Custom Text Chapter 6

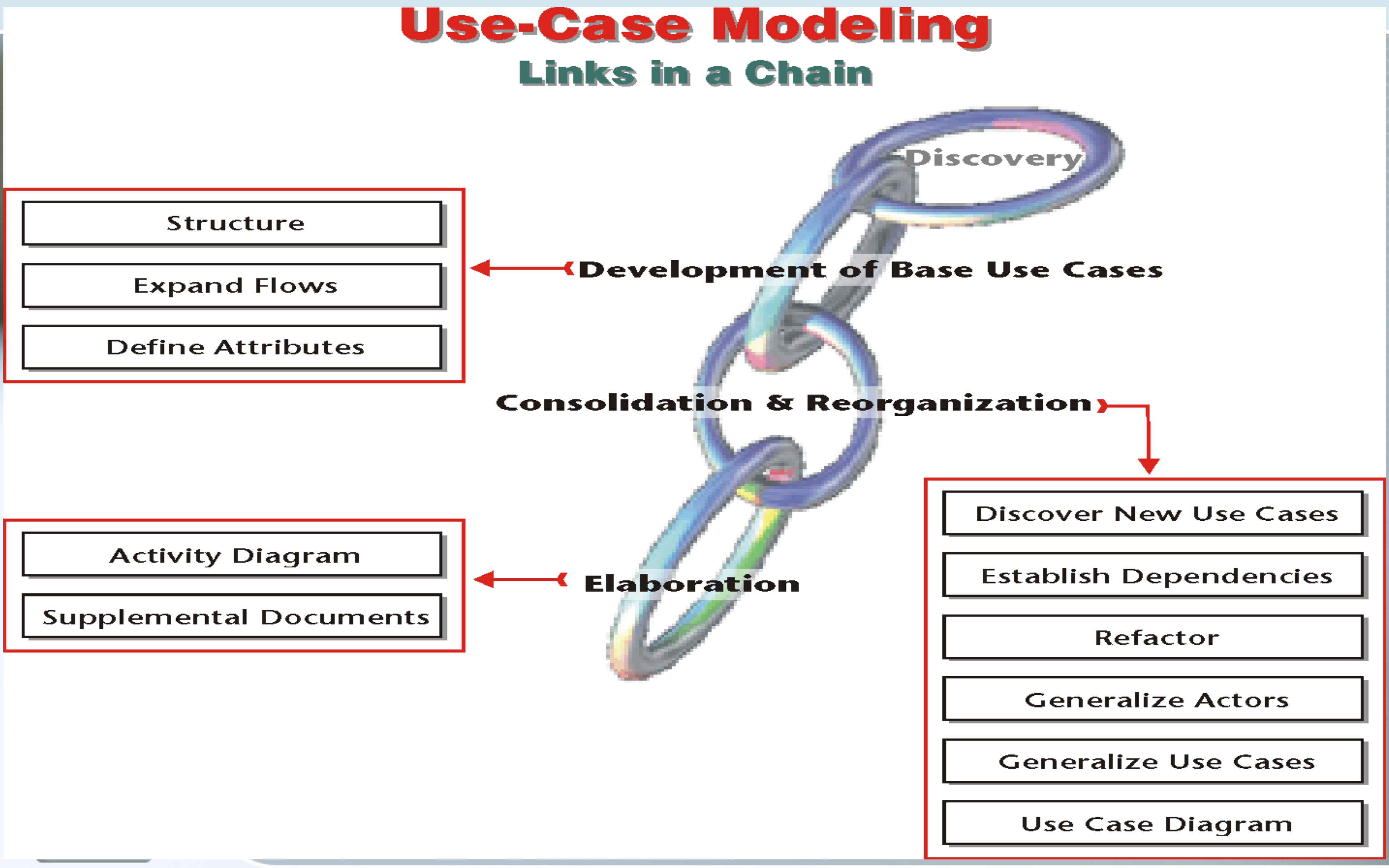
Behavioral Modeling II
Developing Use Cases

Chapter Topics

- Structuring and developing use cases through templates.
- When and how to generalize actors.
- When and how to extend the functionality of a use case.
- When and how to reuse use cases.
- When and how to generalize use cases.
- The features and the purpose of use case diagram.
- When and how to join or divide use case.
- Using activity diagram to clarify the logical flow of use cases.
- Use case modeling as a framework for development activities.
- Managing details by creating supplements to use cases.

- ✓ Inception
- ✓ Requirements Discovery
- Development
  - Analysis
    - ✓ Domain Analysis
    - **Behavioral Modeling**
      - ✓ Use Cases: The Basics
      - **Developing Use Cases**
    - Structural Modeling
    - Dynamic Modeling
  - Design
  - Implementation

A Framework for the Development



Develop Base Use Cases

- What a “base” use case is?
  - A base use case is a fully formed, structured use case which serves as a *base* to develop other analysis and design artifacts.

The Template

- The template structures use cases by providing well-defined and ordered fields.

