Software Engineering

Chapter 6 & 7: Requirements Modeling

A process is a roadmap for building high-quality software.

Contents

- Requirements Modeling
- Requirements Analysis
- Types of Modeling
- Scenario Based Modeling
- Class based Modeling
- Data Modeling
- Flow Oriented Modeling
- Behavioral Modeling

Requirements Modeling

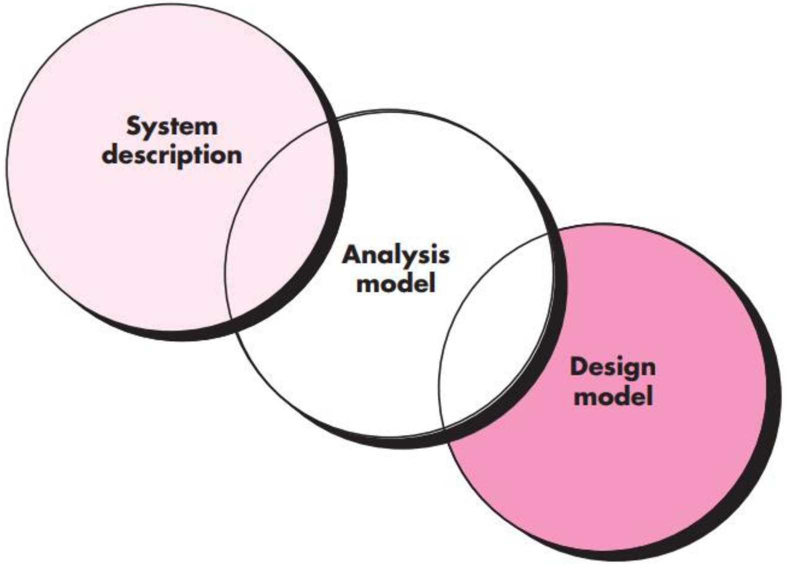
- ***Requirements Modelling (RM)*** is the process of **representing and visualizing** what the system should do, before you actually start designing or coding it.
- It is a ***set of models*** that is the ***first technical representation*** of a system.
- When we model requirements, we try to **understand the system from different perspectives:**
  - What users need?
  - How data will flow?
  - What processes will occur?
  - How the system will behave under different conditions?

Requirements Analysis

- The requirements modeling action results in one or more of the following ***types of models***:
  - ***Scenario based models***: Represents requirements from the point of view of various system actors.
  - ***Data Models:*** It represents the information domain for the problem.
  - ***Class oriented models***: It represents object-oriented classes (attributes and operations) and the manner in which classes collaborate to achieve system requirements.
  - ***Flow-oriented models:*** It represents the functional elements of the system and how they transform data as it moves through the system.
  - ***Behavioral models***: It depicts how the software behaves as a consequence of external "events"

Requirement Analysis (Cont..)

- These models provide a software designer with information that can be ***translated to architectural, interface, and component-level designs.***
- The ***requirements model*** as a bridge between the ***system description*** and the ***design model***
- .



Objectives of Requirements Modeling

- Throughout requirements modeling, your primary focus is on what, not how.
- The requirements model must achieve ***three*** primary objectives:

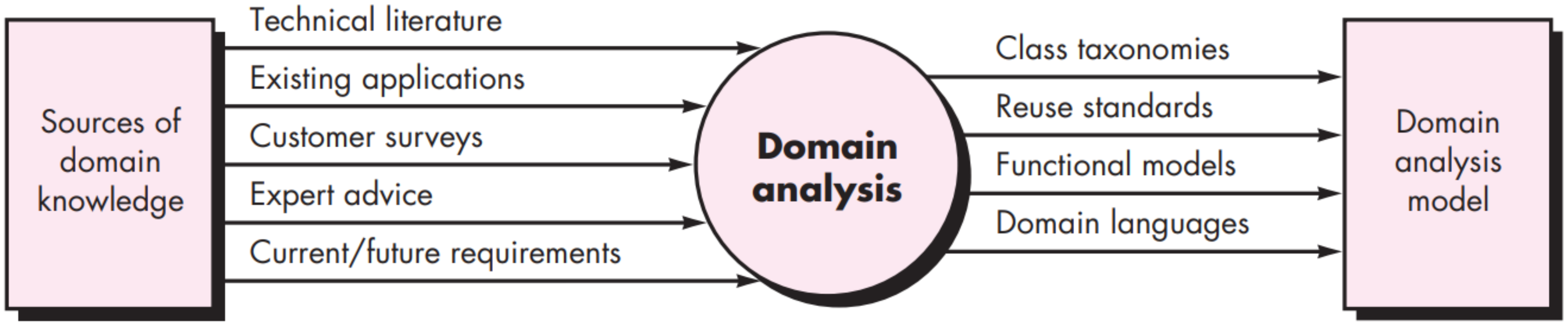
  (1) to describe what the ***customer requires***,

  (2) to establish a ***basis*** for the creation of a ***software design***, and

  (3) to define a set of requirements that can be ***validated*** once the software is built.

