)



Attributes:

Amount (double)

Insurance carrier (text)

Relationships:

Generalization (a-kind-of):	Person
Aggregation (has-parts):	Medical History
Other Associations:	Appointment

Structural Modeling : Identifying Classes

4 approaches for Object Identification

- **Textual analysis / Brainstorming / Common object list / Patterns**

Textual analysis

- analyze the text of use-case descriptions
- identify potential objects, attributes, operations and relationship
  : Nouns suggest classes
  : Verbs suggest operations

Common Object List

- A list of objects that are common to the business domain of the system.
- Object categories
  – physical and tangible things : books, desks, office equipment, etc
  – incident : event that occur in the business domain(meetings, flights, accidents)
  – role : people play in the problem : doctor, nurse, patient, etc
  – interactions : a transaction that take place in the Biz domain(sales transaction)
  – and more such as places, organization, policies, ...

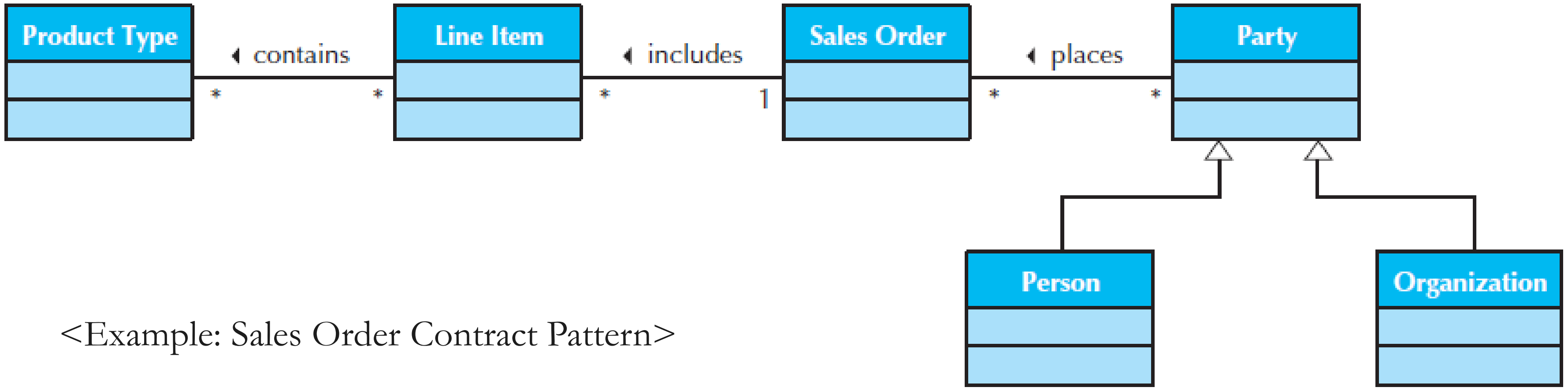
Structural Modeling : Identifying Classes

Brainstorming - people offering ideas

- Initial list of classes (objects) is developed
- Attributes, operations and relationships to other classes can be assigned in a second round

Patterns

- Useful group of collaborating classes that provide a solution to a common occurring problem
- Transactions, for example



<Example: Sales Order Contract Pattern>

Structural Modeling : Class diagram

3 basic relationships : generalization, aggregation, and association

Generalization

- enables inheritance of attributes and operations from other classes
- **a-kind-of** relationship
- relationship between superclass and subclasses