Use Case Template

- Please refer to able 7.1 on page 210 in the text book.

Template Fields

- Template fields represent the building blocks of the use cases, joined in a predefined, orderly manner.

Template Fields

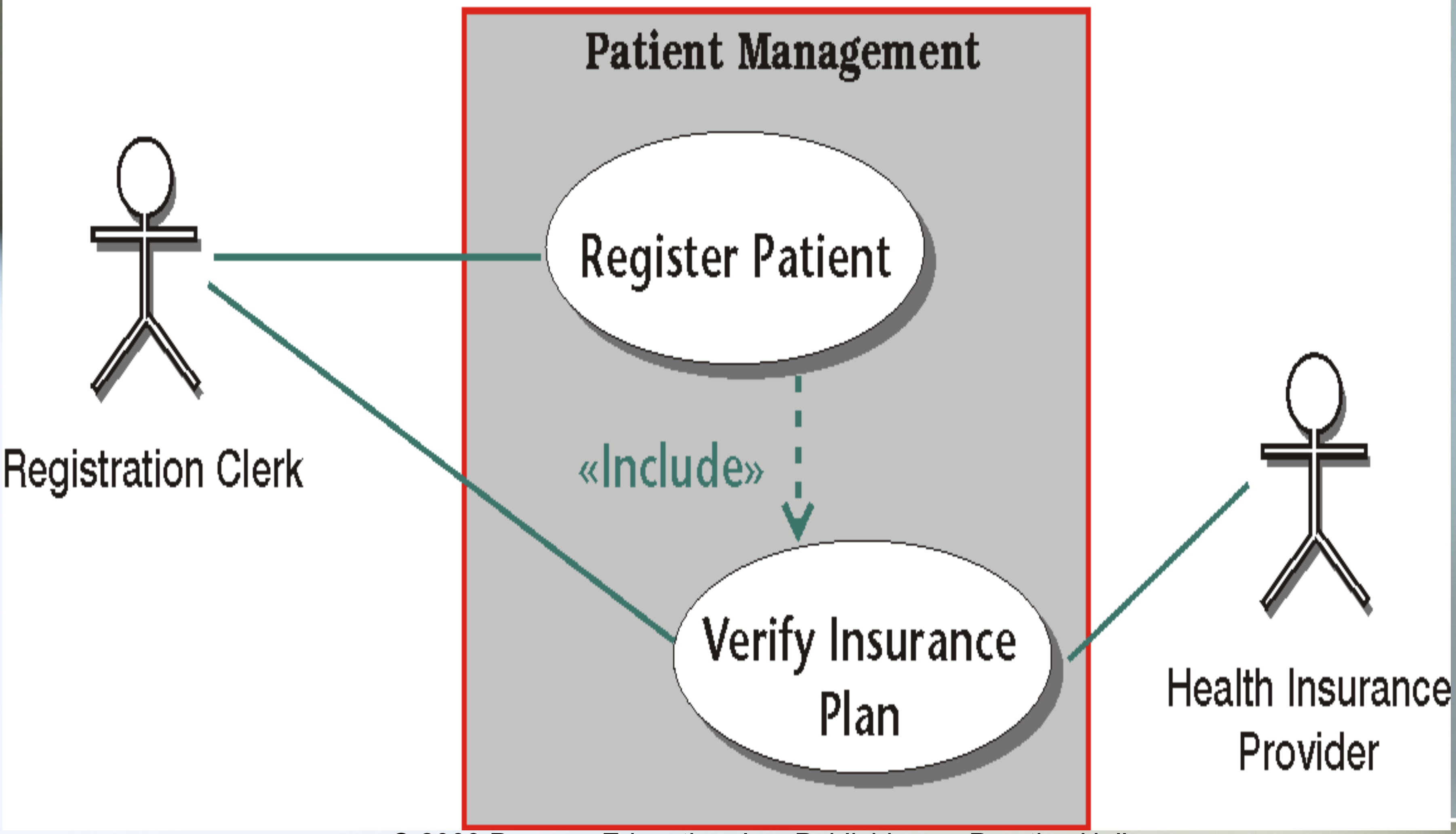
- Name
  - embodies the goal that the use case wants to accomplish.
- ID
  - is *unique* numeric identifier for the use case.
- Scope
  - boundaries of the use case— defined by the system or the subsystem to which it belongs.
- Priority
  - decides the order of design and implementation for use cases.

Template Fields

- Summary
  - a long version of the use case name and a short version of the scenario.
- Primary actor
  - is the actor whose goal identifies and drives the use case.
- Supporting actor
  - assist the primary actor in achieving the goal of the use case.

The Supporting Actor

Helping the Primary Actor to Reach the Goal



Template Fields

- Stakeholder
  - any entity, human or otherwise, who has an interest in the outcome of the use case.
- Precondition
  - defines the state of the system before a use case can start; post-condition defines the state of the system after a use case is complete.
- Trigger
  - the event that starts the use.
- A flow
  - an ordered set of activities that occur as the actors and the system attempt to reach a goal.

