Requirements Engineering

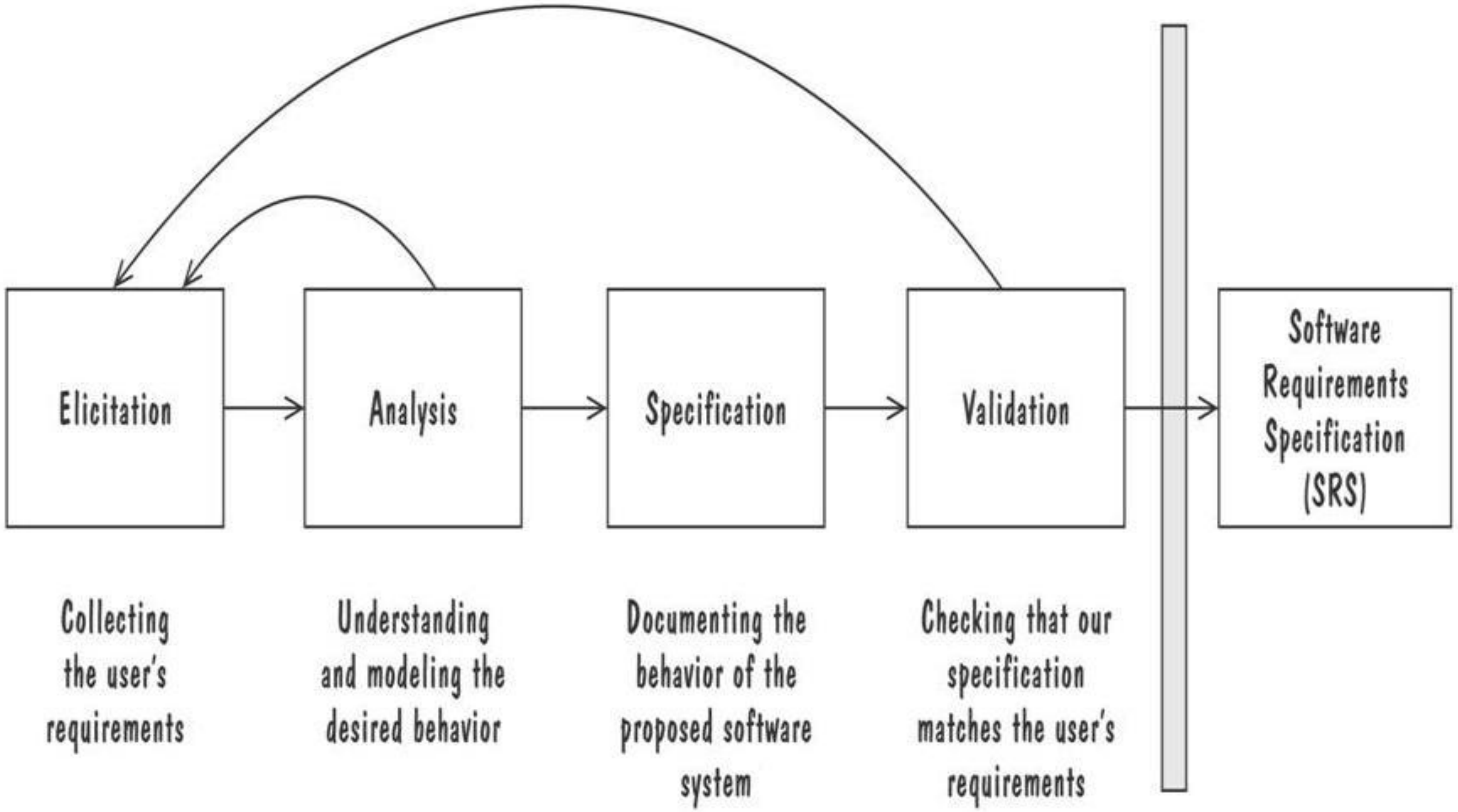
Requirements Engineering-I

The requirements engineering process can be described in seven distinct steps:

- Inception—ask a set of questions that establish …basic understanding of the problem the people who want a solution the nature of the solution that is desired, and the effectiveness of preliminary communication and collaboration between the customer and the developer
- Elicitation—elicit requirements from all stakeholders
- Elaboration—create an analysis model that identifies data, function and behavioral requirements
- Negotiation—agree on a deliverable system that is realistic for developers and customers

Requirements Engineering-I

- Specification—can be any one (or more) of the following: A written document, A set of models, A formal mathematical, A collection of user scenarios (use-cases) and A prototype
- Validation—a review mechanism that looks for errors in content or interpretation areas where clarification may be required missing information in consistencies (a major problem when large products or systems are engineered)conflicting or unrealistic (unachievable) requirements.
- Requirements management


Elicitation
Analysis
Specification
Validation
Software Requirements Specification (SRS)
Collecting the user's requirements
Understanding and modeling the desired behavior
Documenting the behavior of the proposed software system
Checking that our specification matches the user's requirements

Process for Capturing Requirements

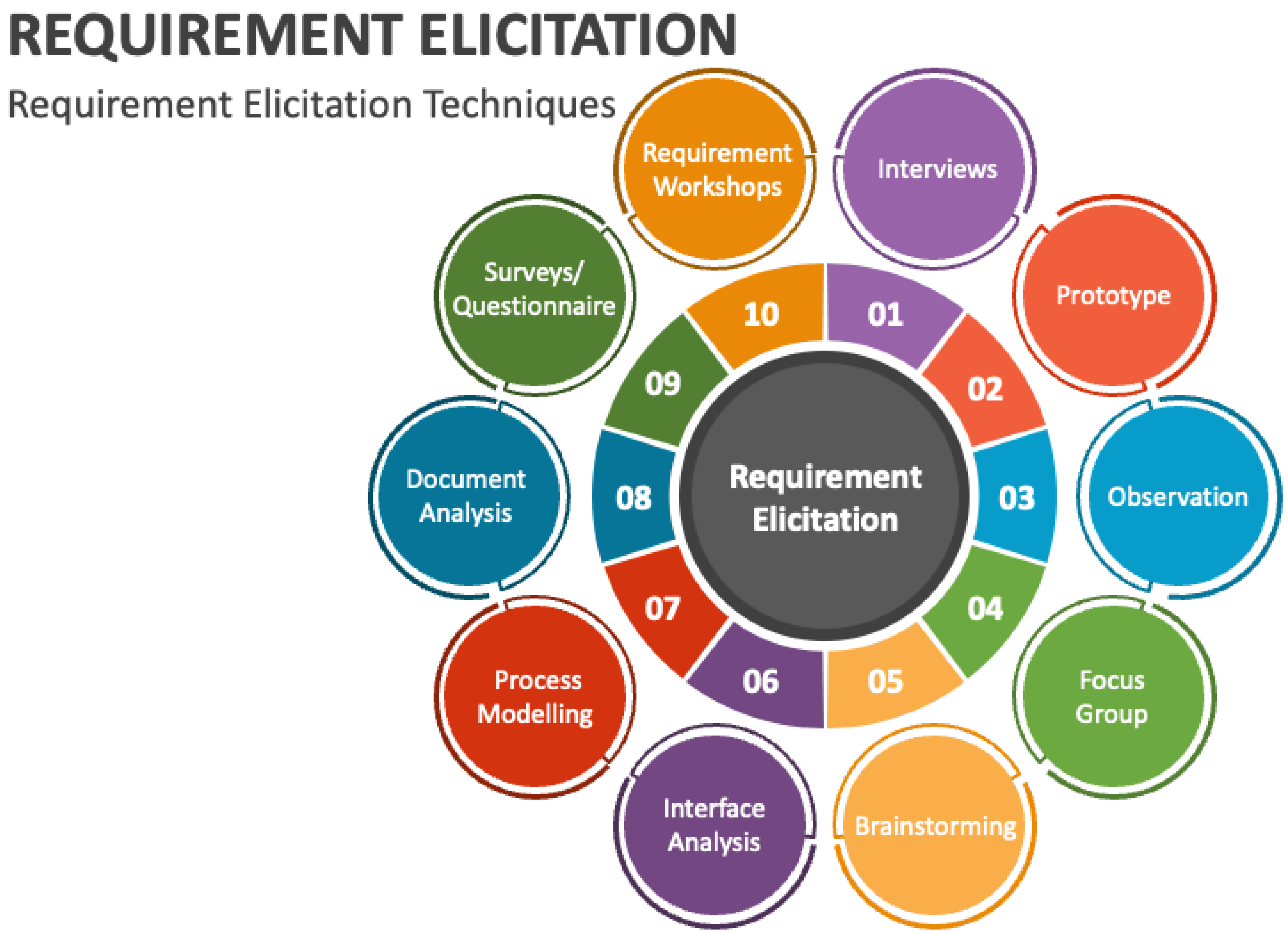
Inception

- Identify: all stakeholders, measurable benefits of successful implementation, possible alternatives
- Ask questions stepwise, as early as possible, first meeting/encounter
- Possible questions at 1st step (stakeholders, overall goals and benefits):
  - Who is behind the request for this work?
  - Who will use this solution?
  - What will be the economic benefit of a successful solution?