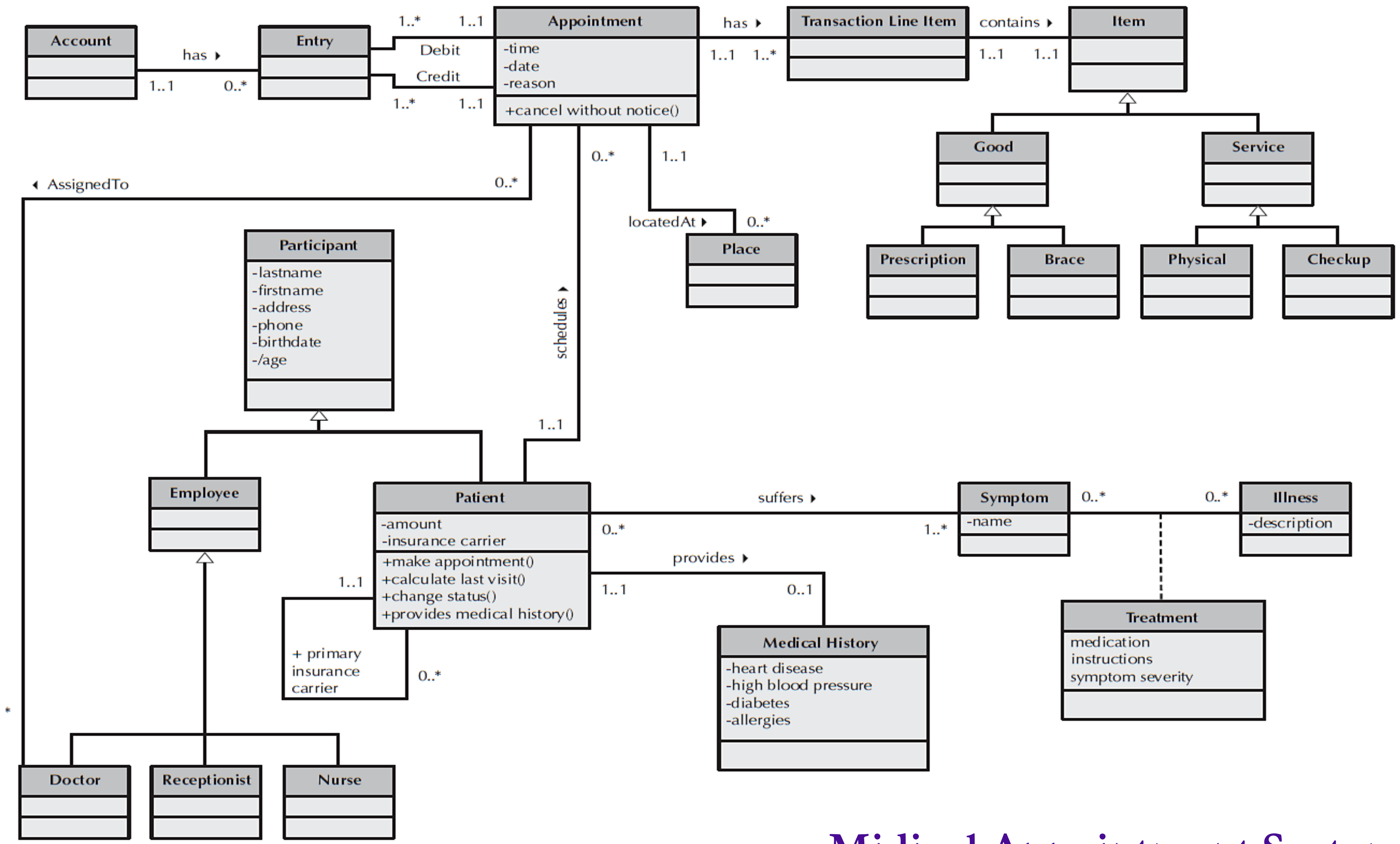
Aggregation

- composition relationship
- **a-part-of** or has-parts relationship

Association

- Miscellaneous relationships between classes
- not fit neatly into generalization or aggregation