Normal Flow

- Normal flow is the best-case scenario

Conduct ATM Transaction	**Normal Flow:**	**1.** Customer inserts the bank card. **2.** Customers enters password. **3.** System verifies password. **4.** System presents a list of transaction types that the customer may conduct. **5.** Customer selects a type of transaction.

Sub-Flows

- Sub-flows identify the details of the steps in the normal flow

Register Patient	**Normal Flow:**	**1.** The registration clerk enters or updates personal data.
	Sub Flows:	**1.1** The registration clerk enters the Social Security Number of the new patient. **1.2** The registration clerk enters or updates patient's address. **1.3** The registration clerk enters or updates patient's phone number. **1.4** The registration clerk enters or updates the name, the address and the phone number of the patient's closest relative.

Alternate Flow and Exceptions

- Alternate steps identify remedies; exceptions signify failure

Receive Patient	**Alternate Flow/ Exceptions:**	**3.a** Patient is new. Reception clerk directs the patient to registration… **3.b**Patient is not new but personal or insurance data has changed. Reception clerk directs the patient to registration… **3.c**Patient has lost the hospital ID card. Reception clerk directs the patient to registration…

Non-Behavioral Requirements

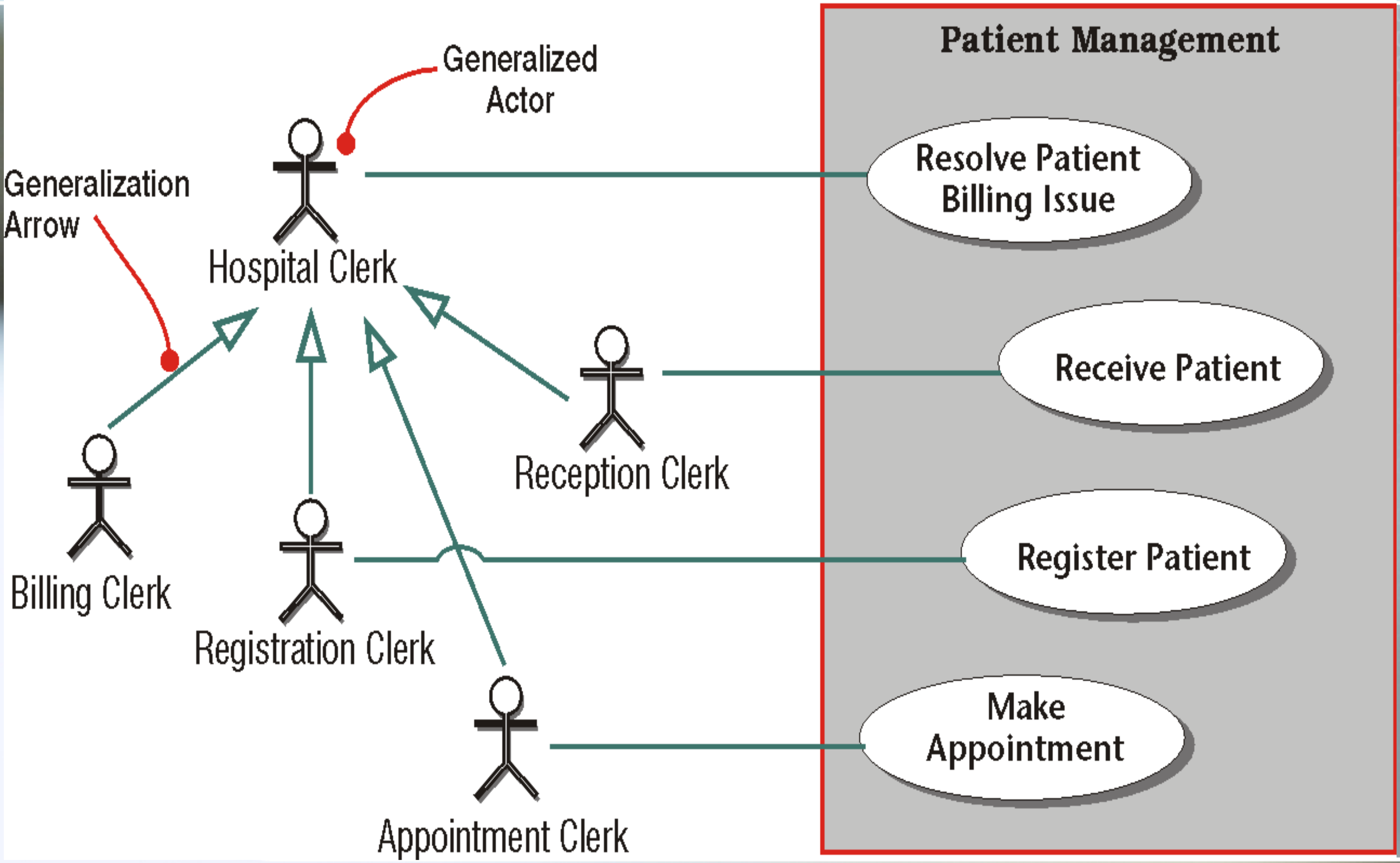
- Only when a non-behavioral requirements applies to a specific use case, the requirement is specified in the template.