Inception

- Possible questions at 2nd step (detailed understanding and customer perception about the solution):
  - What problems(s) will this solution address?
  - Can you show me (or describe) the business environment in which the solution will be used?
  - How do you characterize the ‘good’ output?
- Possible questions at 3rd step (effectiveness of communication):
  - Are you the right person to answer these questions?
  - Are my questions relevant to the problem that you have?
  - Can anyone else provide additional information?

Eliciting Requirements



Elaboration

- Analyze, model, specify..

The requirements modeling action results in one or more of the following types of models:

- Scenario-based models of requirements from the point of view of various system “actors.” (use case, activity, sequence)
- Class-oriented models that represent object-oriented classes (attributes and operations) and how classes collaborate to achieve system requirements. (class diagram, CRC cards)
- Behavioral models that depict how the software reacts to internal or external “events.” (state diagram, activity)
- Data models that depict the information domain for the problem.
- Flow-oriented models that represent the functional elements of the system and how they transform data as they move through the system.

Scenario Based Modeling

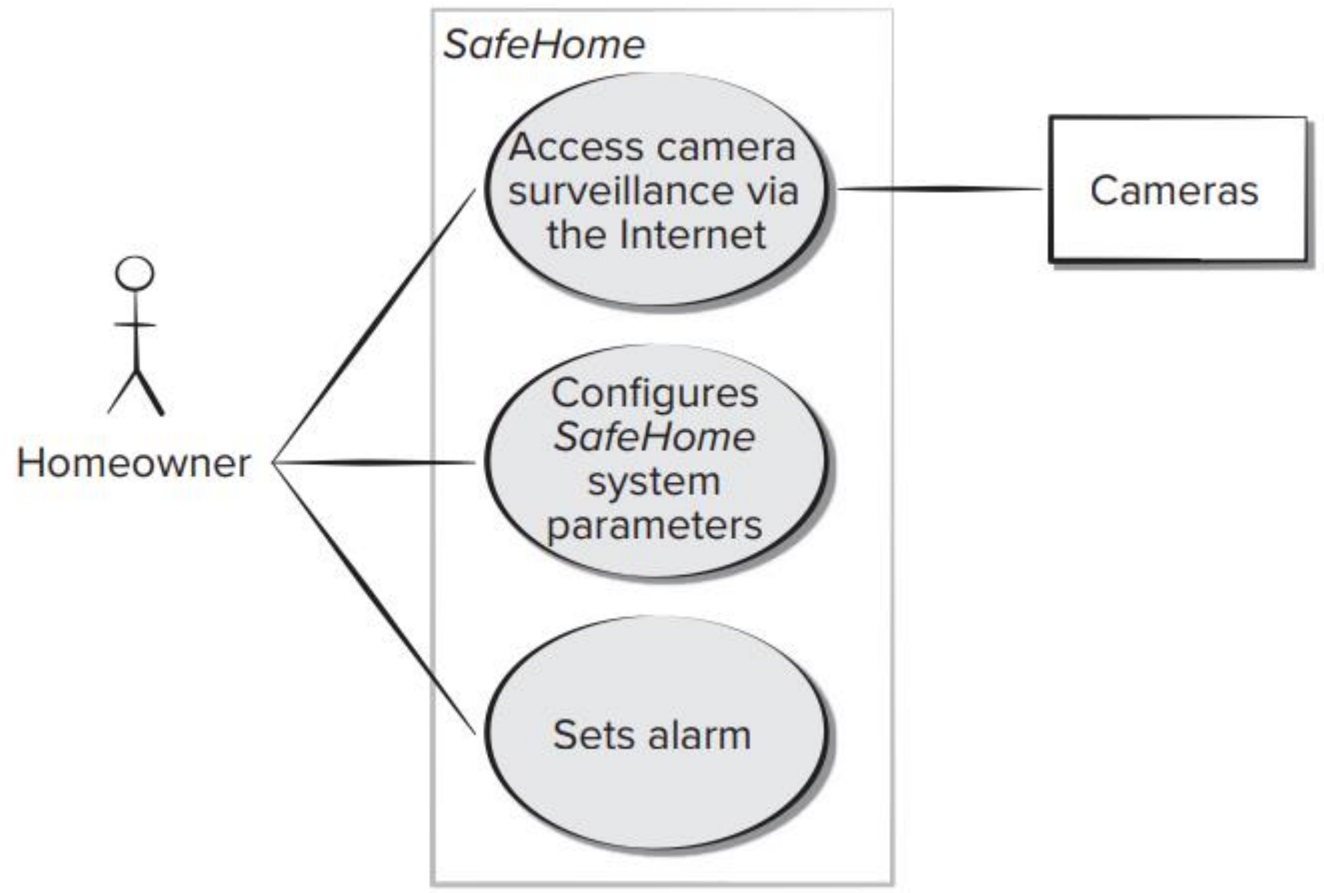
Use Case Diagram

- Use cases were developed originally to support requirements elicitation and now incorporated into the UML.
- Each use case represents a discrete task that involves external interaction with a system.
- Actors in a use case may be people or other systems. Represented diagrammatically to provide an overview of the use case and in a more detailed textual form.
- Components
  - A large box: *system boundary*
  - Stick figures outside the box: *actors*, both human and systems
  - Each oval inside the box: a use case that represents some major required functionality and its variant
  - A line between an actor and use case: the actor participates in the use case

Use Case Diagram

FIGURE 8.2

Preliminary use case diagram for the *SafeHome* system


SafeHome
Access camera surveillance via the Internet
Configures SafeHome system parameters
Sets alarm
Homeowner
Cameras

Class Based Modeling

Class Diagram

- Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.
- An association is a link between classes that indicates that there is some relationship between these classes.
- When you are developing models during the early stages of the software engineering process, objects represent something in the real world, such as a patient, a prescription, doctor, etc.
- A implementation is developed you usually need to define additional implementation objects
- Today we will focus on modeling real world objects as part of the requirement or early stage software design