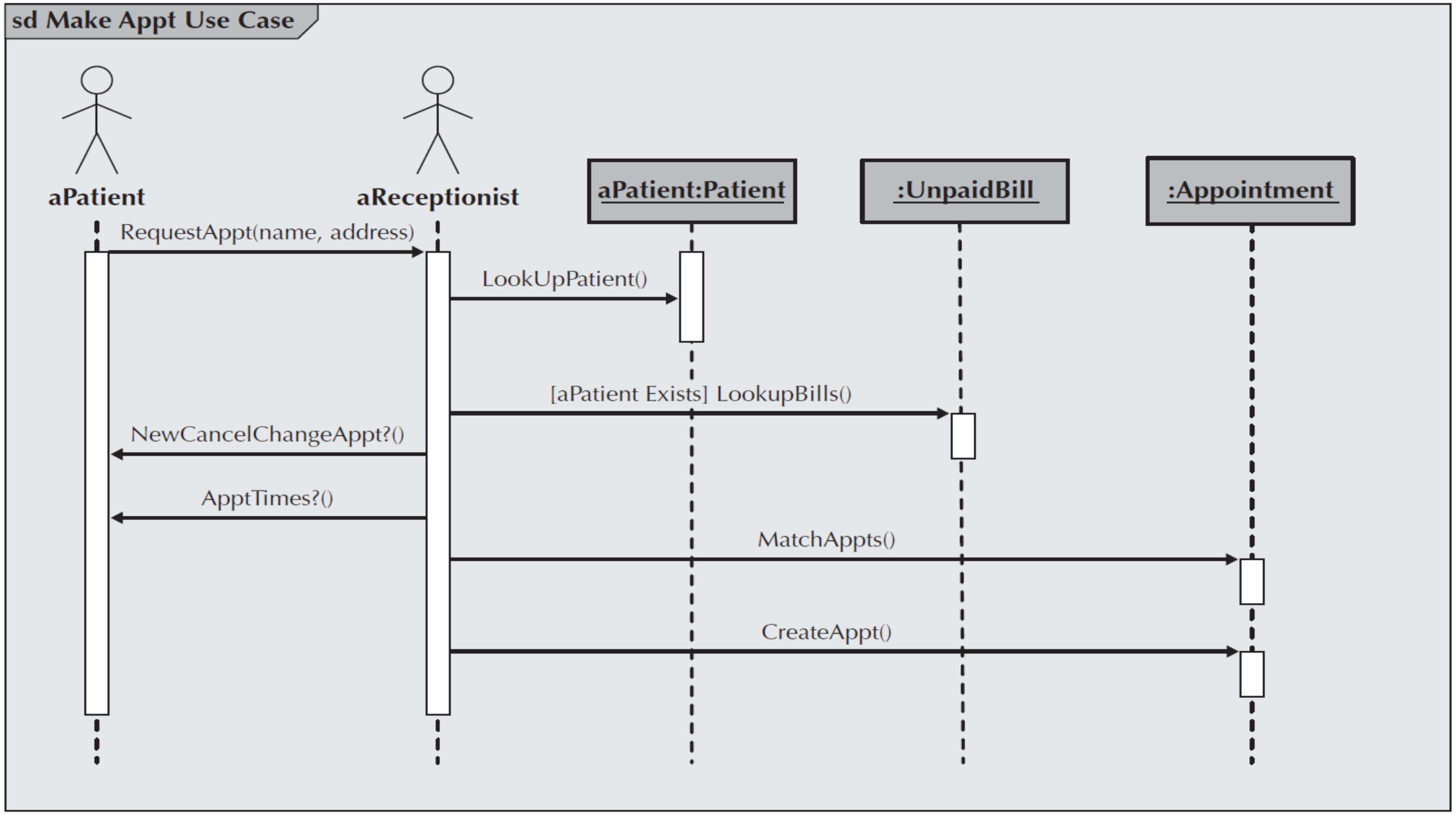
Structural Modeling : Class diagram



Midical Appointment System

Behavioral Modeling : Sequence Diagram

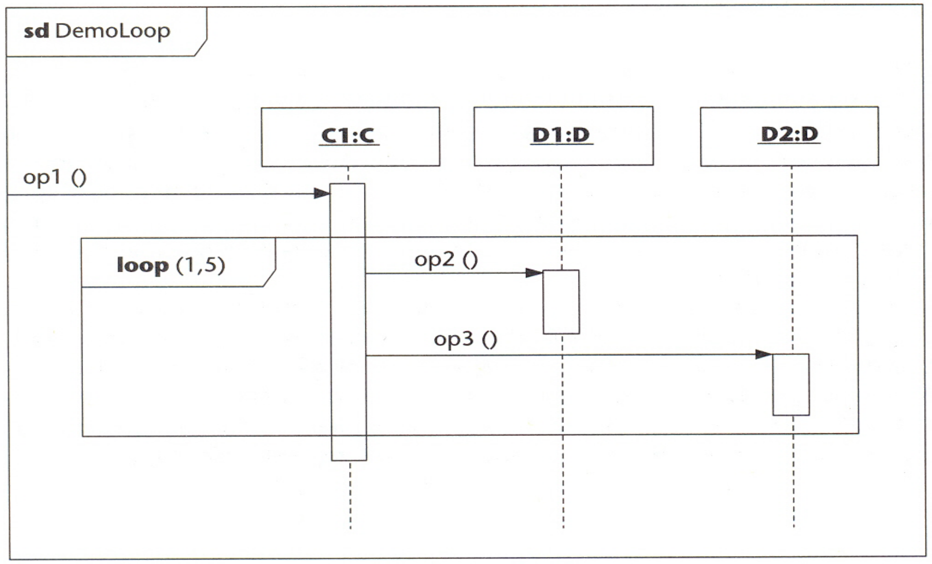
Midical Appointment System: Make Appointment Use Case