Template Fields

- Open Issues
  - questions that must be resolved before the use case can be judged as complete.
- Audit fields
  - help us to keep track of the evolution of the use case.
- Custom Fields
  - specifies an attribute or requirement that is specific to one use case or a set of use cases within the system.

Actor Generalization

Creating a 'Super-Role'



Actor Dictionary

Actor	Description	Abstract	Use Case (s)
Appointment Clerk	Makes appointment for the patient to receive medical service.		Make Appointment
Billing Clerk	Maintains patient billing.		Enter Bulk Payment
Hospital Clerk	Generalizes: ▪ Appointment Clerk ▪ Billing Clerk ▪ Reception Clerk ▪ Registration Clerk	✓	Resolve Patient Billing Issue
Reception Clerk	Receives patient on arrival at the hospital. Verifies registration. Arranges for the patient to receive medical service.		Receive Patient
Registration Clerk	Enters or updates patient's personal and payment data. Issues a hospital card, if necessary.		Register Patient

Dependencies: Include and Extend

- An extend relationship is one in which a use case is created to extend the functionality of a base use case.

Register Patient	**Alternate Flow/ Exceptions:**	**2.a** The patient is not new and insurance data has not changed. Registration clerk does not update the insurance data by default. **2.b** The patient wants to pay the entire bill or the co-payments by a credit card. Registration clerk verifies the credit card (Extend: 142 - Verify Credit Card) and records credit card information.

Include Relationship

- An include relationship is one in which one use case uses the functionality of another, independent, use case.

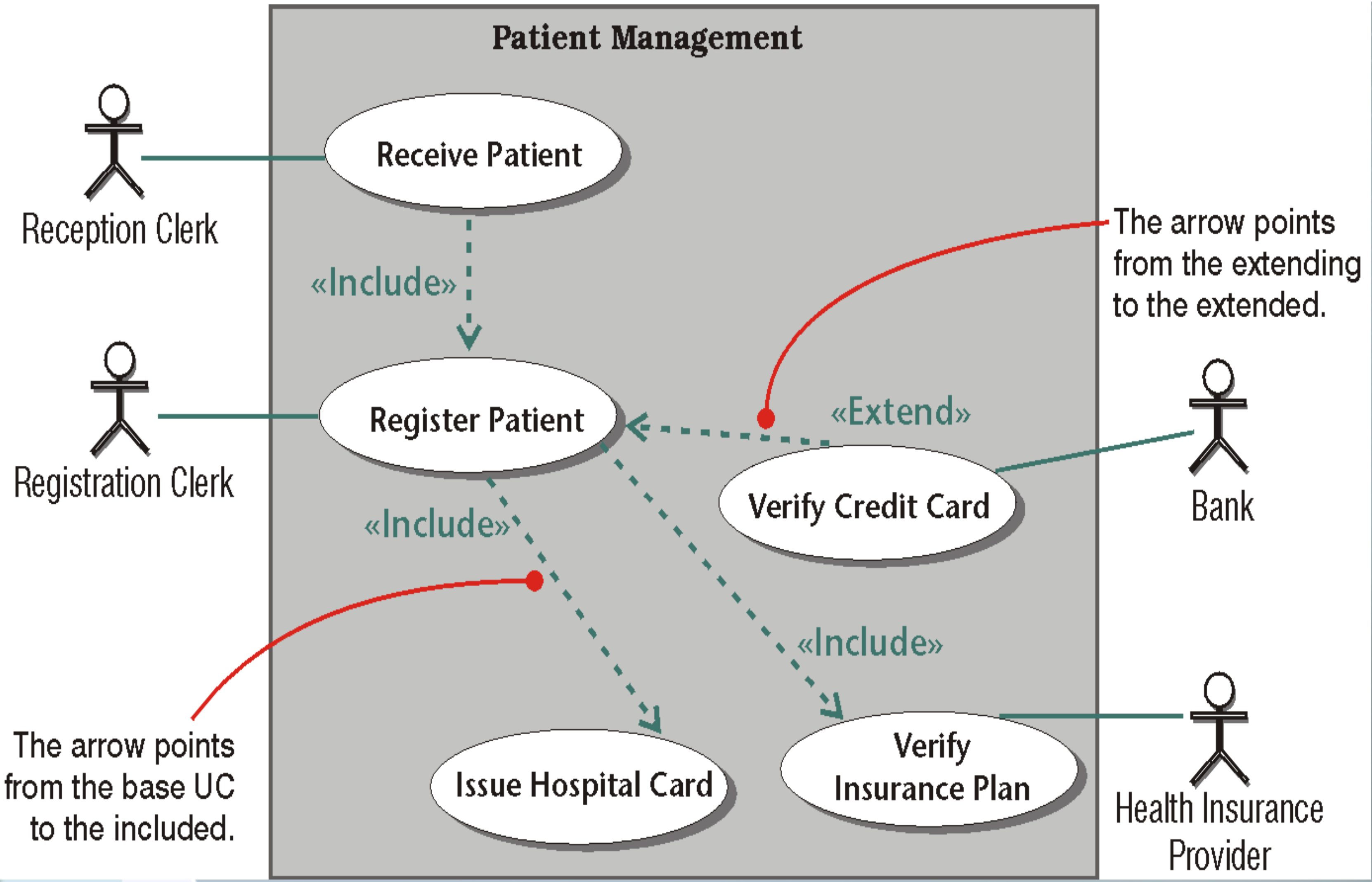
Receive Patient	**Normal Flow:**	… **3.** Reception clerk verifies that patient has been registered and registration is valid. …
	Alternate Flow/ Exceptions:	**3.a** Patient is new. Reception clerk directs the patient to registration. (<u>Include:</u> 140 - Register Patient.) …