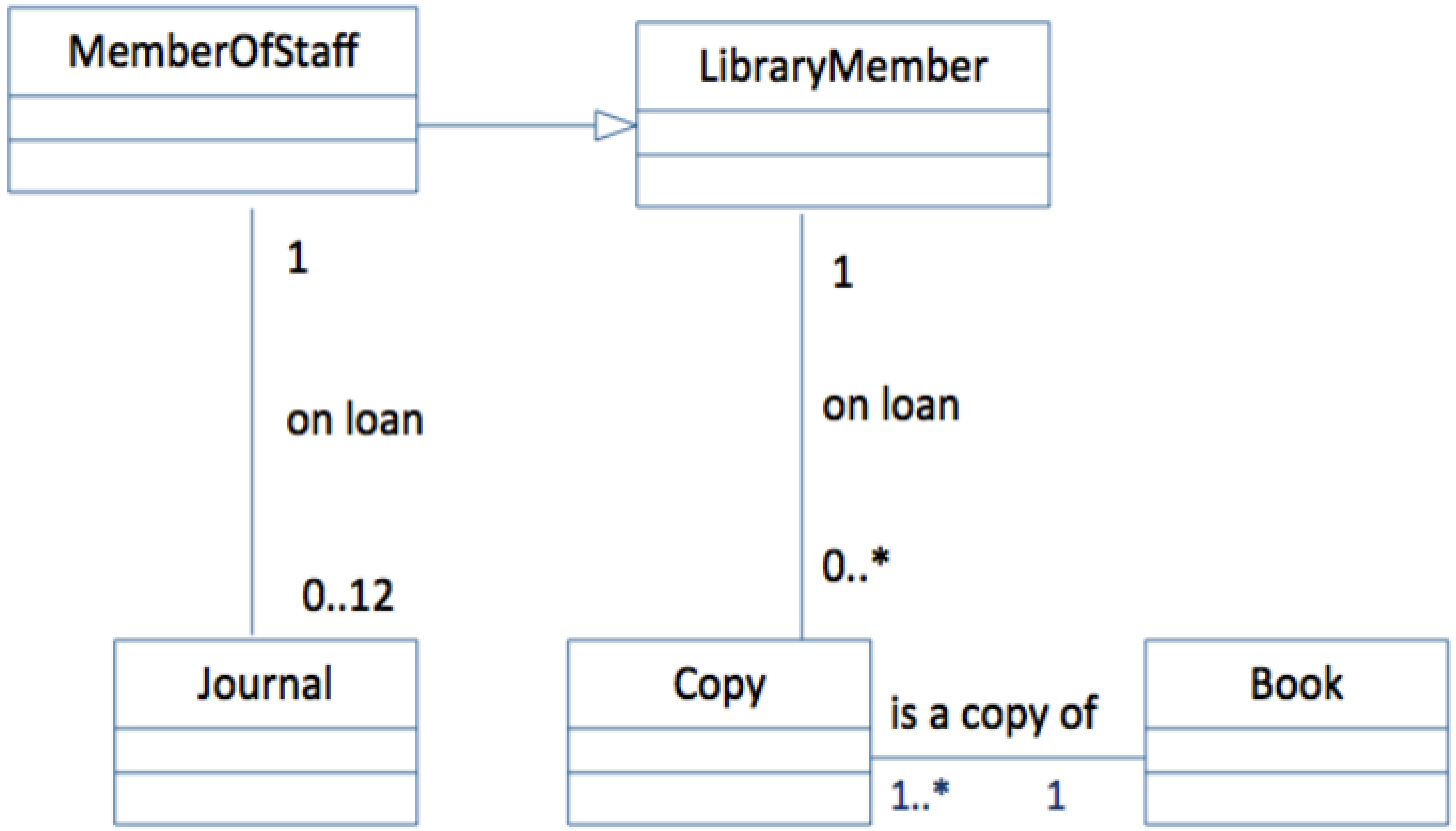
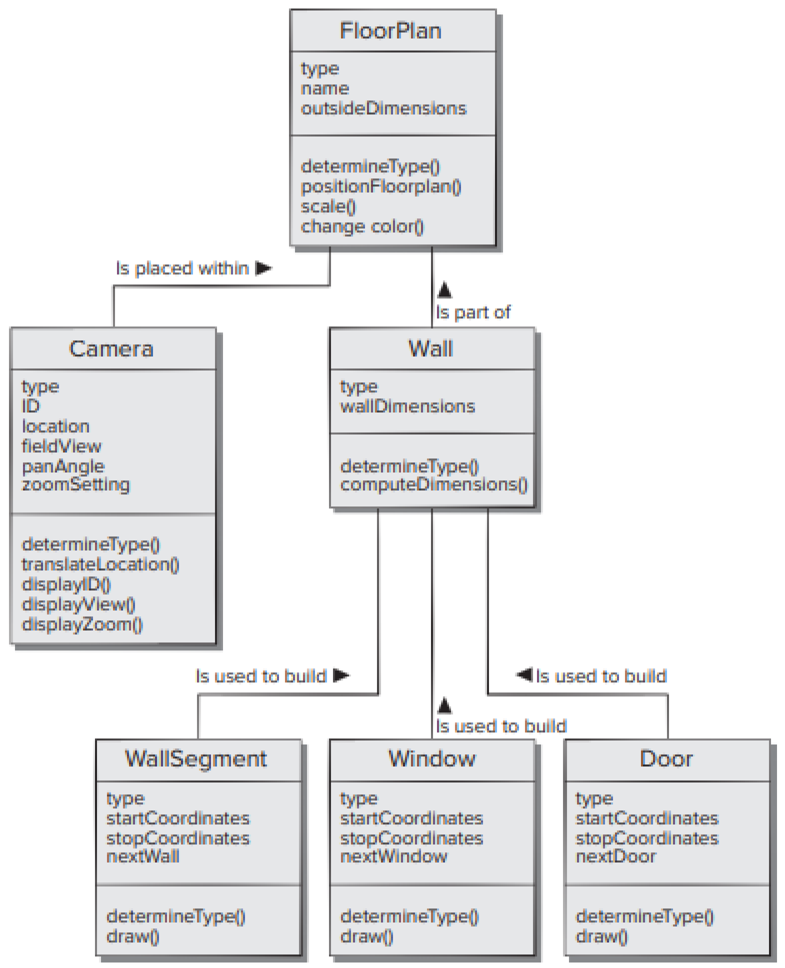

MemberOfStaff
LibraryMember
1
on loan
0..12
1
on loan
0..*
Journal
Copy
is a copy of
1..*
1
Book

FIGURE 8.4

Class diagram for FloorPlan (see sidebar discussion)

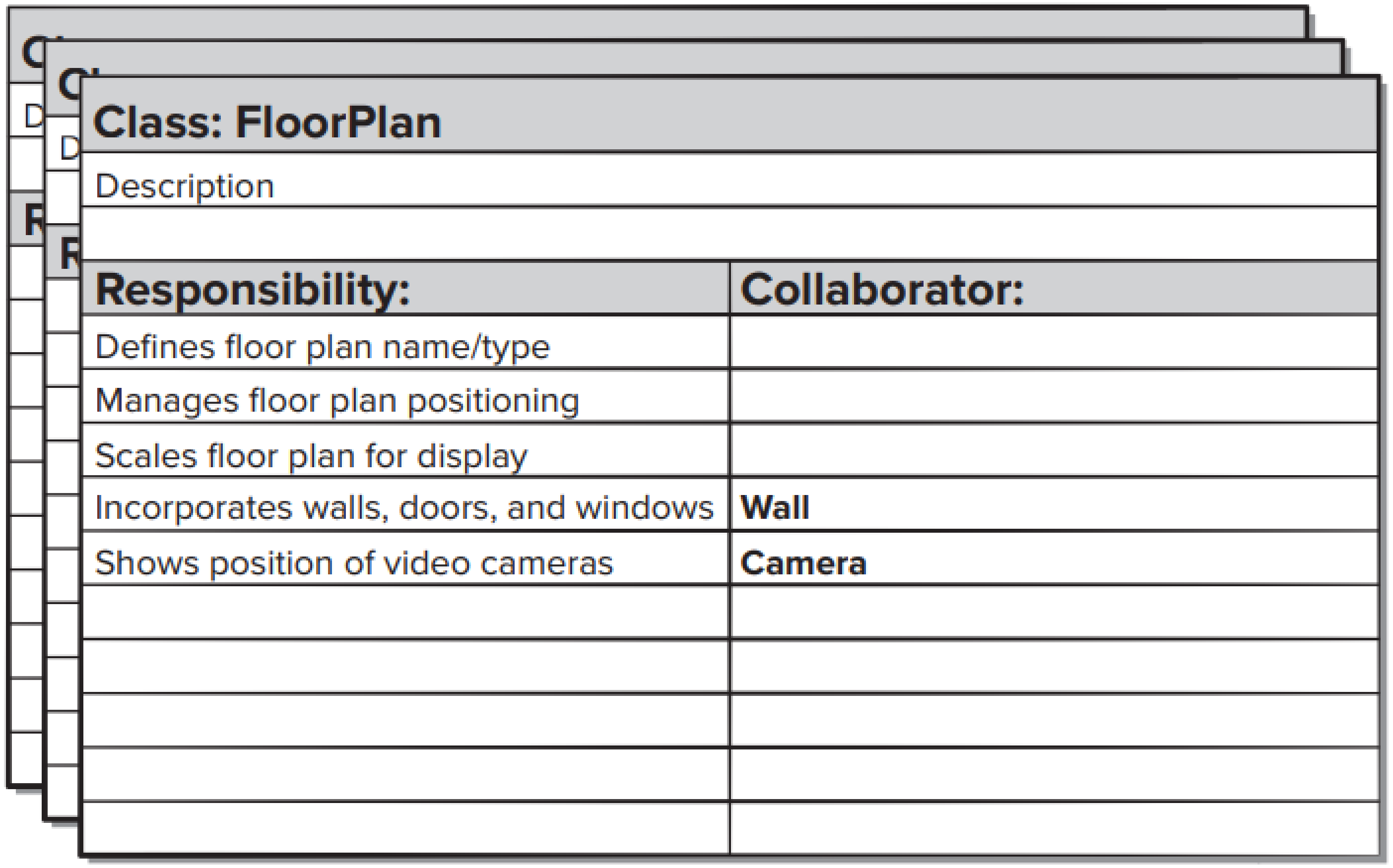


Class-Responsibility-Collaborator Modeling

- For identifying and organizing the classes that are relevant to system or product requirements
- a collaborator is another class (or object) that a specific class interacts with to fulfill its responsibilities.