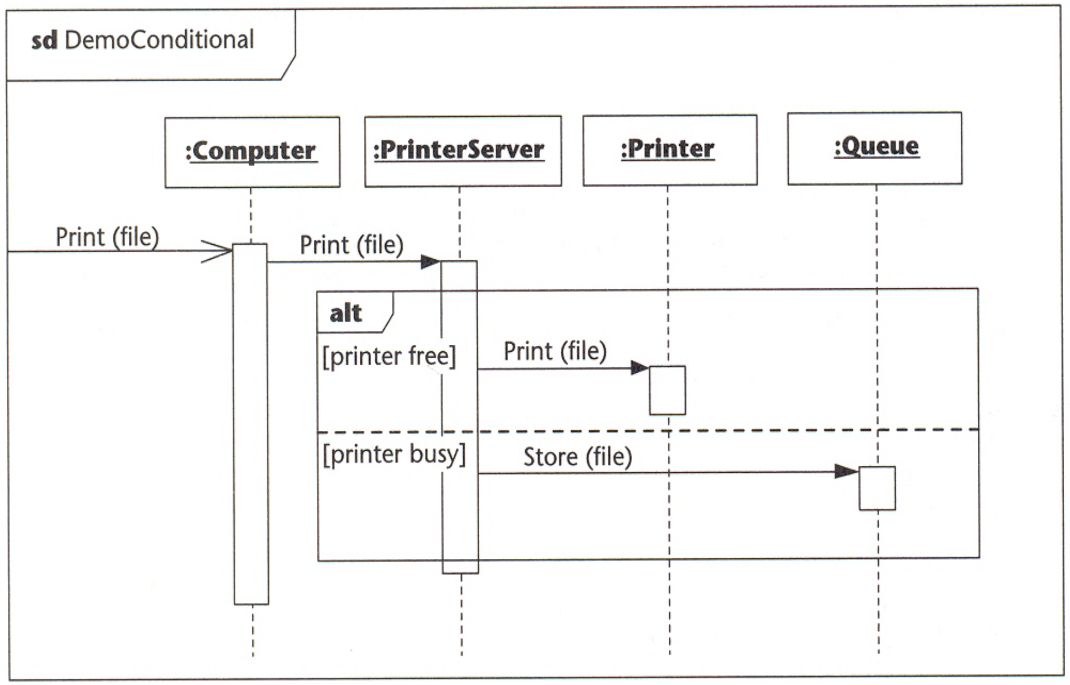


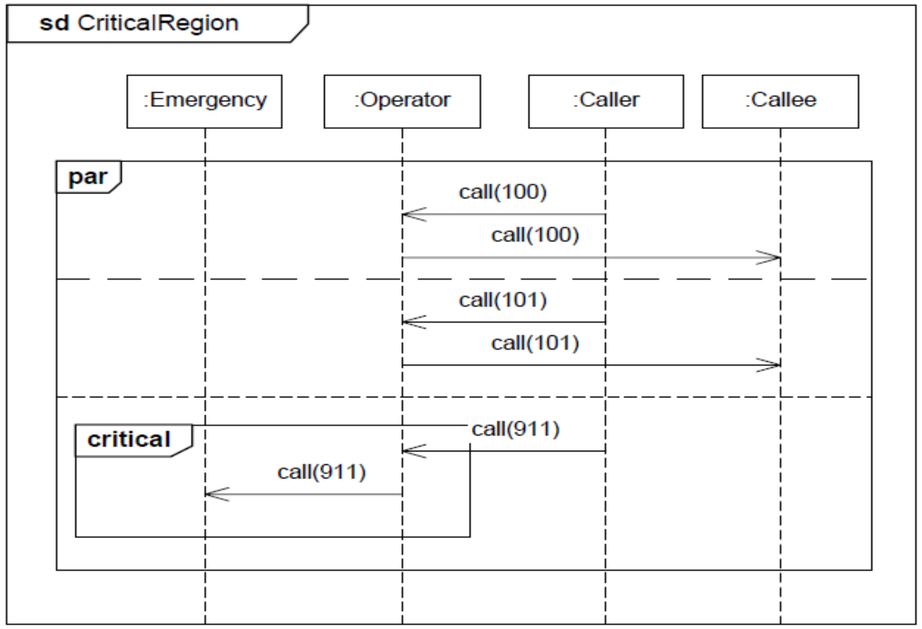
Behavioral Modeling : Sequence Diagram

Combined Fragments

- Alternatives, Option, Parallel, Loop, Critical region, etc