Domain Analysis

FIGURE 6.2 **Input and output for domain analysis**



Sources of domain knowledge

Technical literature

Existing applications

Customer surveys

Expert advice

Current/future requirements

Domain analysis

Class taxonomies

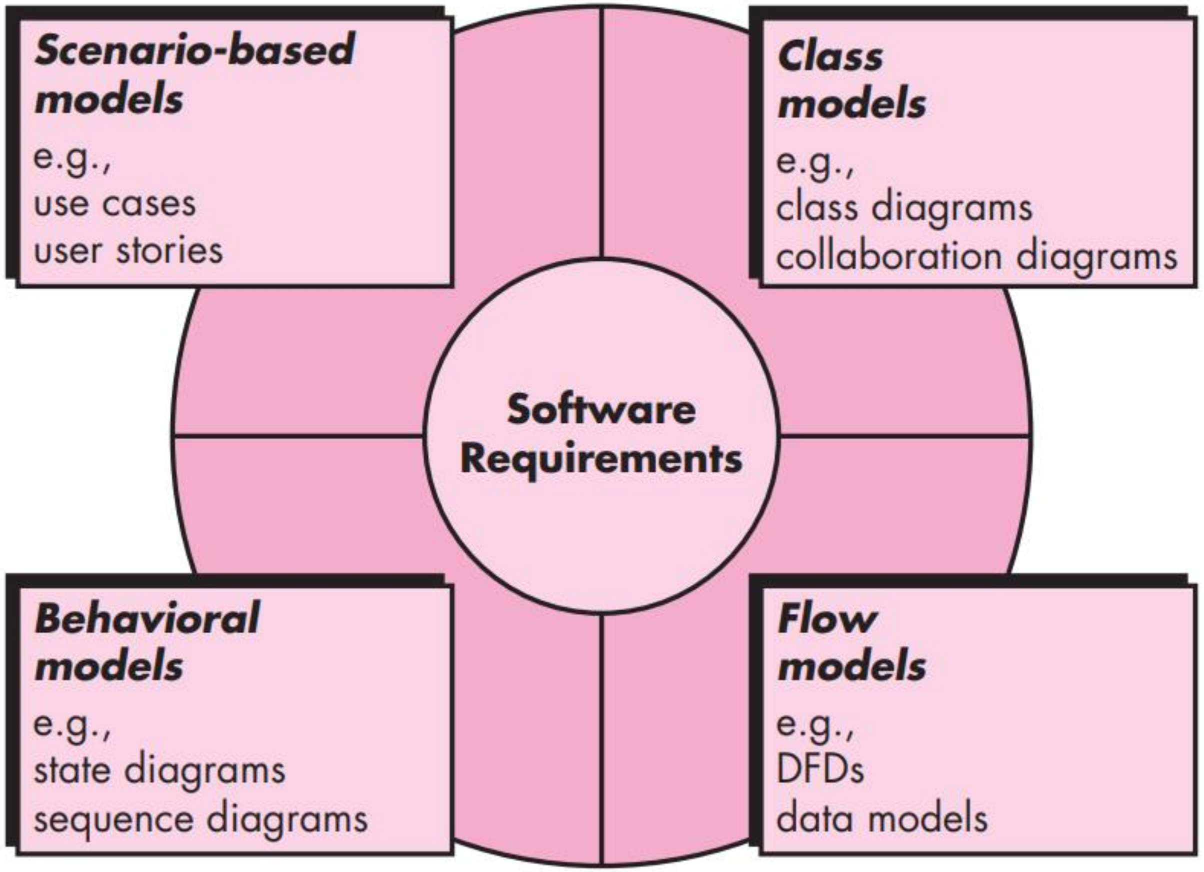
Reuse standards

Functional models

Domain languages

Domain analysis model

Requirements Modeling Approaches



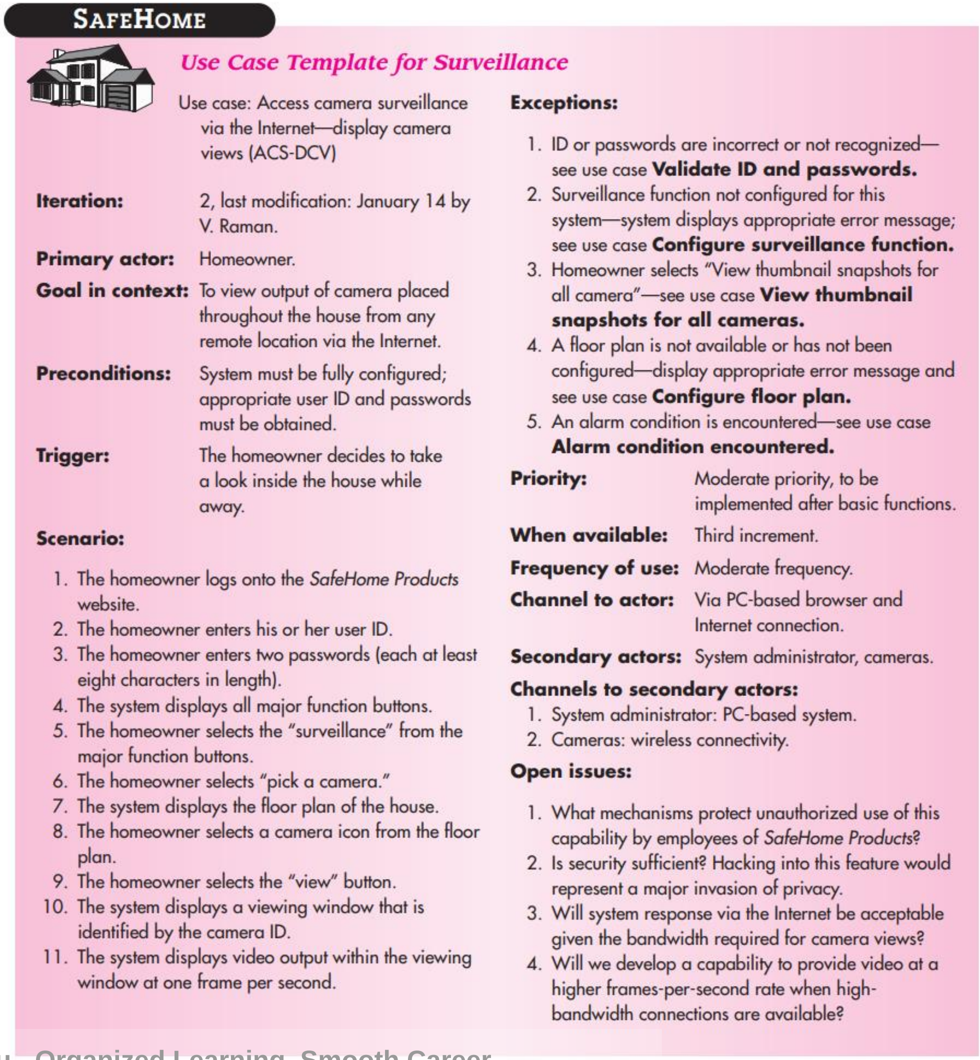
1. Scenario Based Modeling

- Scenario-based modeling is a requirements engineering technique to describe **how users use the system**, step by step, in real-world situations.
- Each scenario tells a story of how the system behaves from the user’s perspective.
- It answers:
  - Who uses the system?
  - What do they want to achieve?
  - How does the system respond to each action?
- It’s basically “storytelling for software requirements.”

1. Scenario Based Modeling

- Creating a Preliminary Use Case
- Refining a Preliminary Use Case
- ***Writing a formal Use Case***
- ***Use Case Diagram***
- UML Models for Representing Use Case (Supplement of the Use Case)
  - ***Activity Diagram***
  - ***Swimlane Diagram***

Use Case Template



SAFEHOME

Use Case Template for Surveillance

Use case: Access camera surveillance via the Internet—display camera views (ACS-DCV)

Iteration: 2, last modification: January 14 by V. Raman.

Primary actor: Homeowner.

Goal in context: To view output of camera placed throughout the house from any remote location via the Internet.

Preconditions: System must be fully configured; appropriate user ID and passwords must be obtained.

Trigger: The homeowner decides to take a look inside the house while away.

Scenario:

1. The homeowner logs onto the *SafeHome Products* website.
2. The homeowner enters his or her user ID.
3. The homeowner enters two passwords (each at least eight characters in length).
4. The system displays all major function buttons.
5. The homeowner selects the "surveillance" from the major function buttons.
6. The homeowner selects "pick a camera."
7. The system displays the floor plan of the house.
8. The homeowner selects a camera icon from the floor plan.
9. The homeowner selects the "view" button.
10. The system displays a viewing window that is identified by the camera ID.
11. The system displays video output within the viewing window at one frame per second.

Exceptions:

1. ID or passwords are incorrect or not recognized—see use case **Validate ID and passwords.**
2. Surveillance function not configured for this system—system displays appropriate error message; see use case **Configure surveillance function.**
3. Homeowner selects "View thumbnail snapshots for all camera"—see use case **View thumbnail snapshots for all cameras.**
4. A floor plan is not available or has not been configured—display appropriate error message and see use case **Configure floor plan.**
5. An alarm condition is encountered—see use case **Alarm condition encountered.**

Priority: Moderate priority, to be implemented after basic functions.

When available: Third increment.

Frequency of use: Moderate frequency.

Channel to actor: Via PC-based browser and Internet connection.

Secondary actors: System administrator, cameras.

Channels to secondary actors:

1. System administrator: PC-based system.
2. Cameras: wireless connectivity.

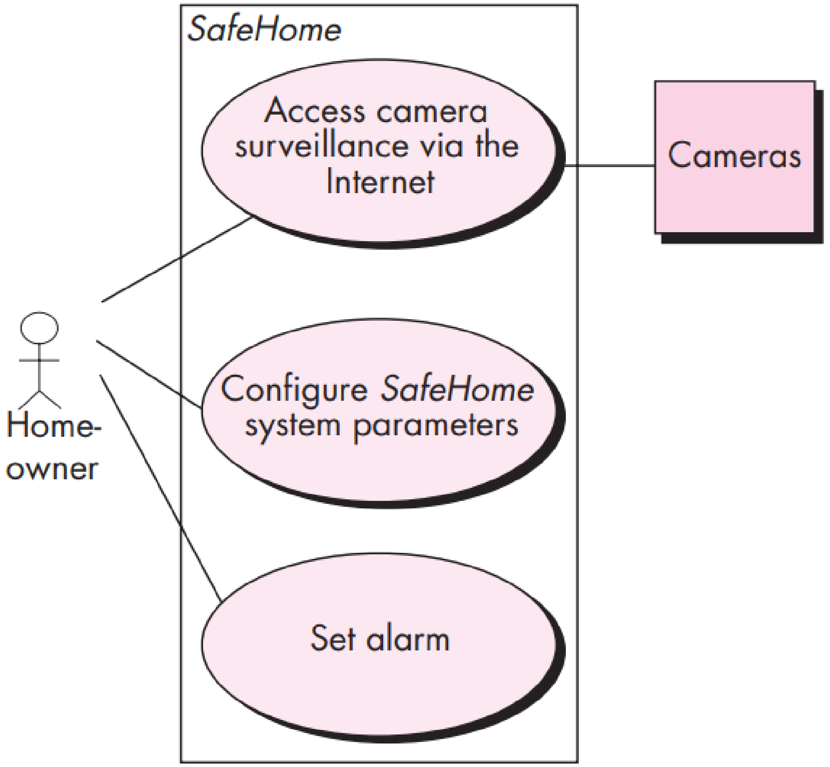
Open issues:

1. What mechanisms protect unauthorized use of this capability by employees of *SafeHome Products*?
2. Is security sufficient? Hacking into this feature would represent a major invasion of privacy.
3. Will system response via the Internet be acceptable given the bandwidth required for camera views?
4. Will we develop a capability to provide video at a higher frames-per-second rate when high-bandwidth connections are available?

Use case Diagram

FIGURE 6.4

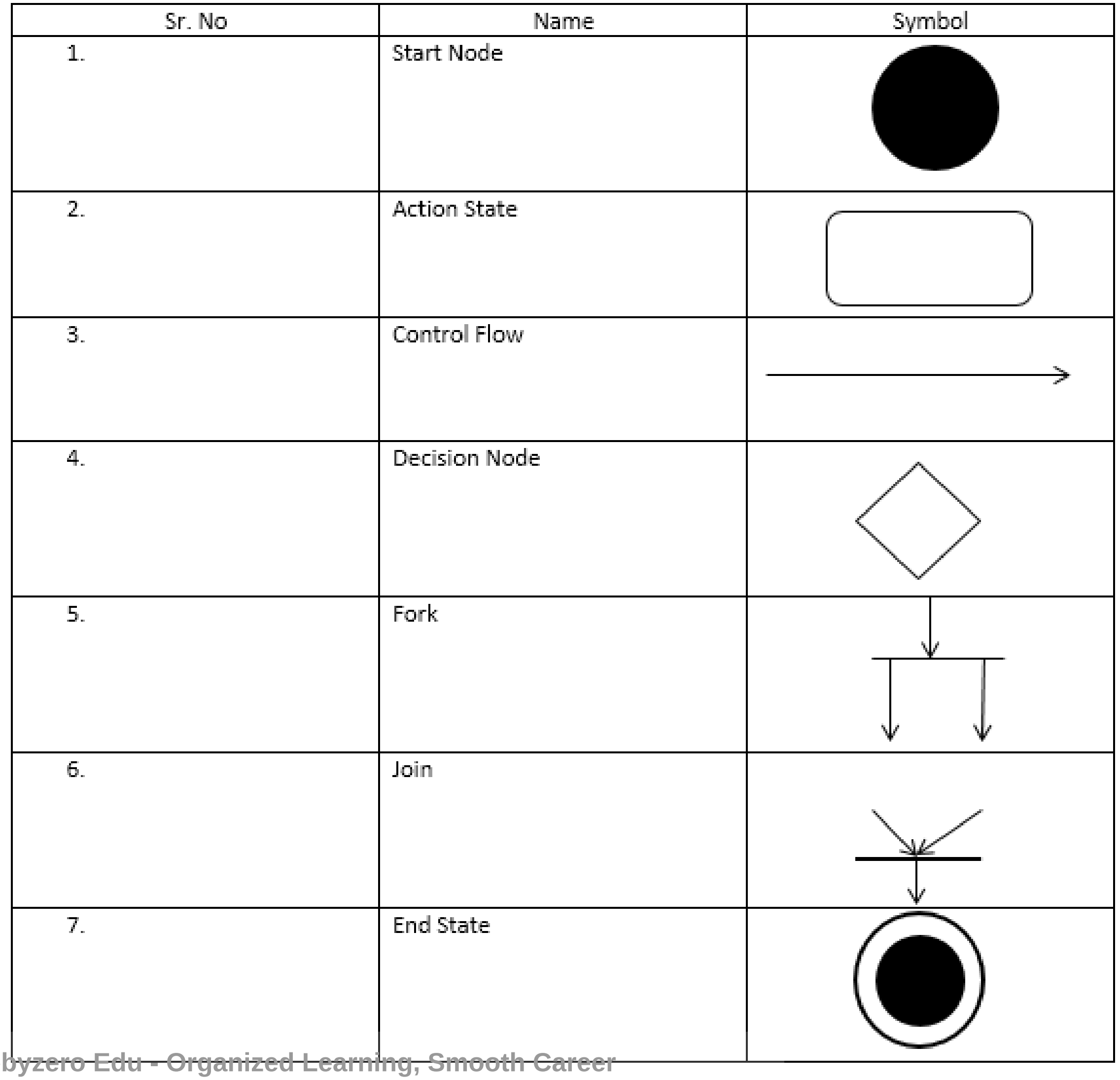
Preliminary use-case diagram for the *SafeHome* system