FIGURE 8.5

A CRC model index card



Class: FloorPlan

Description

Responsibility:	**Collaborator:**
Defines floor plan name/type	
Manages floor plan positioning	
Scales floor plan for display	
Incorporates walls, doors, and windows	**Wall**
Shows position of video cameras	**Camera**

Behavioral Modeling

Behavioral Model

- To create the model, you should perform the following steps:

(1) evaluate all use cases to fully understand the sequence of interaction within the system

(2) identify events that drive the interaction sequence and understand how these events relate to specific objects

(3) create a sequence for each use case

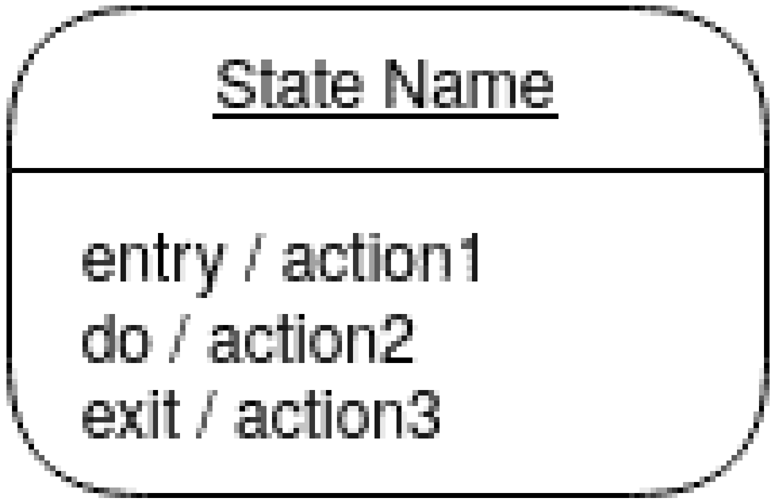
(4) build a state diagram for the system

(5) review the behavioral model to verify accuracy and consistency

State Diagram

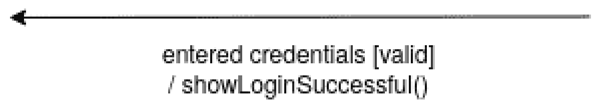
- Represents active states for each class and the events (triggers) that cause changes between these active states
- Each arrow shown in Figure 8.8 represents a transition from one active state of an object to another.
- The labels shown for each arrow represent the event that triggers the transition.
- Although the active state model provides useful insight into the “life history” of an object, it is possible to specify additional information to provide more depth in understanding the behavior of an object.

State Diagram



1. Transition Actions

- They are functions that are invoked after a transition is finished:

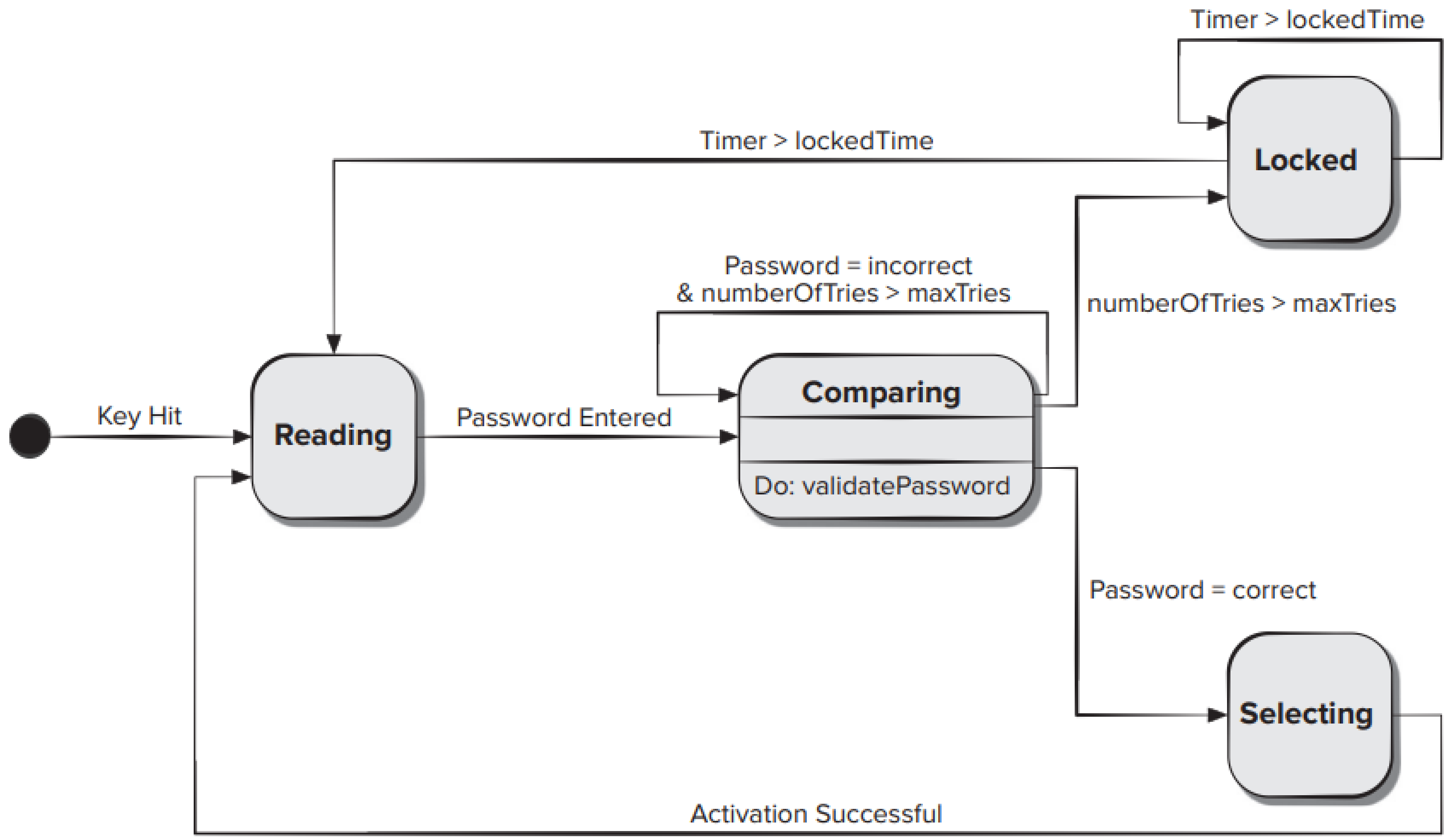


2. State Actions

- State actions are invoked when the system is in a specific state. There are 3 types of state actions:

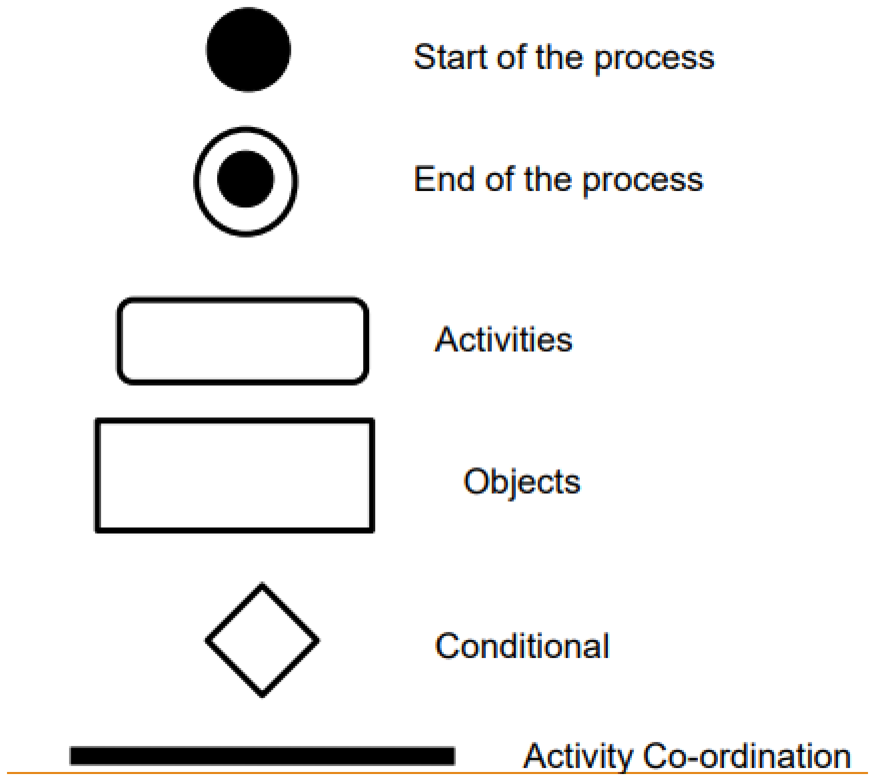
- Entry actions: they are executed when the system enters the state
- Do actions: they are continuously executed while the system is in the state
- Exit actions: they are executed when the system exits the state

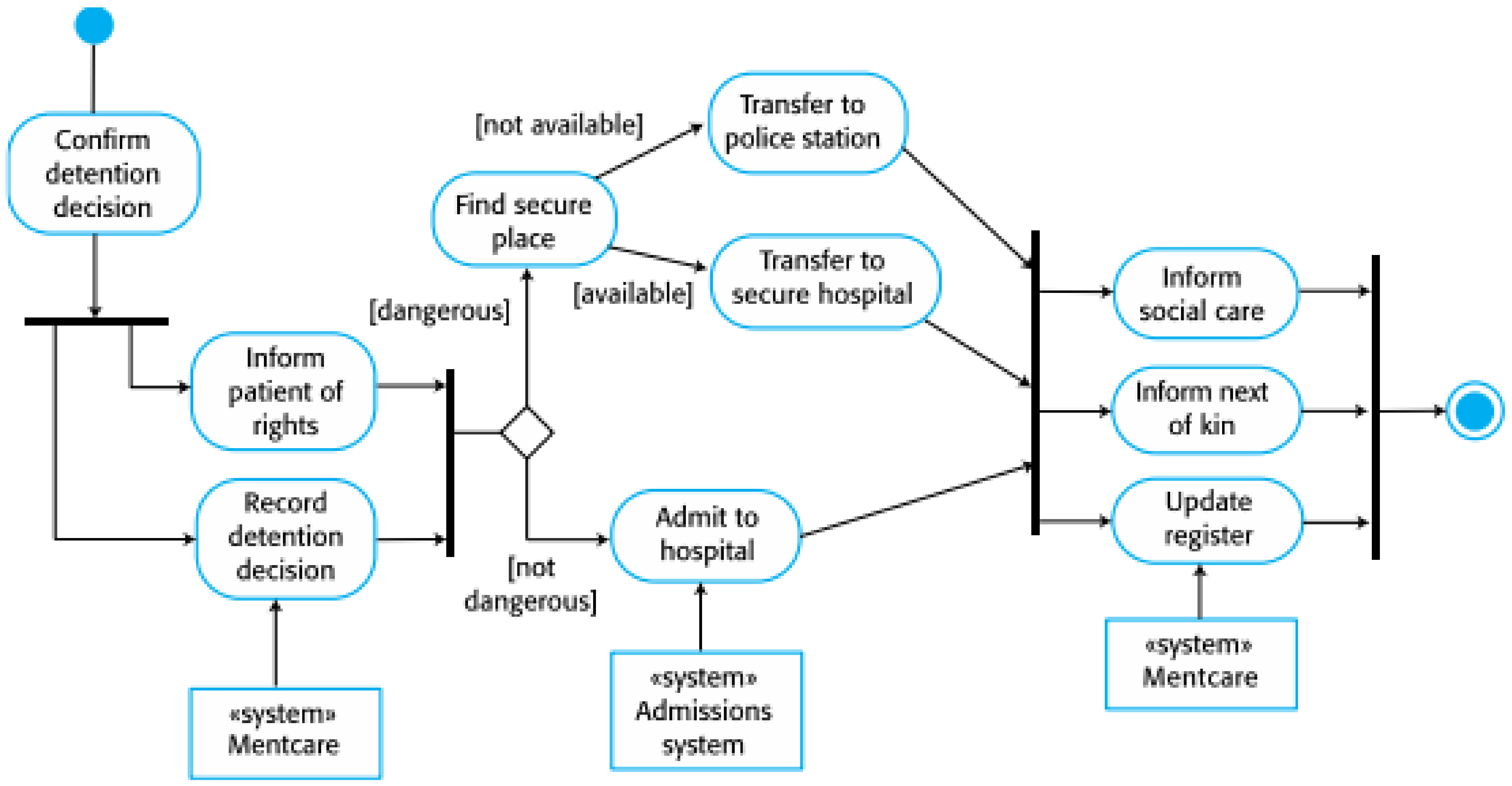
FIGURE 8.8 **State diagram for the ControlPanel class**



Activity Diagram

- Depicts behavior of the system
  - Dynamic aspects
  - Control flow from start to finish