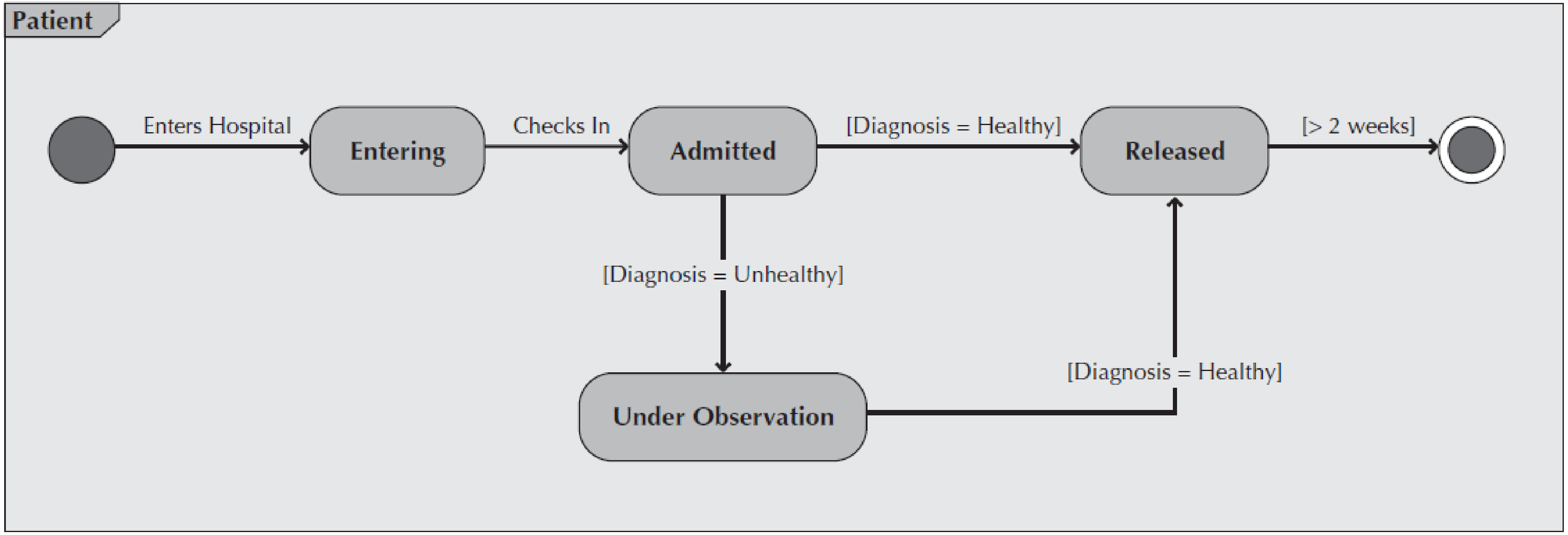


Behavioral Modeling : State Machine Diagram

Show all the possible states that objects of the class can have during a life-cycle instance