SafeHome
Access camera surveillance via the Internet
Configure *SafeHome* system parameters
Set alarm
Cameras
Home-owner

Activity Diagram

- An **activity diagram** is a UML diagram that represents the **workflow** or **step-by-step activities** of a system or business process.
- You can think of it as:
  - A visual script of how tasks are completed
  - A map of actions, decisions, and parallel activities
- *Purposes*
  - Understand the complete *sequence of operations*
  - *Visualize* complex workflows clearly
  - Communicate processes easily with both ***technical*** and ***non-technical*** people
  - Discover *missing steps* in requirements

Symbols for drawing Activity Diagram

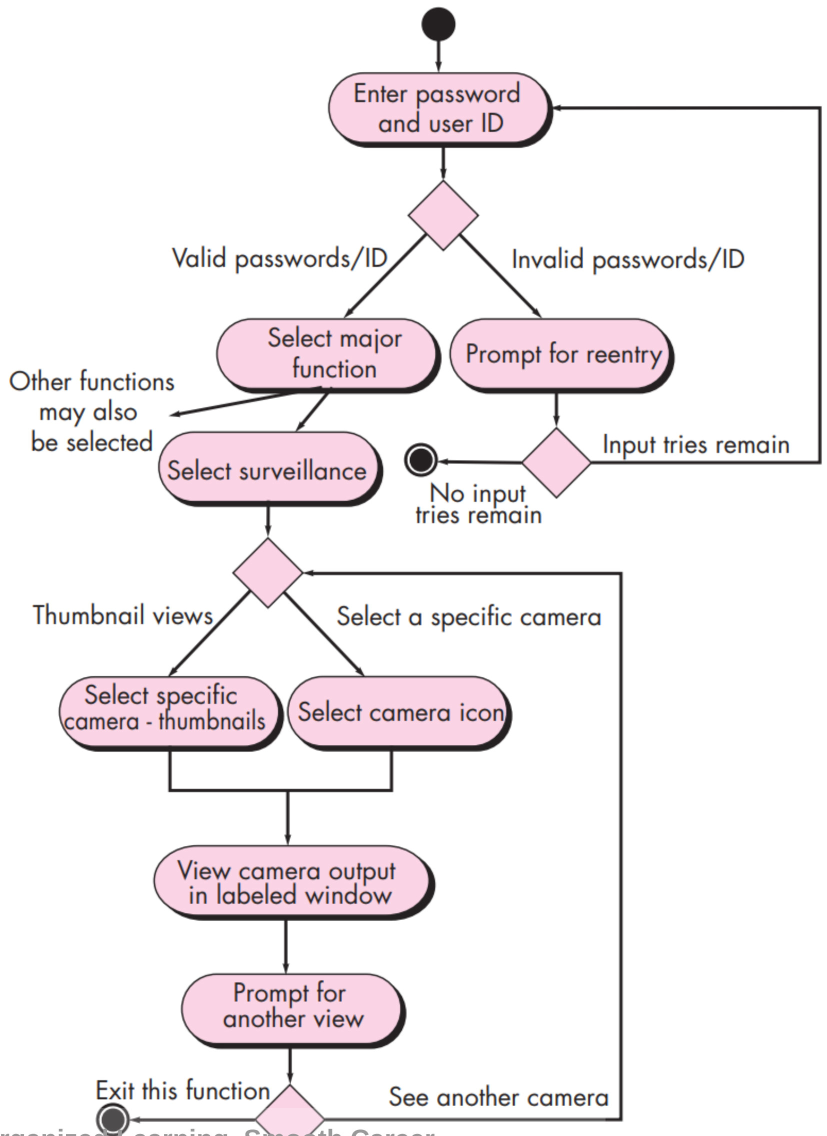


Sr. No	Name	Symbol
1.	Start Node	●
2.	Action State	▢
3.	Control Flow	→
4.	Decision Node	◇
5.	Fork	
6.	Join	
7.	End State	◉

Activity Diagram

FIGURE 6.5

Activity diagram for Access camera surveillance via the Internet—display camera views function.

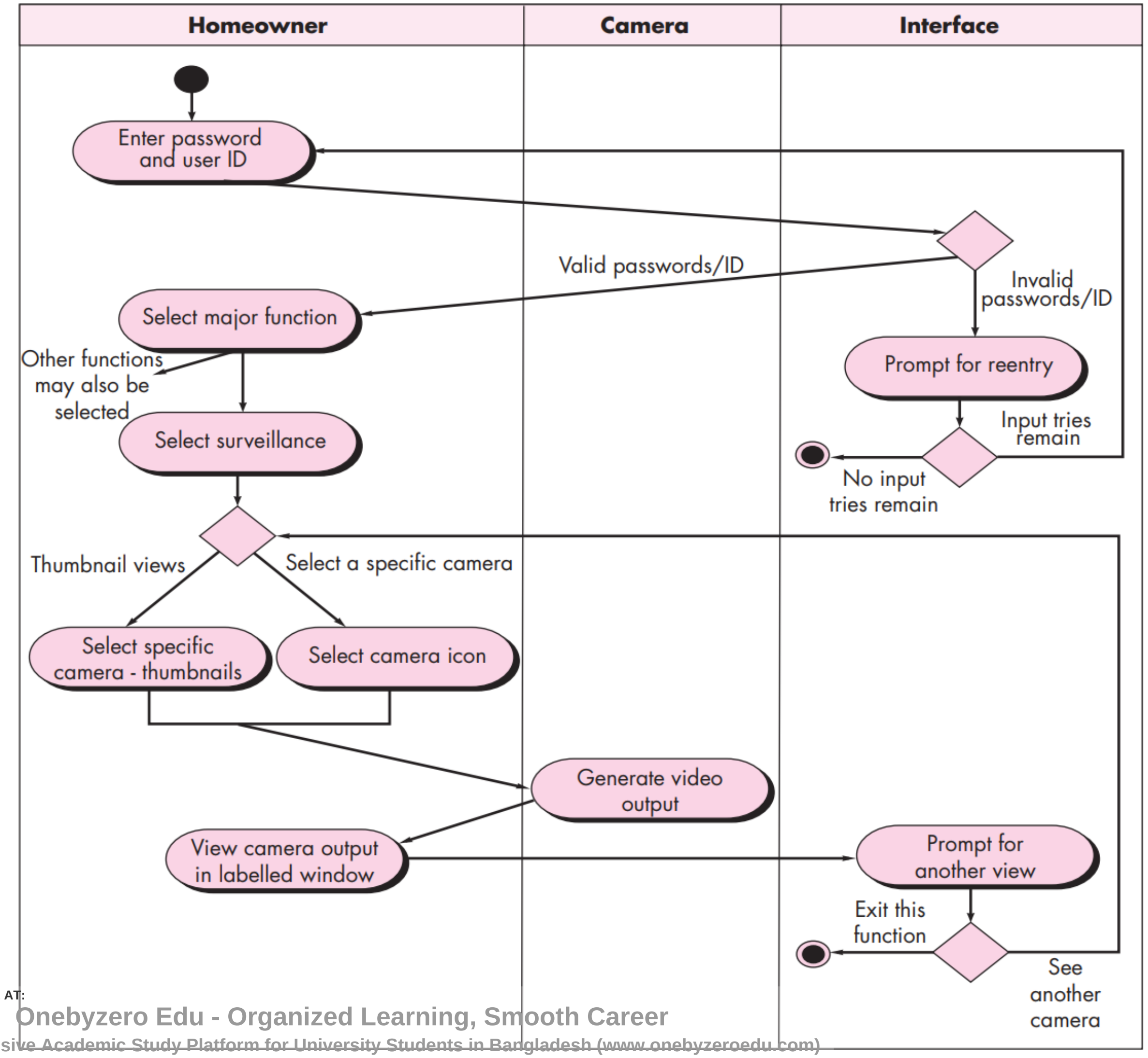


Swimlane Diagram

- A **swimlane diagram** is a special type of ***activity diagram*** (or flowchart) where the activities are arranged inside horizontal or vertical lanes.
- Each ***lane*** represents an ***actor or unit responsible for that part*** of the process.
- *It looks like a swimming pool with lanes, and each lane "belongs" to someone.*
- ***Lanes: Represent roles (Student, Librarian, System), departments, devices, etc.***

Swimlane Diagram

FIGURE 6.6 Swimlane diagram for Access camera surveillance via the Internet—display camera views function



2. Data Modeling

- ERD
  - Data Objects (Entity)
  - Data Attributes
  - Ralationships
- Schema

3. Class Oriented Modeling

- Class-based modeling represents:
  - ***objects*** that the system will manipulate
  - ***operations*** (also called methods or services) that will be applied to the objects to effect the manipulation
  - ***relationships*** (some hierarchical) between the objects
  - ***collaborations*** that occur between the classes that are defined.
- The elements of a class-based model include ***classes and objects, attributes, operations***, ***CRC models***, *collaboration diagrams and packages*.