Use Case Diagram for Dependencies

- In a use case diagram, dependency type is indicated by the direction of an arrow.

Use Case Dependencies

Include & Extend



Base UC	Arrow's Direction	Referenced UC
Extended UC Register Patient	←	**Extending UC** Verify Credit Card
Including UC Receive Patient	→	**Included UC** Register Patient

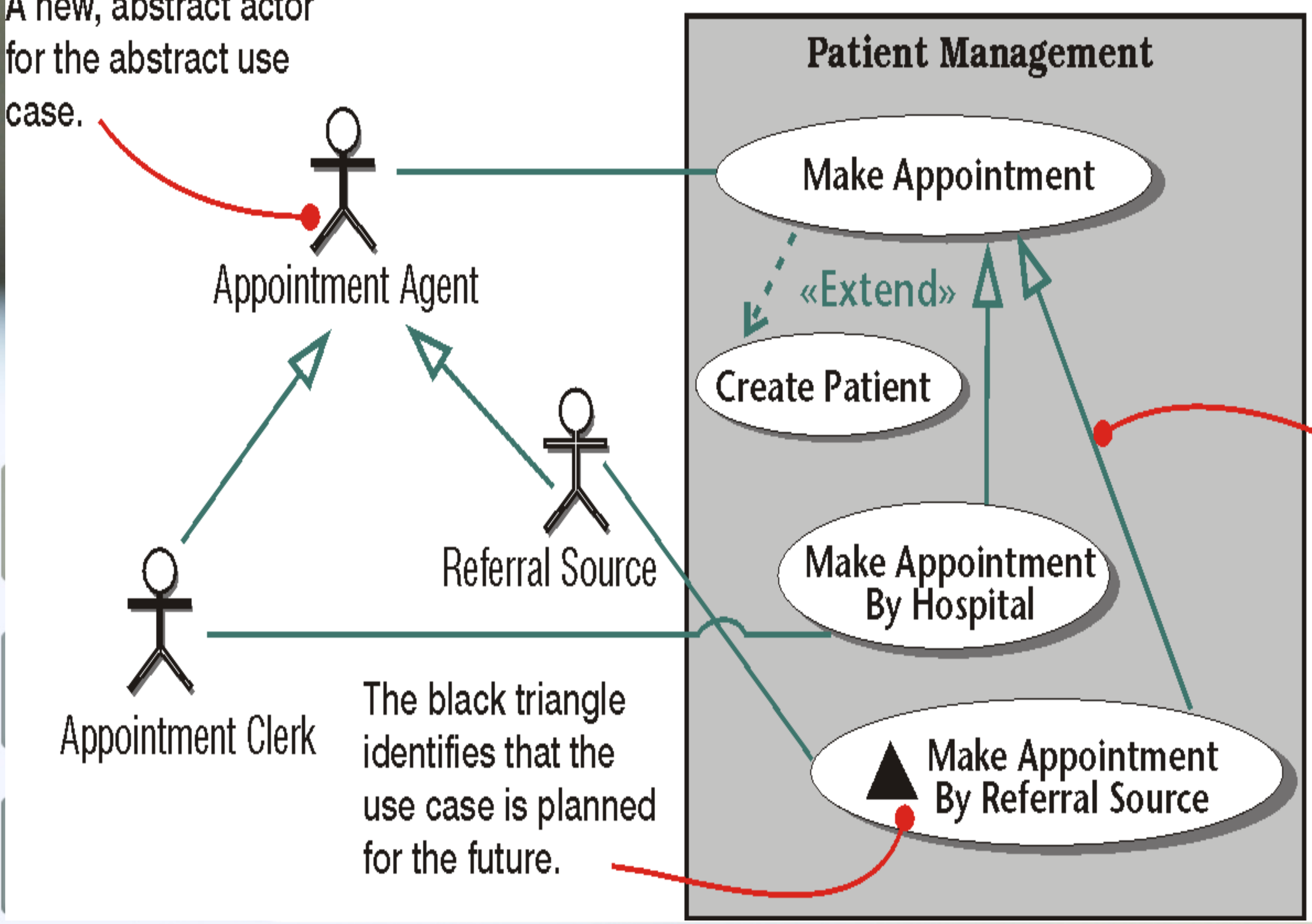
Use Case Generalization

- We generalize use cases when the they achieve the same goal by different means.