Elements

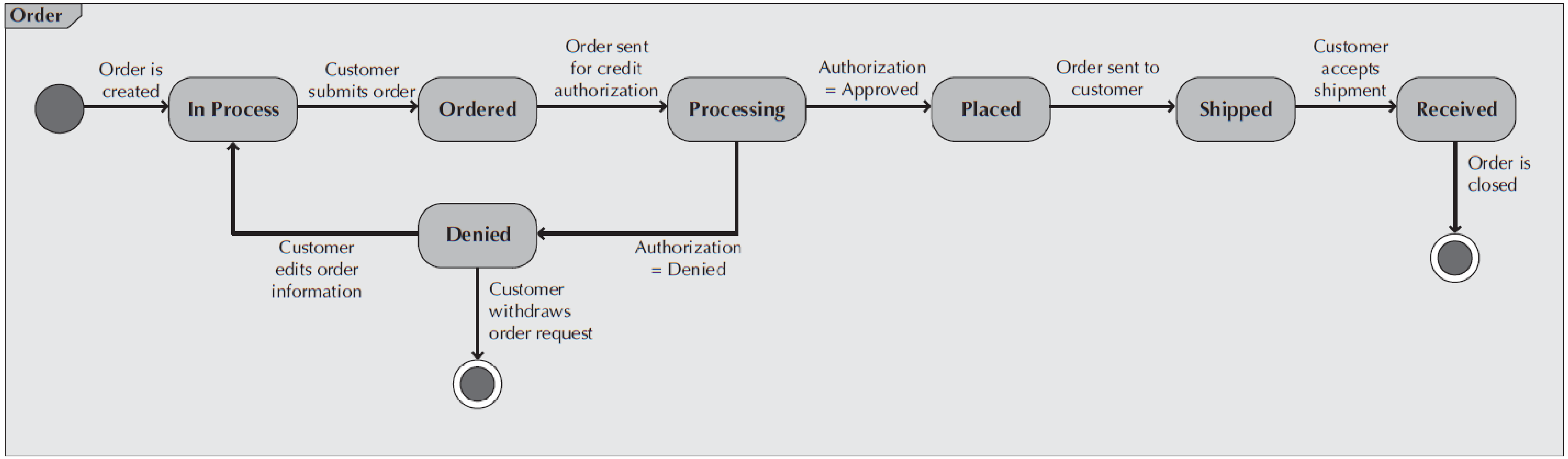
- States; Events; Transitions; Guard_Conditions; Action