Process of COM

- ***Identify Possible Classes***
- ***Select Potential Classes***
- ***Create Class Card***
- ***Create Class Diagram***
- CRC Model Index card
- CRC Diagram

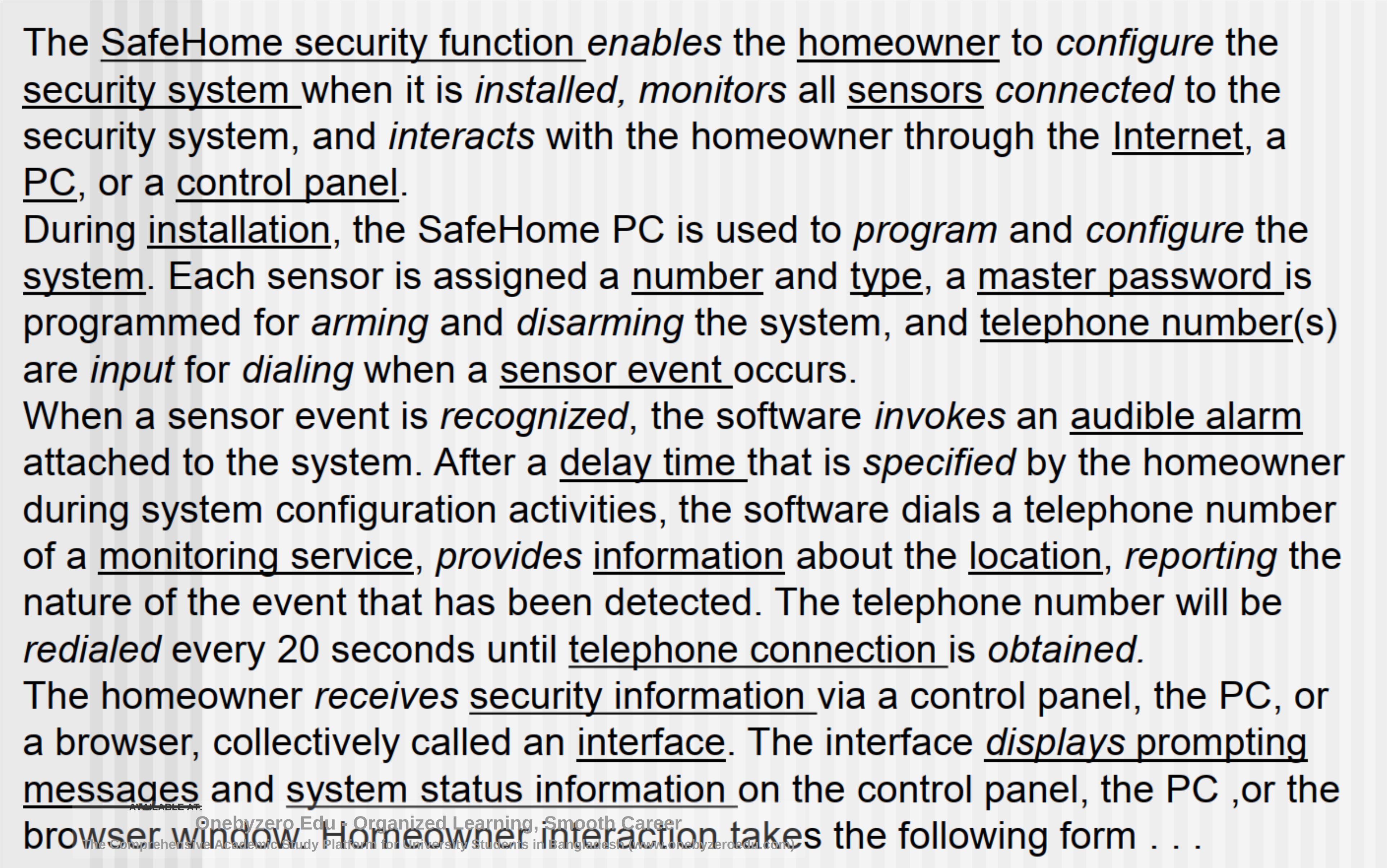
Identifying Analysis Classes

- ***External entities*** (e.g., other systems, devices, people) that produce or consume information
- ***Things*** (e.g, reports, displays, letters, signals) that are part of the information domain for the problem
- ***Occurrences or events*** (e.g., a property transfer or the completion of a series of robot movements) that occur within the context of system operation
- ***Roles*** (e.g., manager, engineer, salesperson) played by people who interact with the system
- ***Organizational units*** (e.g., division, group, team) that are relevant to an application
- ***Places*** (e.g., manufacturing floor or loading dock) that establish the context of the problem and the overall function
- ***Structures*** (e.g., sensors, four-wheeled vehicles, or computers) that define a class of objects or related classes of objects

Samples classes on Library Management System

Category	**Description**	**Example (Library System)**
External Entities	People or systems that interact with the system	Librarian, Member, Payment Gateway
Things	Information objects	Book Record, Borrowing Slip
Occurrences/Events	Activities or system triggers	Book Issued, Book Returned
Roles	Responsibilities or user types	Member, Librarian, Administrator
Organizational Units	Divisions or departments	Library Department, Accounts Section
Places	Physical locations	Issue Desk, Library Hall
Structures	Physical or logical objects	Barcode Scanner, Database Server

Performing Grammatical Parse (Identify classes/ Nouns)



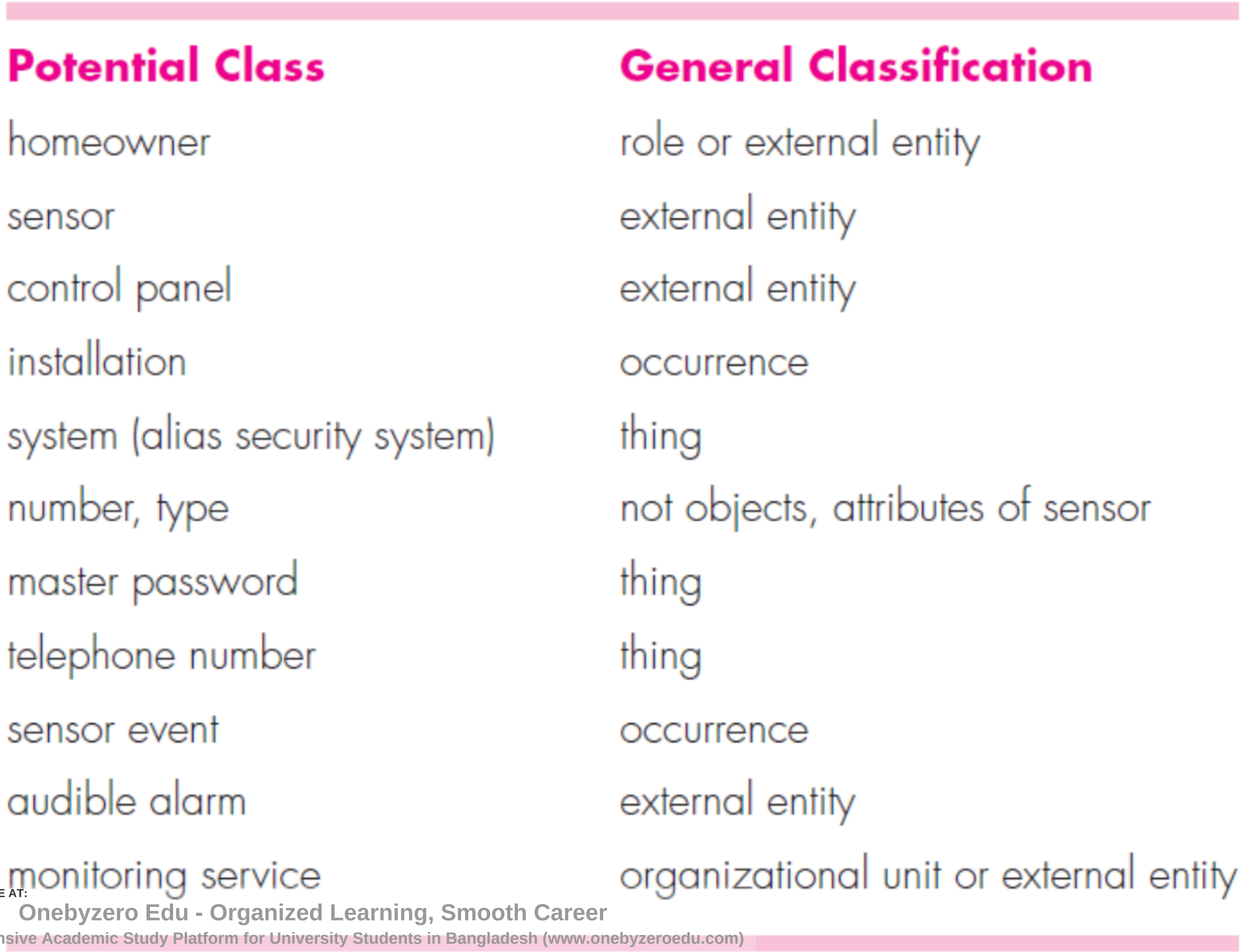
The SafeHome security function *enables* the homeowner to *configure* the security system when it is *installed, monitors* all sensors *connected* to the security system, and *interacts* with the homeowner through the Internet, a PC, or a control panel.

During installation, the SafeHome PC is used to *program* and *configure* the system. Each sensor is assigned a number and type, a master password is programmed for *arming* and *disarming* the system, and telephone number(s) are *input* for *dialing* when a sensor event occurs.