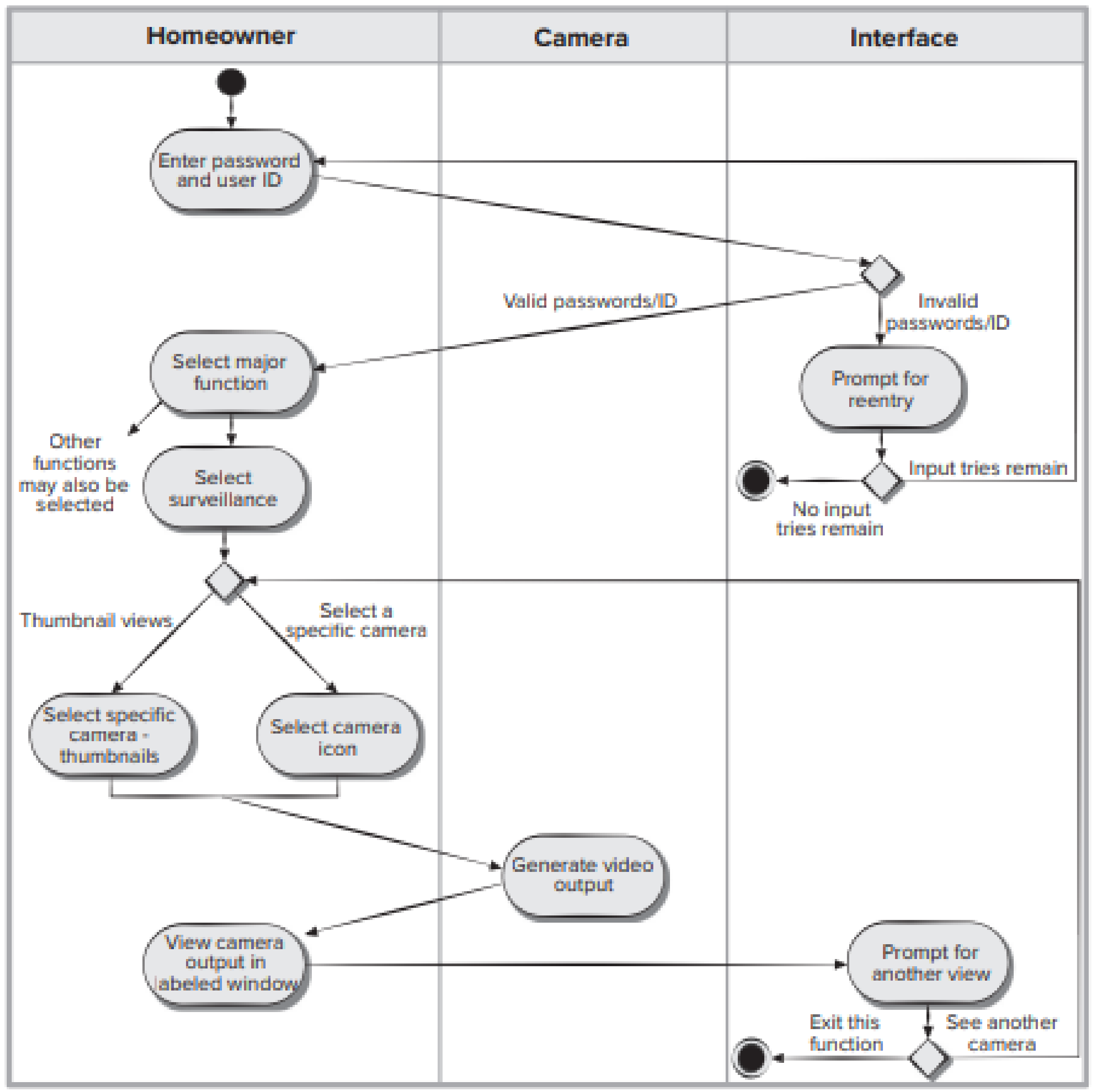




Confirm detention decision
Inform patient of rights
Record detention decision
«system» Mentcare
[dangerous]
Find secure place
[not available]
Transfer to police station
[available]
Transfer to secure hospital
[not dangerous]
Admit to hospital
«system» Admissions system
Inform social care
Inform next of kin
Update register
«system» Mentcare

Swimlane Diagram

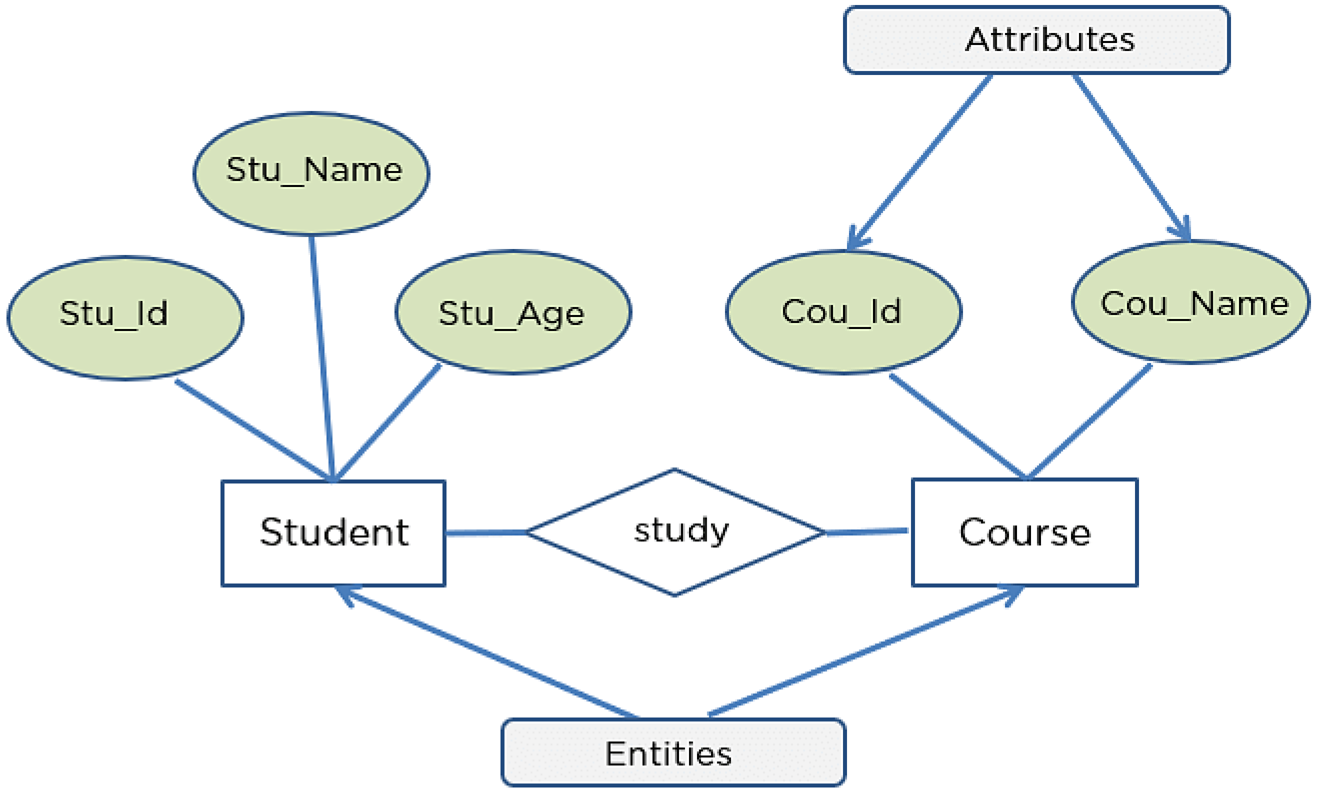
FIGURE 8.10 Swimlane diagram for Access camera surveillance via the Internet—display camera views function



Data Models

Entity-Relationship (ER) Diagram

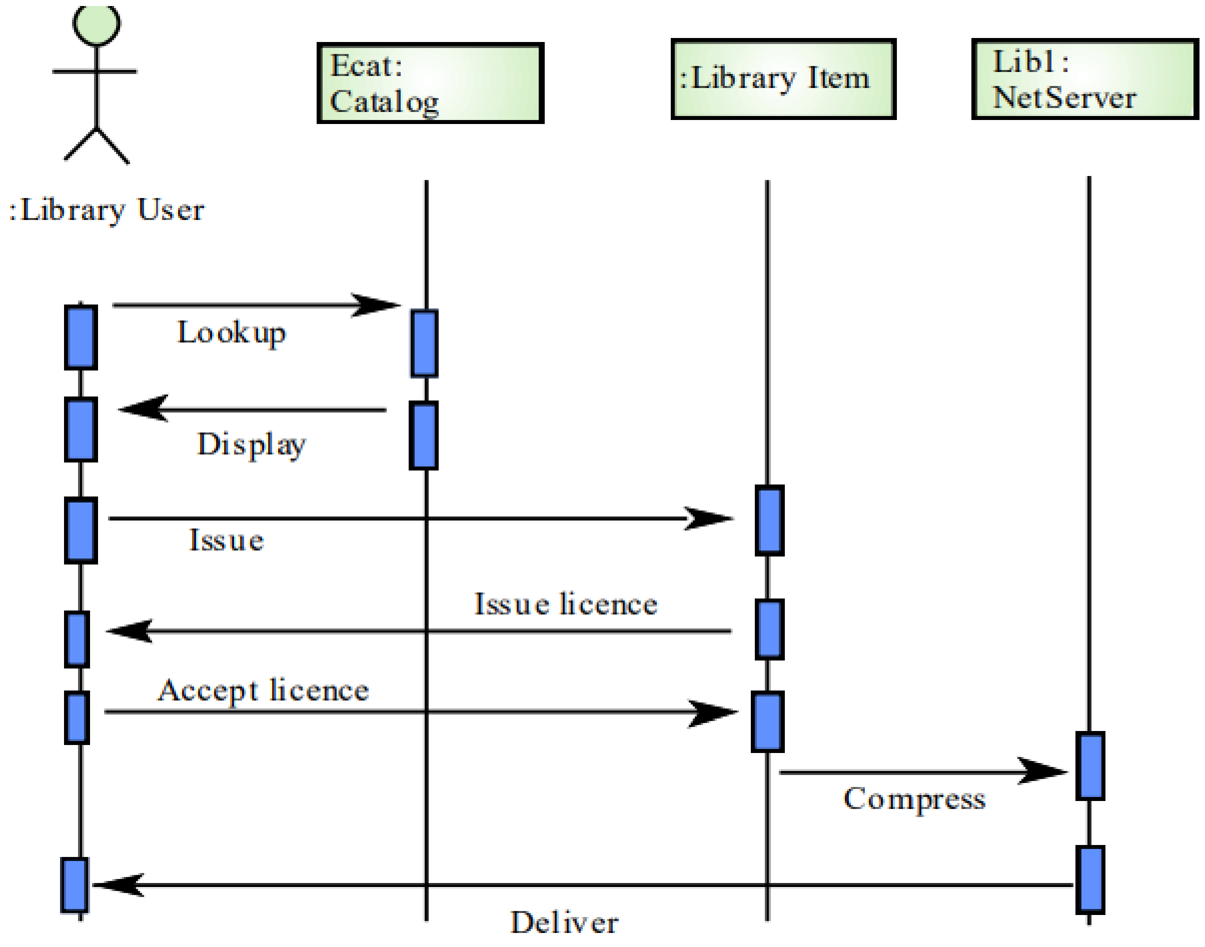
- A popular graphical notational paradigm for representing conceptual models
- Has three core constructs
  - An *entity*: depicted as a rectangle, represents a collection of real-world objects that have common properties
  - A *relationship*: depicted as an edge between two entities, with diamond in the middle of the edge specifying the type of relationship
  - An *attribute*: an annotation on an entity that describes data or properties associated with the entity