Behavioral Modeling : State Machine Diagram

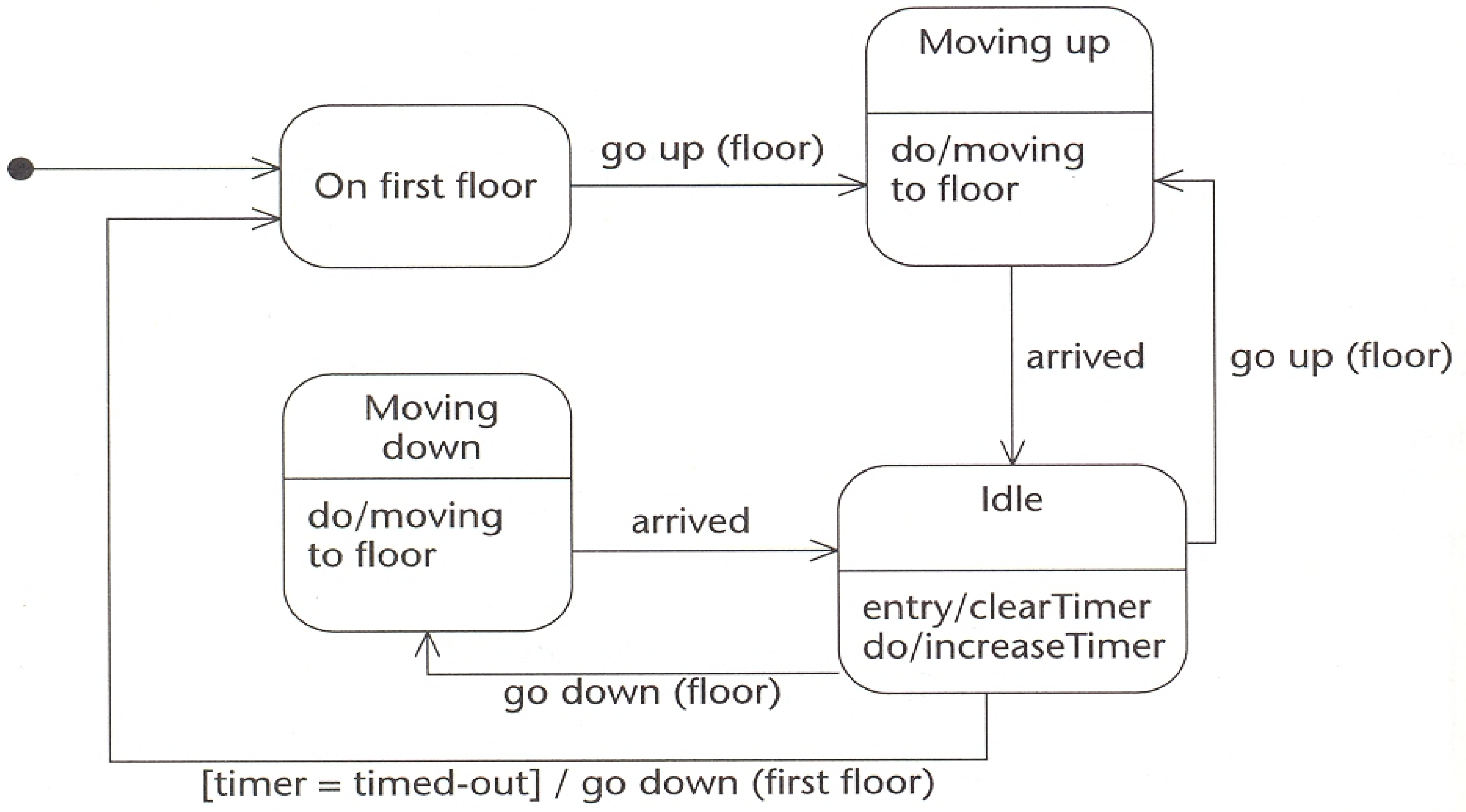
State Identification

1. The customer creates an order on the Web
2. The customer submits the order once he or she is finished
3. The credit authorization needs to be approved for the order to be accepted
4. If denied, the order is returned to the customer for changes or deletion
5. If accepted, the order is placed
6. The order is shipped to the customer
7. The customer receives the order
8. The order is closed



Behavioral Modeling : State Machine Diagram

Event [guard-condition] / Action-expression for "elevator" object



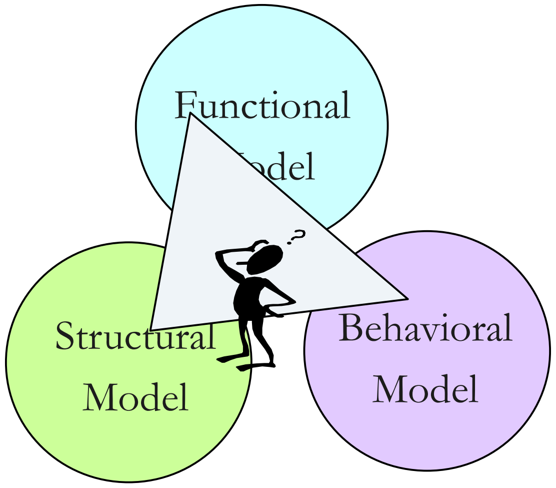
Balancing the Models

The process of ensuring the consistency among analysis models, functional, structural, and behavioral models

To ensure that the current set of analysis models faithfully represent the problem domain under consideration

Intention

- Ensuring the consistency among different models
- Verifying and validating the intersections of the analysis models using a set of rules



Balancing the Models

Balancing Functional and Structural models

- Use Case Description vs. CRC cards, Class
- Activities / Actions vs. Class responsibility, Class operations

Balancing Functional and Behavioural models

- Use Case Description vs. Sequence diagram
- Actors vs. Actors in Seq. diagram, Comm. diagram

Balancing Structural and Behavioural models

- State machine vs. Class in class diagram
- Objets on Seq. diagrams vs. Instancs (objects) of classes
- Messages on seq. diagrams, Transitions on state machines vs. __________

Contents of SRS

* The contents of document depends on the organization's standards

1. Introduction
- Purpose of this document
- Project overview
- Related documents, terms, abbreviations

2. Requirements Specification
- Functional Modeling (system context)
- Structural Modeling
- Behavioral Modeling

3. Other Requirements or Constraints
- Performance requirements
- Hardware requirements
- Exception handling
- User interface constraints
- Other constraints

4. Acceptance Criteria
- Functionality test criteria
- Performance test criteria

5. Others
- Considerable issues

6. Traceability Analysis
- Traceability matrix

7. References and Appendix

Summary and Discussion

Activities & Artifacts in Object-Oriented Analysis

- Functional Modeling
  - Activity diagram, Use Case diagram (including, UC Description)
- Structural Modeling
  - Class diagram(including CRC cards), Object diagram, Comm. diagram, etc
- Behavioural Modeling
  - Sequence diagram, State machine diagram, etc

Balancing the models

- Functional vs. Structural vs. Behavioral