When a sensor event is *recognized*, the software *invokes* an audible alarm attached to the system. After a delay time that is *specified* by the homeowner during system configuration activities, the software dials a telephone number of a monitoring service, *provides* information about the location, *reporting* the nature of the event that has been detected. The telephone number will be *redialed* every 20 seconds until telephone connection is *obtained.*

The homeowner *receives* security information via a control panel, the PC, or a browser, collectively called an interface. The interface *displays* prompting messages and system status information on the control panel, the PC ,or the browser window. Homeowner interaction takes the following form . . .

Potential Classes

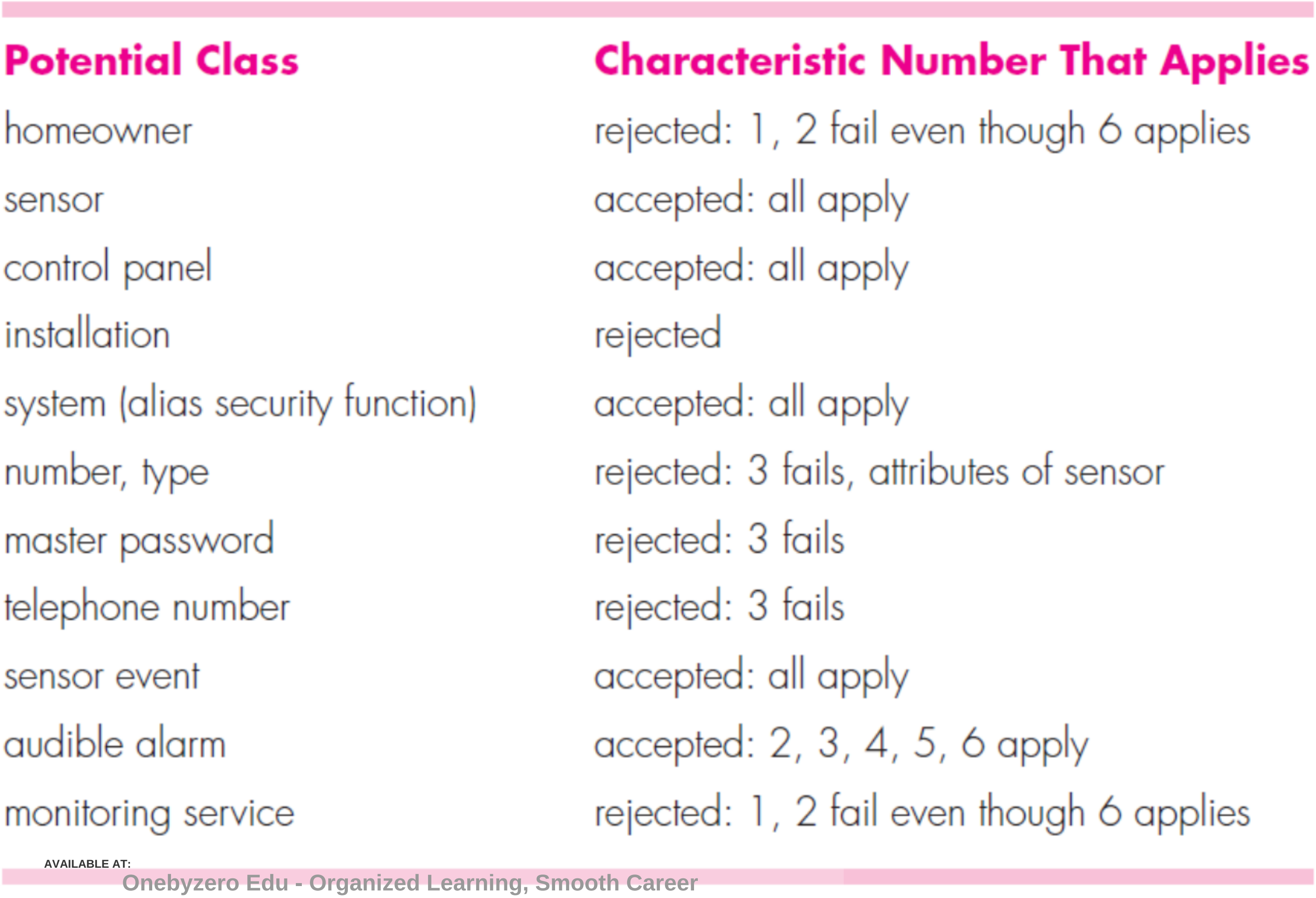


Potential Class	General Classification
homeowner	role or external entity
sensor	external entity
control panel	external entity
installation	occurrence
system (alias security system)	thing
number, type	not objects, attributes of sensor
master password	thing
telephone number	thing
sensor event	occurrence
audible alarm	external entity
monitoring service	organizational unit or external entity

Class Selection Criteria (Coad & Yourdon)

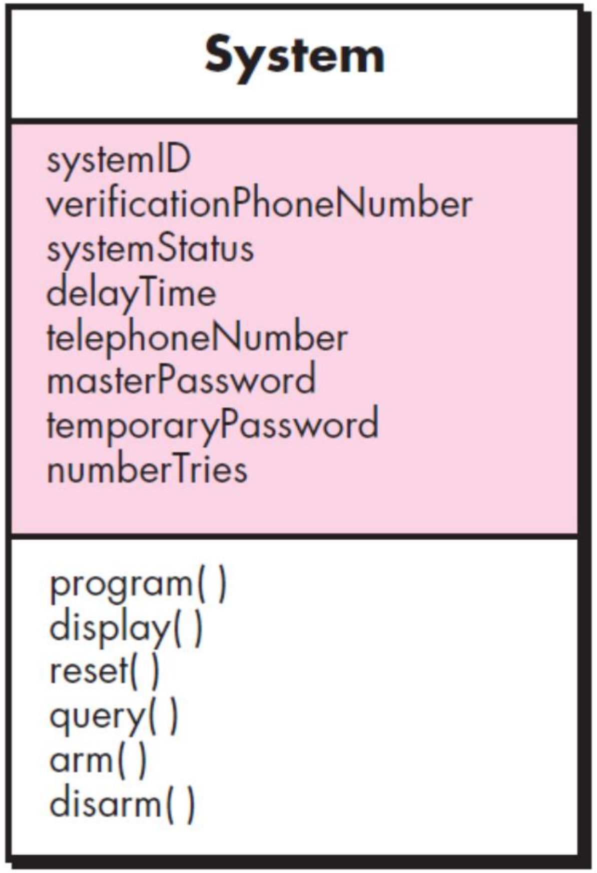
- ***Retained information:*** The potential class will be useful during analysis only if information about it must be remembered so that the system can function. *Class Must hold necessary information. [**Book, member**]*
- ***Needed services:*** The potential class must have a set of identifiable ***operations*** that can ***change the value of its attributes*** in some way. [***book.issueBook()]***
- ***Multiple attributes:*** During requirement analysis, the focus should be on "major" information; a class with a single attribute may, in fact, be useful during design, but is probably better represented as an attribute of another class during the analysis activity.
- ***Common attributes:*** A set of attributes can be defined for the potential class and these attributes apply to all instances of the class. [***Book(title, author, ISBN)***]
- ***Common operations:*** A set of operations can be defined for the potential class and these operations apply to all instances of the class. [***book.add()/edit()***]
- ***Essential requirements :*** ***External entities*** that appear in the problem space and produce or ***consume information*** essential to the operation of any solution for the system will almost always be defined as classes in the requirements model. ***[Student, Teacher, Admin]***

Potential Classes (SafeHome)



Potential Class	Characteristic Number That Applies
homeowner	rejected: 1, 2 fail even though 6 applies
sensor	accepted: all apply
control panel	accepted: all apply
installation	rejected
system (alias security function)	accepted: all apply
number, type	rejected: 3 fails, attributes of sensor
master password	rejected: 3 fails
telephone number	rejected: 3 fails
sensor event	accepted: all apply
audible alarm	accepted: 2, 3, 4, 5, 6 apply
monitoring service	rejected: 1, 2 fail even though 6 applies

System Class for SafeHome (*Class Card*)