Use Case Generalization

When the Same Goal Is Achieved by Different Means

A new, abstract actor for the abstract use case.



The arrow points from the child use case to the parent use case.

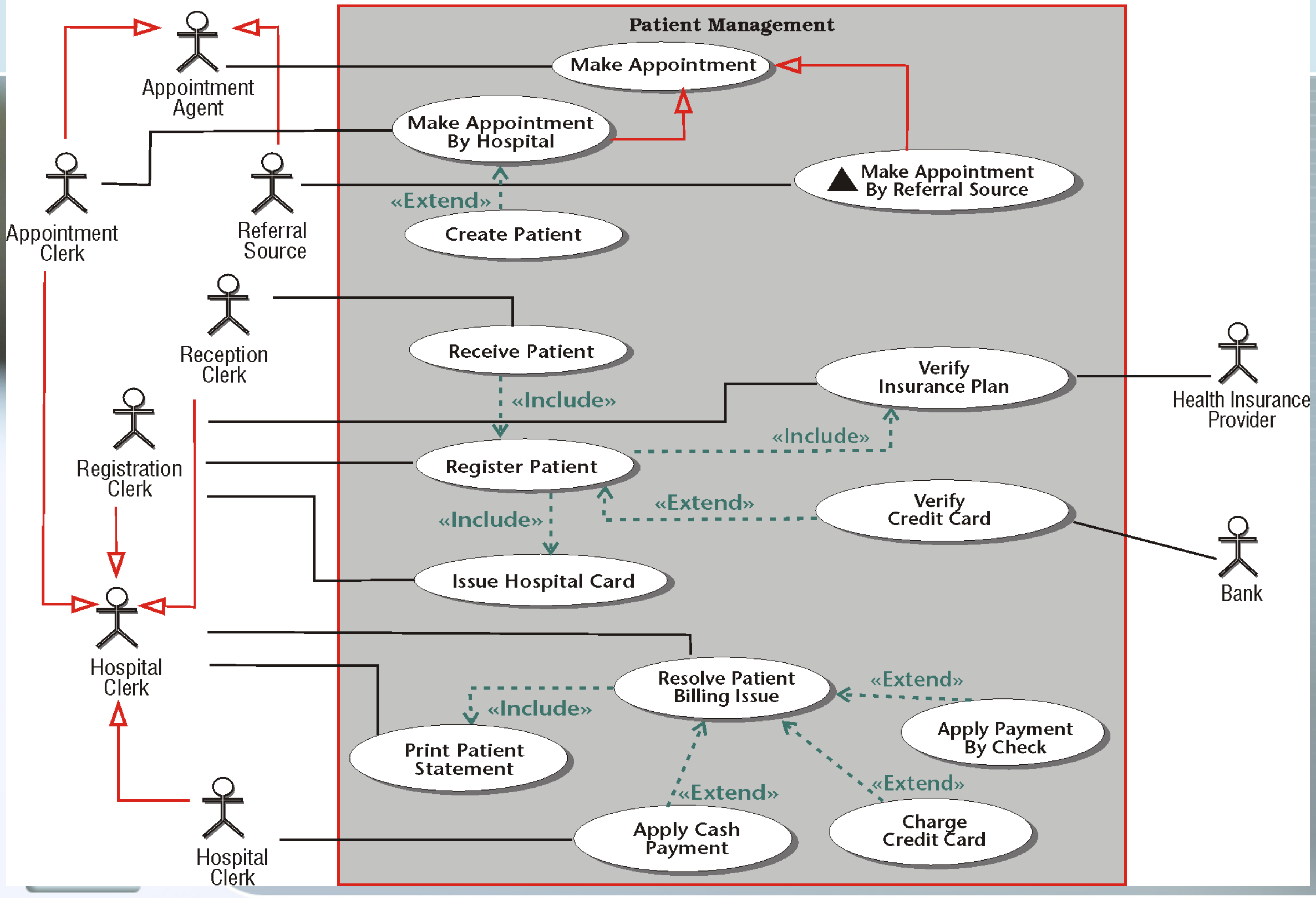
The black triangle identifies that the use case is planned for the future.

Use Case Diagram

- Use case diagram is a meta-model that portrays associations among actors, use cases and the system.

Use Case Diagram

A Meta-Model for the 'Big Picture'



Separating and Joining Use Cases

❶ We delineate them.

❷ We divide them into more use cases.

❸ We combine them.

Delineating Use Cases

- One use case must have one primary actor, one useful goal and one system.

Dividing Use Cases

- New requirements or the challenge of complexity may demand that a use case be divided:
  - Vertical division is necessary if the use case has too many parallel steps.
  - Horizontal division is necessary if the flow is too complex or the building blocks of the use case lack unity.