Attributes
Stu_Name
Stu_Id
Stu_Age
Cou_Id
Cou_Name
Student
study
Course
Entities

Functional Modeling

Sequence Diagram

- A graphical description of a sequence of events that are exchanged between real-world entities
  - *Vertical line*: the timeline of distinct entity, whose name appear at the top of the line
  - *Horizontal line*: an event or interaction between the two entities bounding the line
  - Time progresses from top to bottom
- Each graph depicts a single trace, representing one of several possible behaviours
- Traces have a semantic that is relatively precise, simple and easy to understand

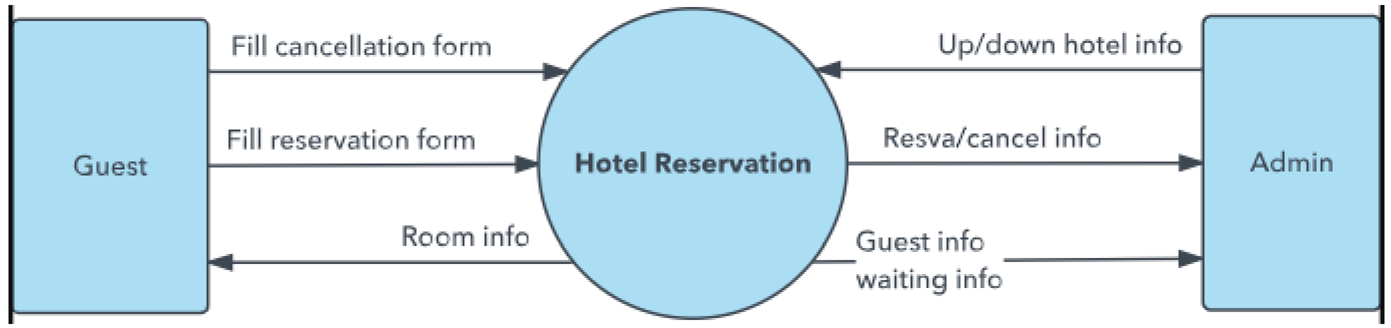

:Library User
Ecat: Catalog
:Library Item
Lib1: NetServer
Lookup
Display
Issue
Issue licence
Accept licence
Compress
Deliver

Data Flow Diagram

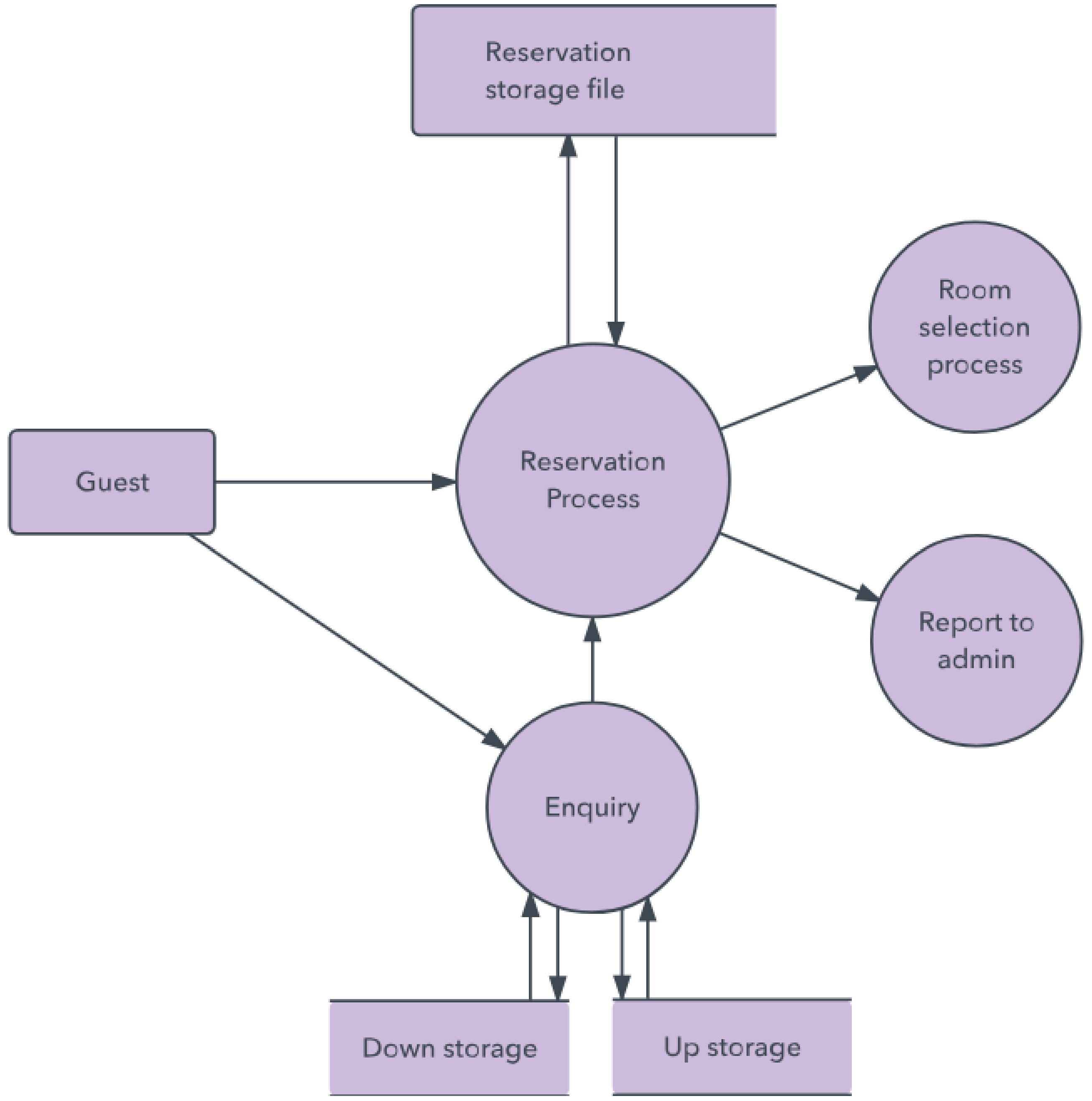
DFD rules and tips

- DFDs are valuable for understanding the high-level functionality and interactions within a system.
- Each process should have at least one input and an output.
- Each data store should have at least one data flow in and one data flow out.
- Data stored in a system must go through a process.
- All processes in a DFD go to another process or a data store.