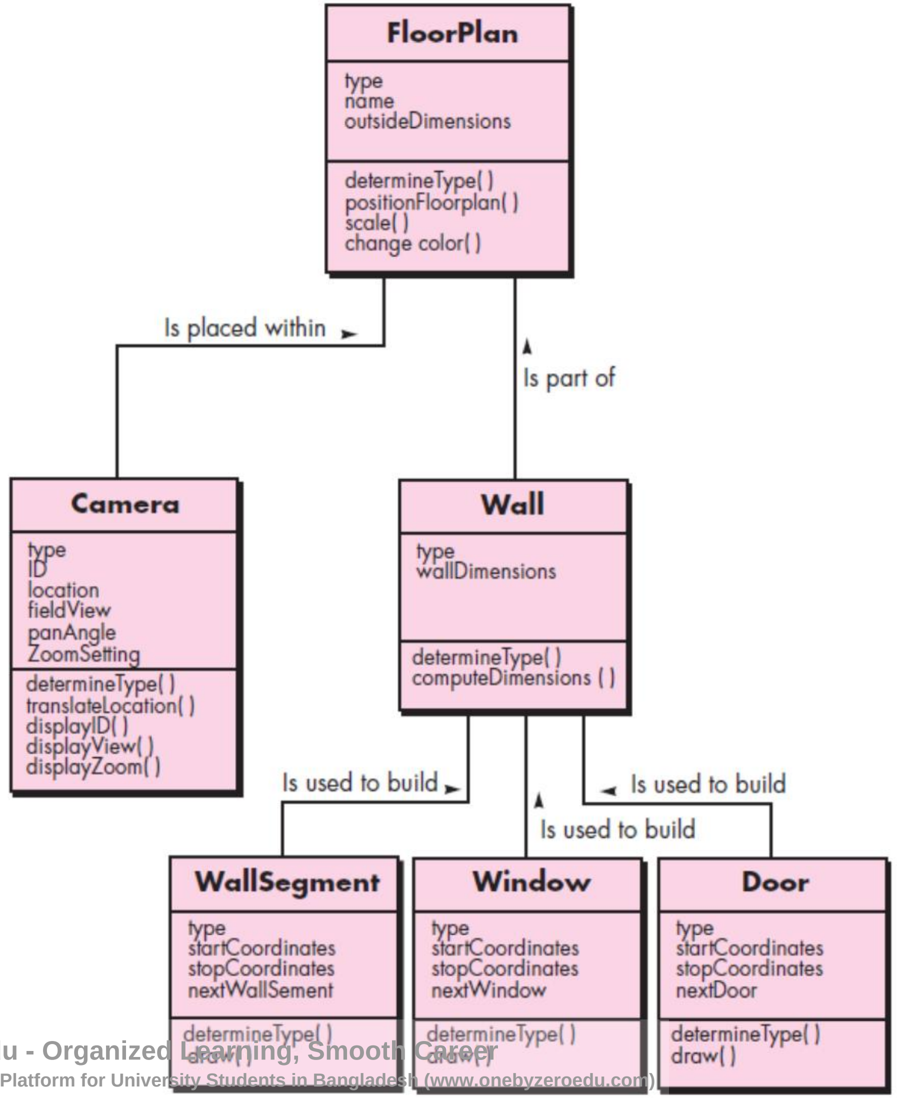


System

systemID
verificationPhoneNumber
systemStatus
delayTime
telephoneNumber
masterPassword
temporaryPassword
numberTries

program( )
display( )
reset( )
query( )
arm( )
disarm( )

Class Diagram for FloorPlan



Chapter 7

- Flow Oriented Modeling
  - Data Flow Diagram
- Behavioral Modeling
  - State Transition Diagram
  - Sequence Diagram

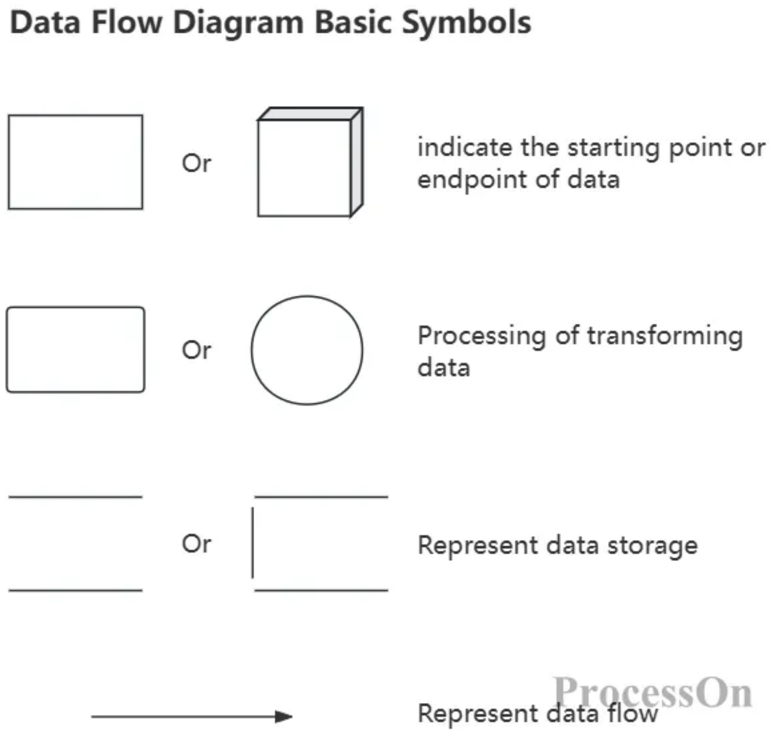
4. Flow Oriented Modeling

- Flow-oriented modeling focuses on **how information moves through a system** and **how that information is transformed**.
- This modeling is used heavily during **requirements analysis** to understand the *functionality* of the system in terms of data movement and data transformation.
- **Flow-oriented modeling : data enters → gets processed → output.**
- **Data Flow Diagram (DFD)** as the *primary tool* of flow-oriented modeling.
- A DFD is a graphical representation that shows:
  - **How data enters the system**
  - **What processes transform the data**
  - **Where data is stored**
  - **How data leaves the system**

DFD

- DFD is presented in a **hierarchical** fashion.
- It starts with a context diagram (or Level 0 DFD) which represents the whole system.
- Then break it down into Level 1, Level 2, etc.
- Represent only ***data flow***, *not control flow*
- Focus on what ***happens***, *not how it happens*
- *Avoid* implementation details (no loops, conditions, IF statements)
- ***Data objects*** are represented by ***labeled arrows,*** and ***transformations*** are represented by ***circles*** (also called bubbles).