Refactoring

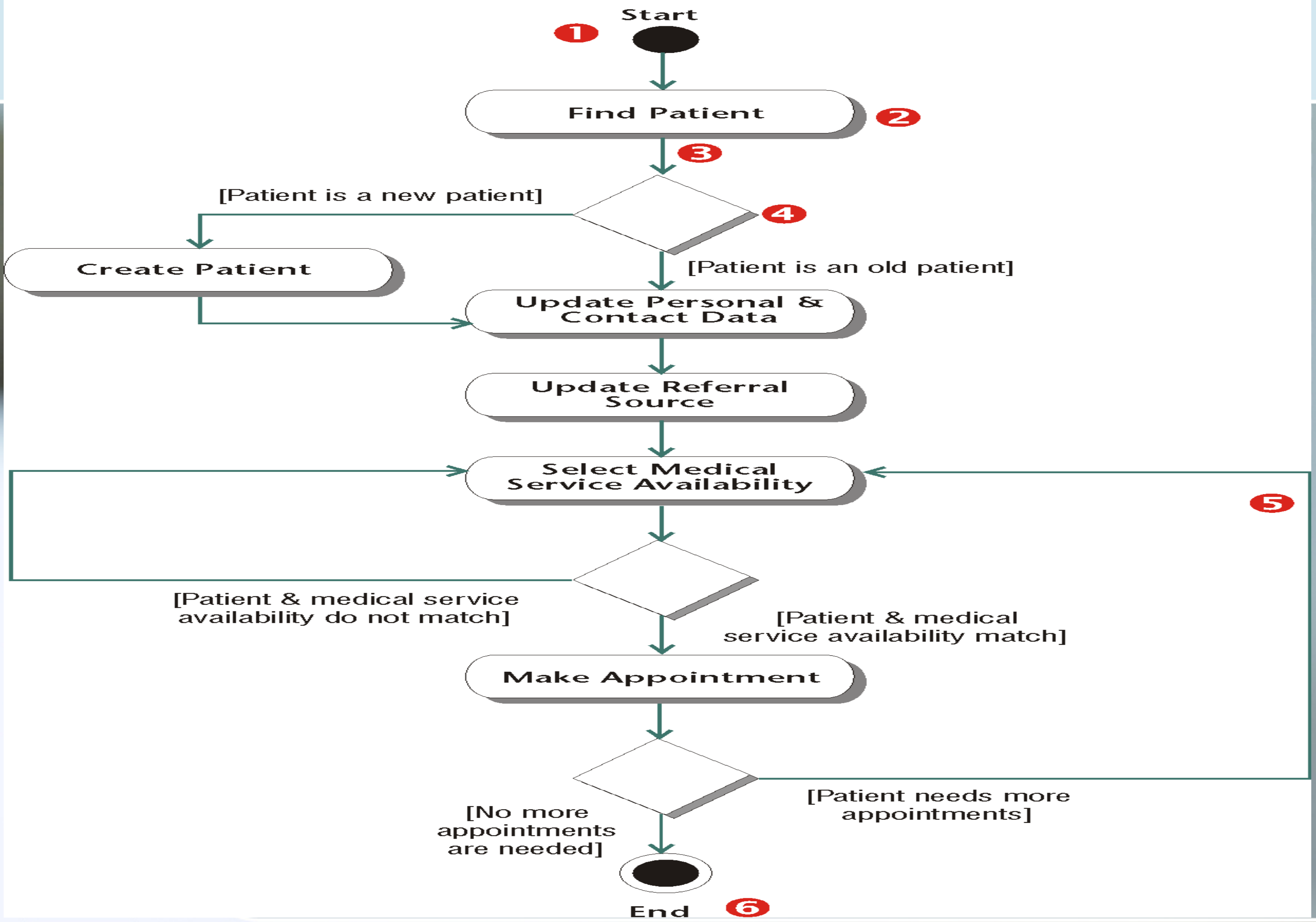
- Refactoring abstracts and reorganizes common behavior among use cases into new use cases.

Activity Diagram

- Activity diagram depicts the flow from activity to activity. It presents a visual, dynamic view of the system and its components.

Activity Diagram

The Logical Flow of Make Appointment



Start
1
Find Patient
2
3
[Patient is a new patient]
4
Create Patient
[Patient is an old patient]
Update Personal & Contact Data
Update Referral Source
Select Medical Service Availability
5
[Patient & medical service availability do not match]
[Patient & medical service availability match]
Make Appointment
[No more appointments are needed]
[Patient needs more appointments]
End
6

The Building Blocks of Activity Diagram

- Refer to Table 7.4 on page 240 in the text book.

Uses of Use Cases

- Use cases provide a crucial framework for analysis, design, implementation and deployment activities.