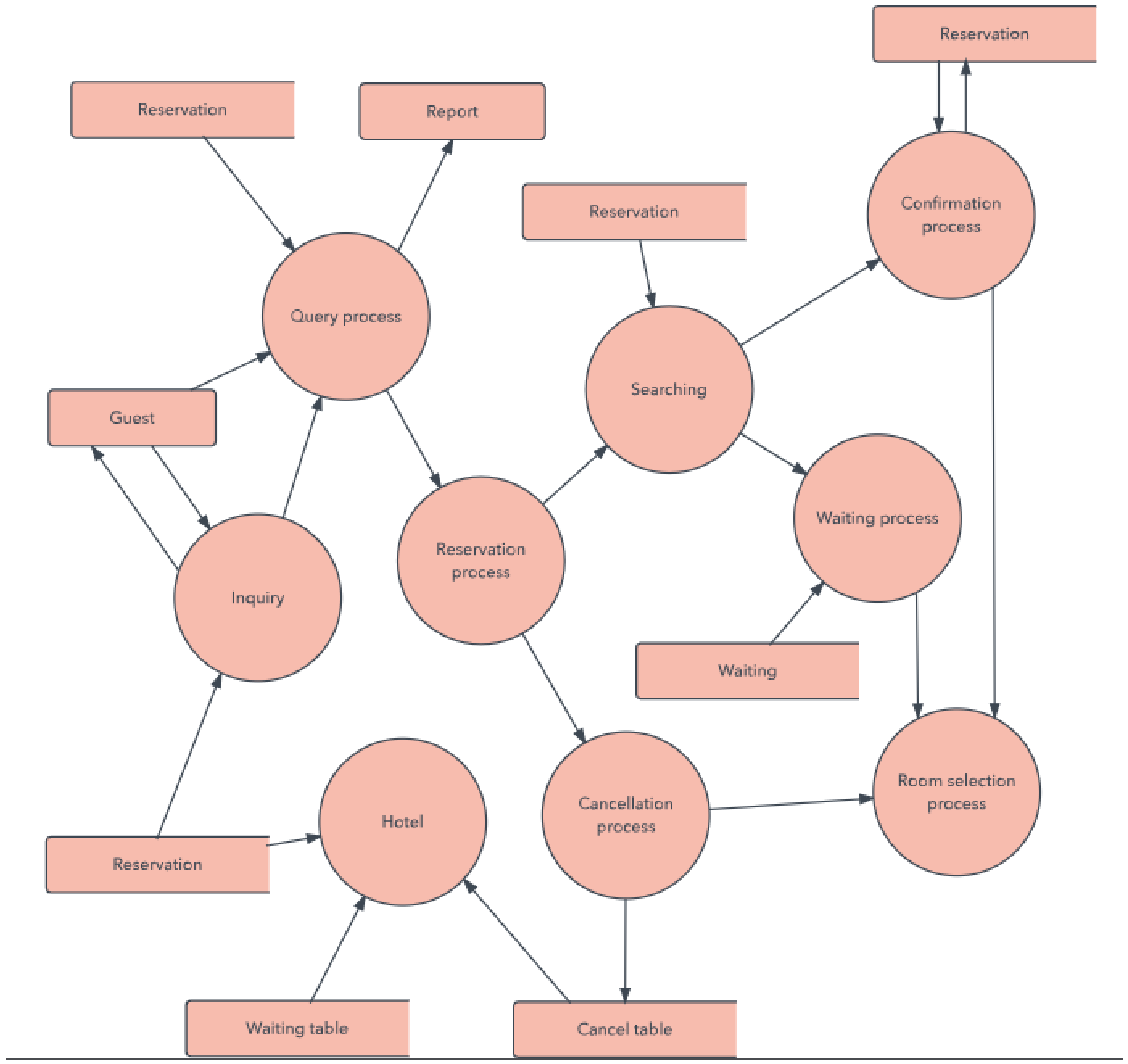
DFD-Level 0


Guest
Fill cancellation form
Fill reservation form
Room info
Hotel Reservation
Up/down hotel info
Resva/cancel info
Guest info
waiting info
Admin

DFD-Level 1


Reservation storage file
Room selection process
Guest
Reservation Process
Report to admin
Enquiry
Down storage
Up storage

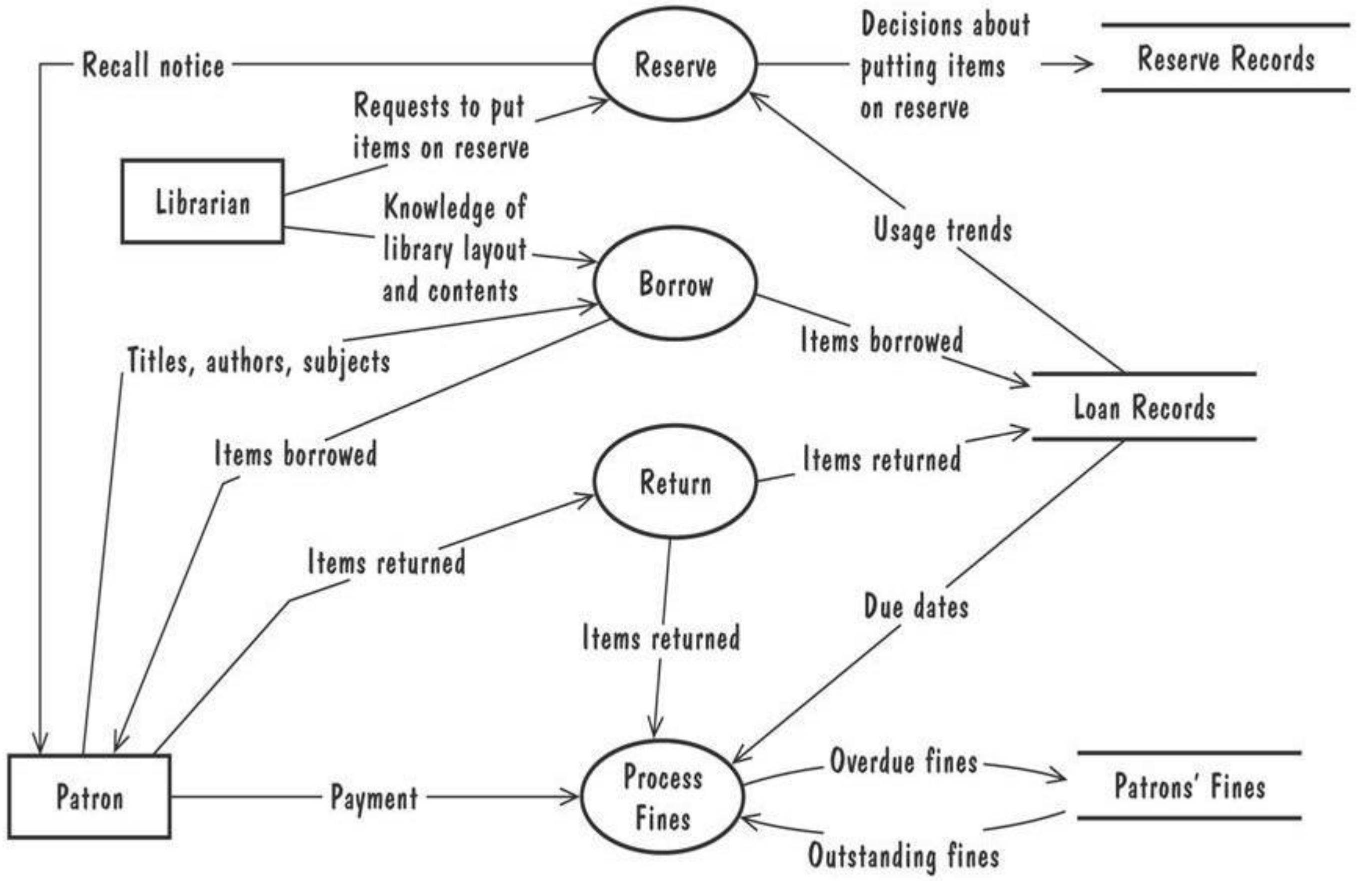
DFD-Level 2


Reservation
Report
Reservation
Reservation
Confirmation process
Query process
Searching
Guest
Reservation process
Waiting process
Inquiry
Waiting
Room selection process
Hotel
Cancellation process
Reservation
Waiting table
Cancel table

DFD

- Consider a library management system where circulation services allow the Patron to borrow or return a book/item/publication without involvement of Librarian, so the system needs to process these requests. The Librarian controls which items can be borrowed and how can the items be shown to the patrons. The system processes fines also when a borrowed item is returned after the due date. The librarian is able to reserve an item if required and a recall notice is sent to the patron as a consequence. The record of reserved items is kept so that the librarian can view which items have been reserved already.

Level-1 DFD


Recall notice
Reserve
Decisions about putting items on reserve
Reserve Records
Requests to put items on reserve
Librarian
Knowledge of library layout and contents
Borrow
Usage trends
Titles, authors, subjects
Items borrowed
Loan Records
Items borrowed
Return
Items returned
Items returned
Items returned
Due dates
Patron
Payment
Process Fines
Overdue fines
Patrons' Fines
Outstanding fines

References

- SE Pressman
- What is a Data Flow Diagram | Lucidchart