Symbols for DFD



Data Flow Diagram Basic Symbols

indicate the starting point or endpoint of data

Processing of transforming data

Represent data storage

Represent data flow

DFD Level 0 or Context Levele Diagram of SafeHome Security Function

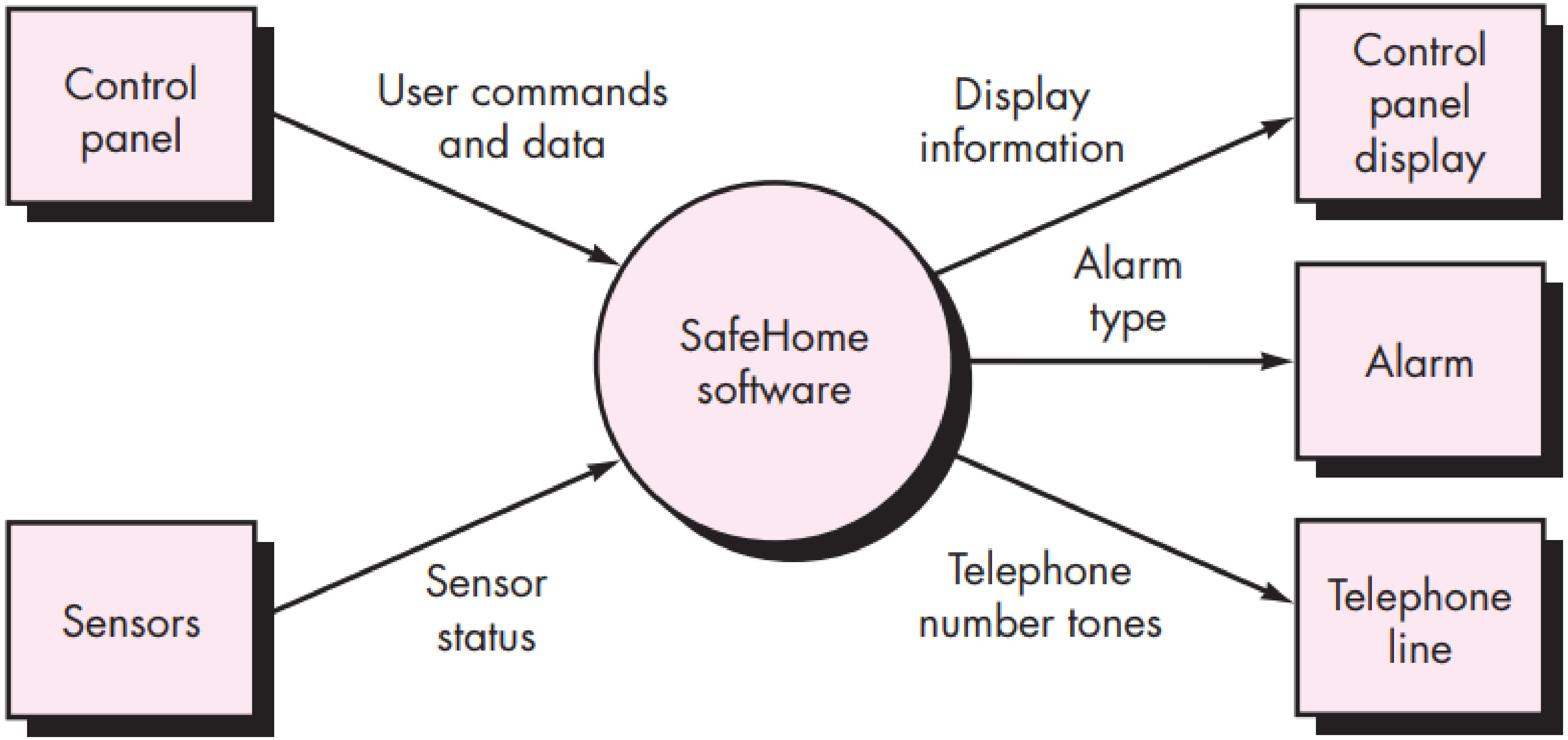


FIGURE 7.2

Level 1 DFD for *SafeHome* security function

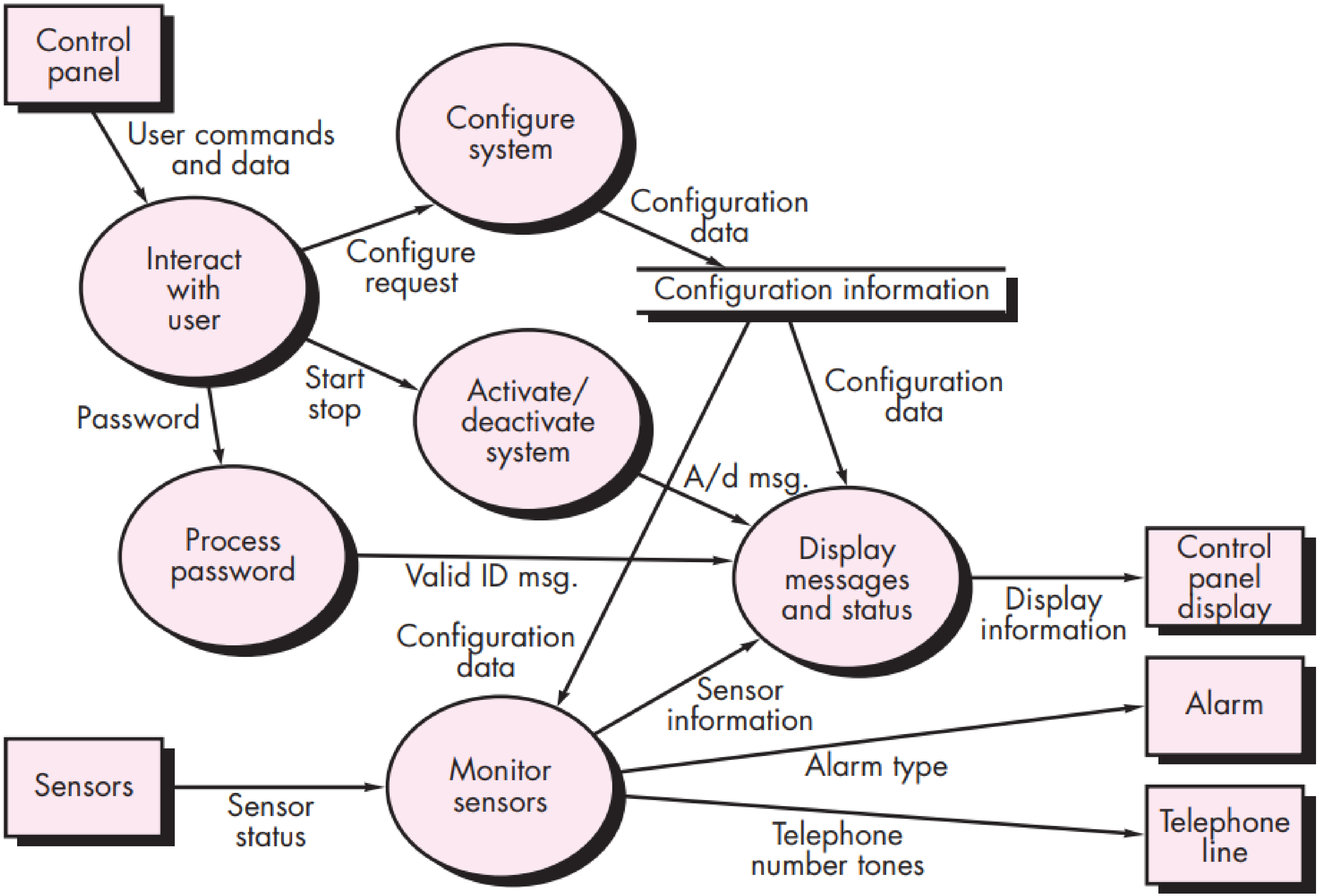
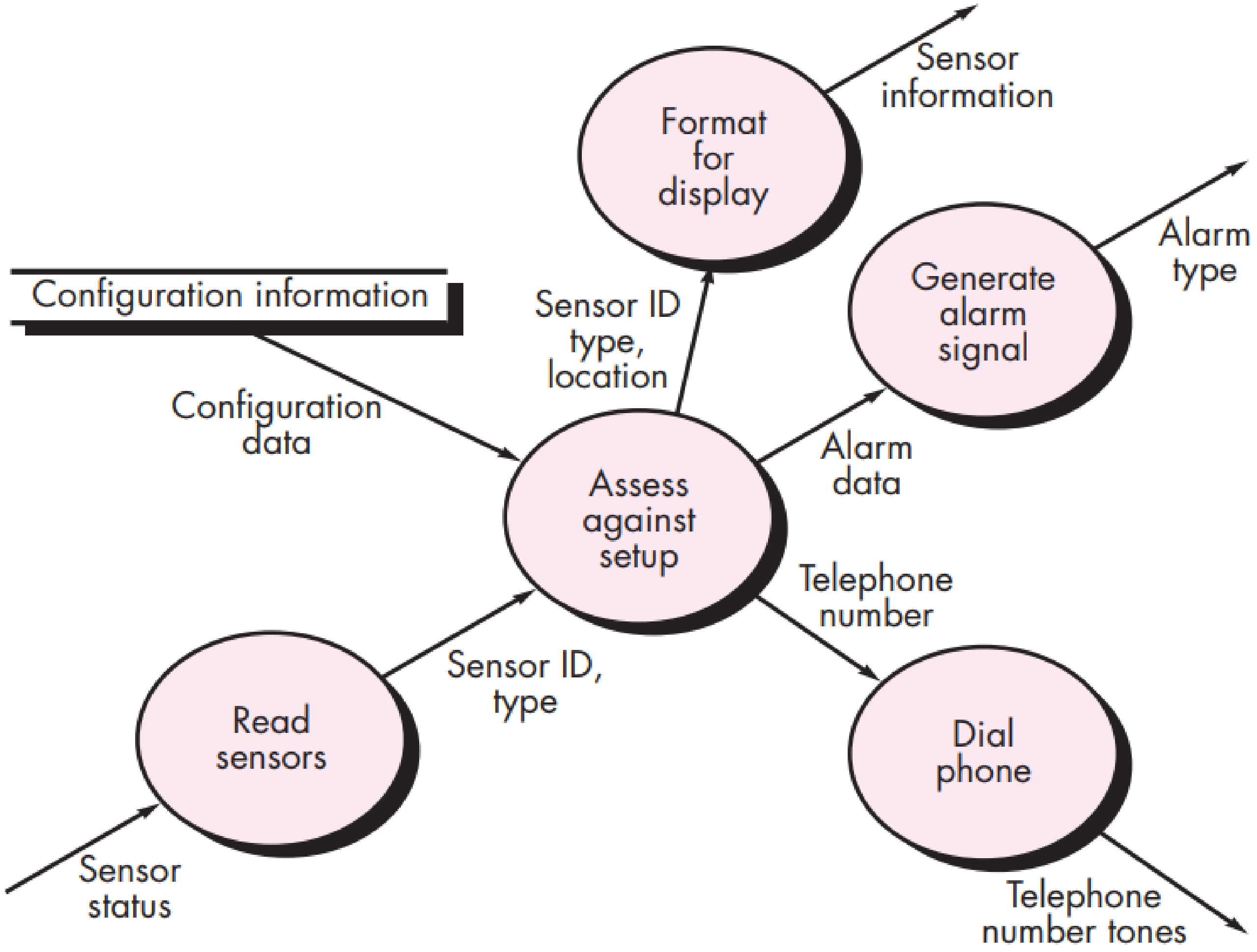


FIGURE 7.3

Level 2 DFD that refines the *monitor sensors* process



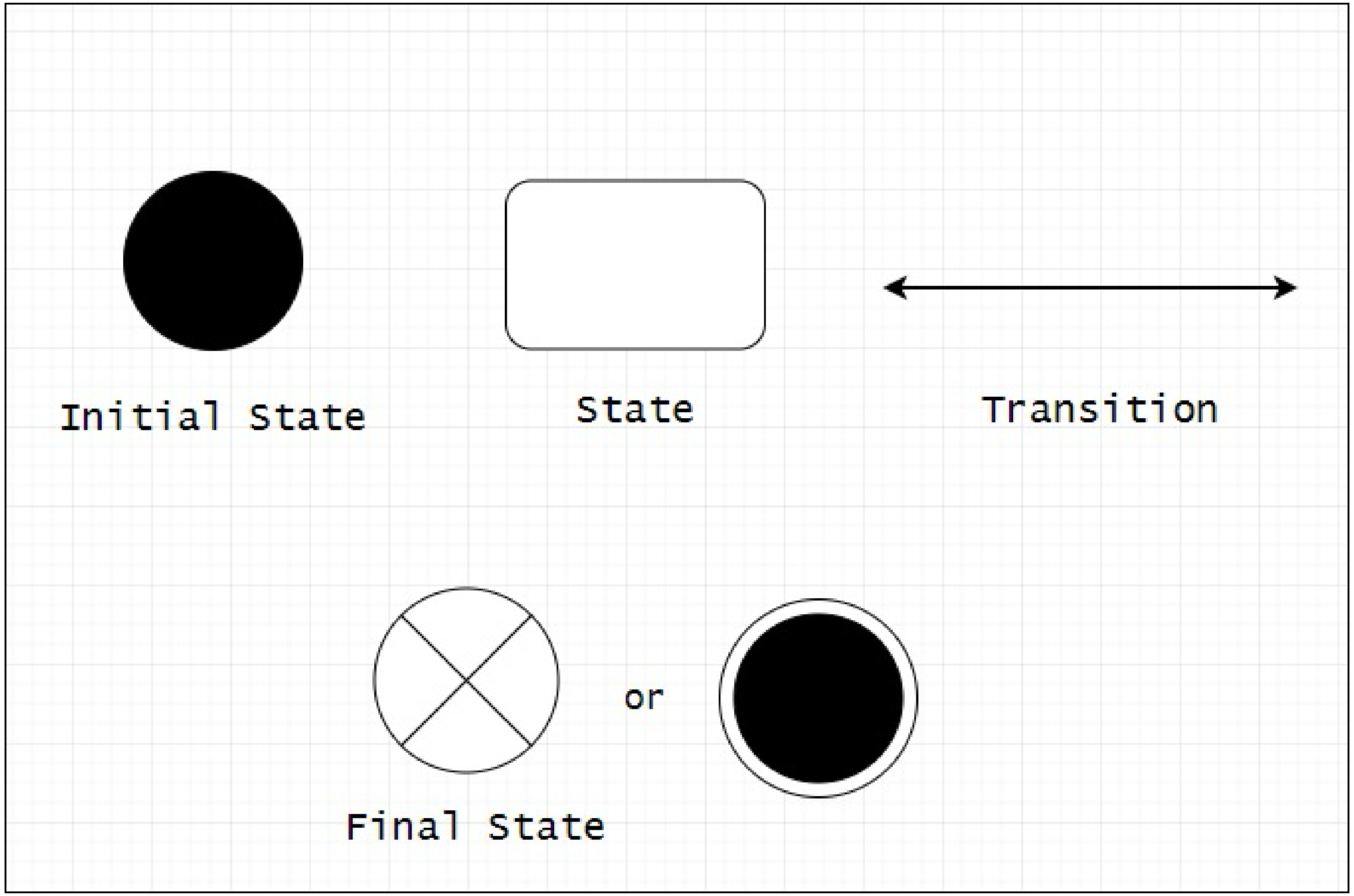
5. Behavioral Modeling

- Behavioral modeling focuses on **how a system behaves**, rather than how it is ***structured***.
- It captures the **dynamic aspects** of a system, how objects or components ***interact over time***.
- To understand **system responses to external and internal events**.
- Behavioral modeling can be represented in two ways:
    - State Transition Diagram
    - Sequence Diagram