Uses of Use Cases

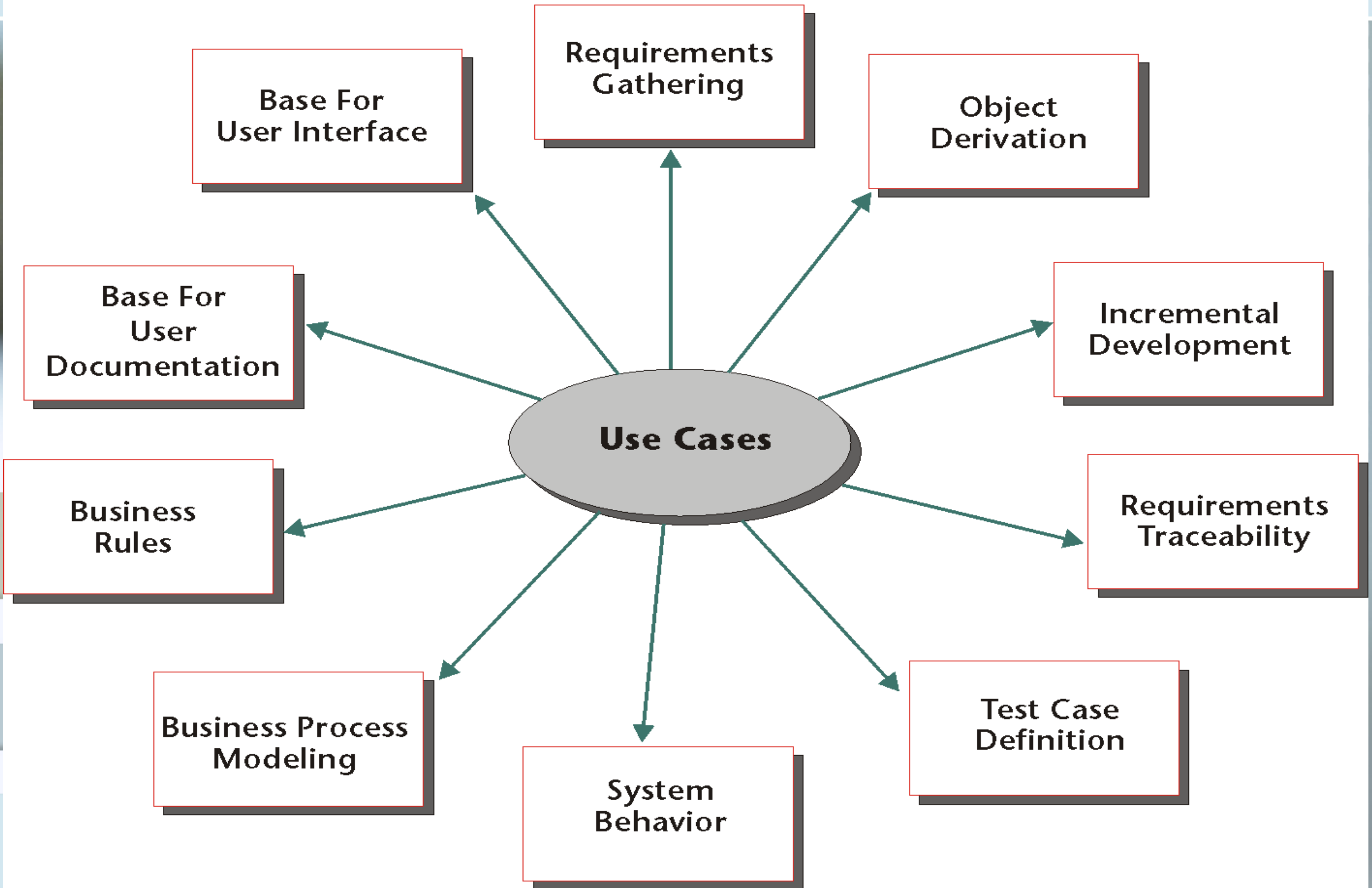
- Requirements Gathering
  - Use cases provide the base tools for gathering requirements within a meaningful context.
- Requirements Traceability
  - Use cases and their supporting documents are the prime sources for tracing requirements.
- Business Rules
  - Use cases are the framework for gathering business rules.
- System Behavior
  - The external behavior of any open system can be captured effectively through use cases.

Uses of Use Cases

- Object Derivation
  - By launching a cycle of gathering requirements from the use cases, we can arrive at many of the objects that would form the structure of the system.
- Incremental Development
  - By prioritizing use cases and their dependencies, we can build a system incrementally.
- Base for User Interface
  - Use cases describe the basics messages that the actor and the system must exchange to achieve a goal.
- Test Case Definition
  - Use cases are the conceptual blueprints for functional test cases.
- Base for User Documentation
  - Use cases are built to describe the interaction between a user type and a system.
- Business Process Modeling
  - Use cases can be used to model business processes, prior to, after, or independent from an information system.

Uses of Use Case Modeling

A Framework for Development



Next: Structural Modeling

- The basic building blocks of an information systems are objects.
- An object is created from a mold called class.
- To make objects we have to make classes, and this is the starting point of the next chapter.