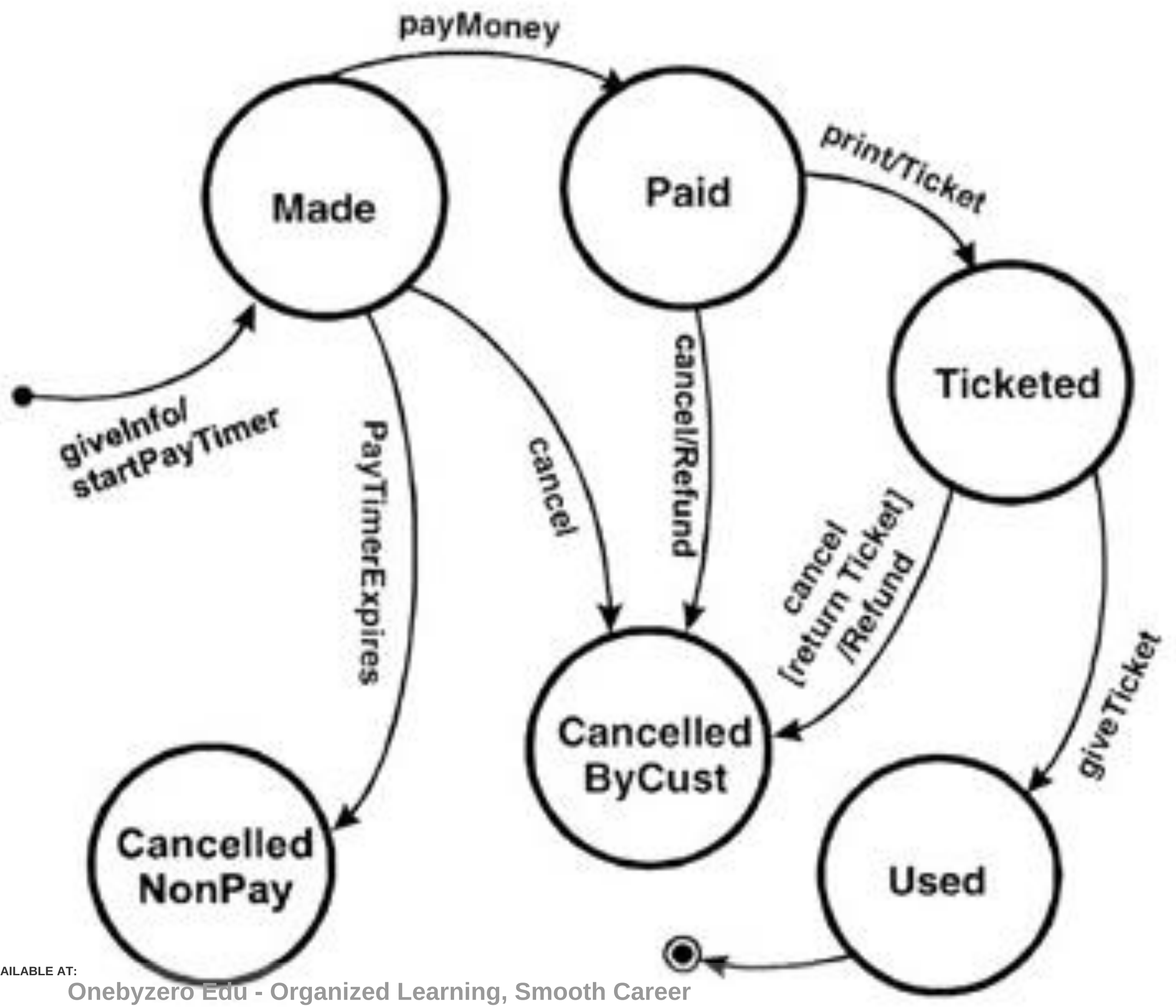
State Transition Diagram

- Shows the different **states** an object or system can be in, and how it transitions from one state to another in response to **events**.
- Helps to understand **how the system reacts** to events over time.
- **Elements:**
  - **States**: Represent different conditions of the system/object.
  - **Transitions**: Arrows showing movement from one state to another.
  - **Events**: Triggers causing the transitions.
  - **Actions**: Activities performed during a transition.

Symbols for STD



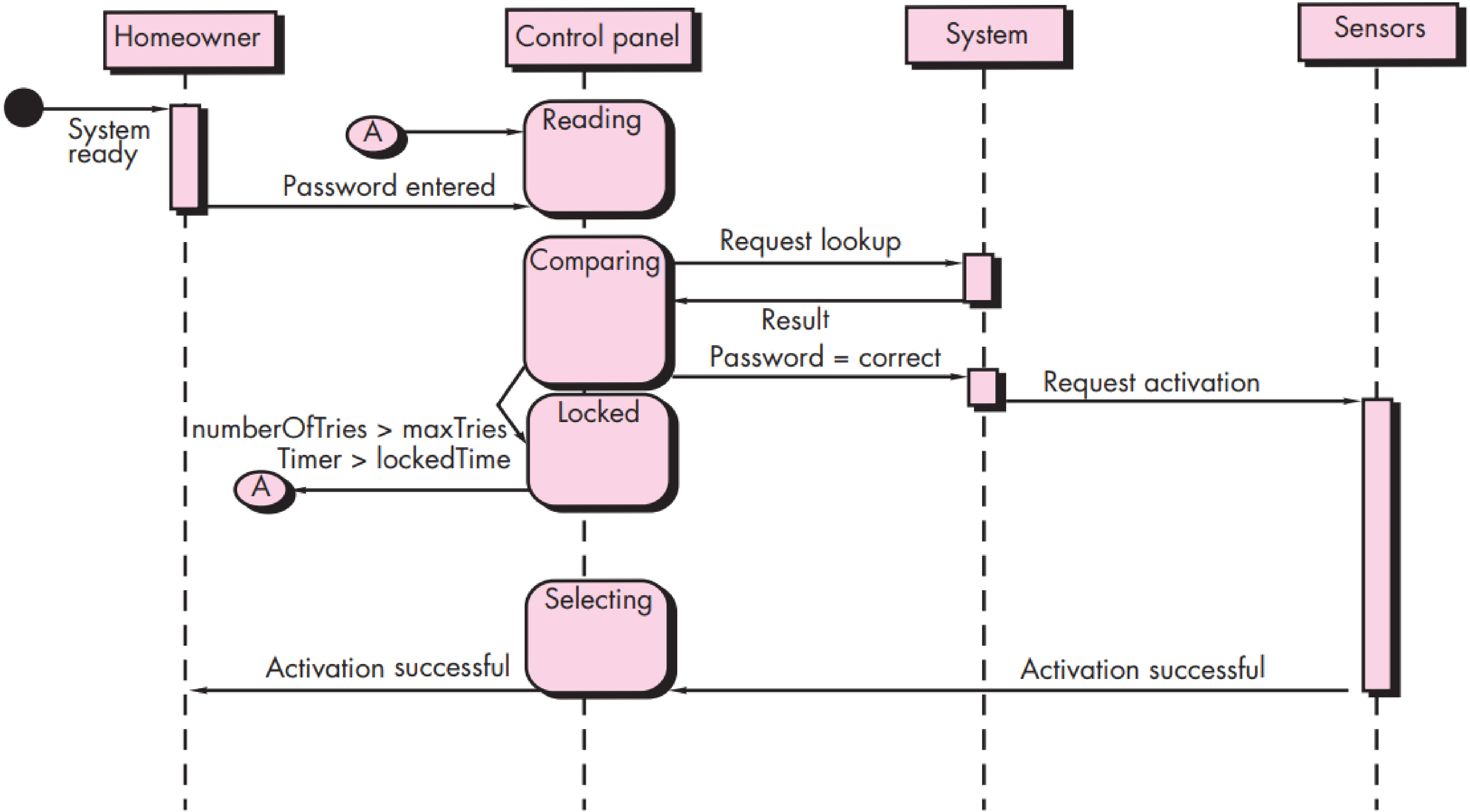
STD for Ticket reservation System



Sequence Diagram

- Illustrates **how objects interact** with each other in a **time sequenced manner** to perform a particular functionality.
- Interaction between **objects** and the **order of messages** exchanged.
- Helps in understanding **message flow**, **interaction patterns**, and **dynamic collaboration**.
- **Elements:**
  - **Objects**: Represented by vertical lifelines.
  - **Messages**: Horizontal arrows showing communication between objects.
  - **Activation bars**: Periods during which an object is performing an action.

FIGURE 7.7 Sequence diagram (partial) for the *SafeHome* security function



References

- Chpater 6 & 7: Requirements Modeling(Pressman - 7 edition)

Thank you