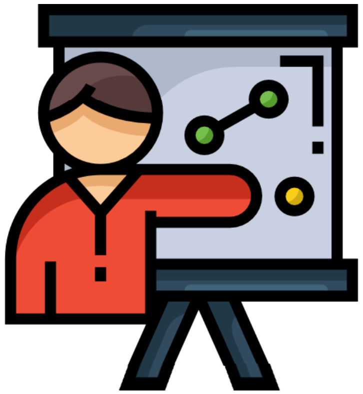
Unit 2.3
Requirement Analysis

/ Elaboration

Requirement Analysis & Specification

Requirements Engineering Tasks cont.

3 Elaboration



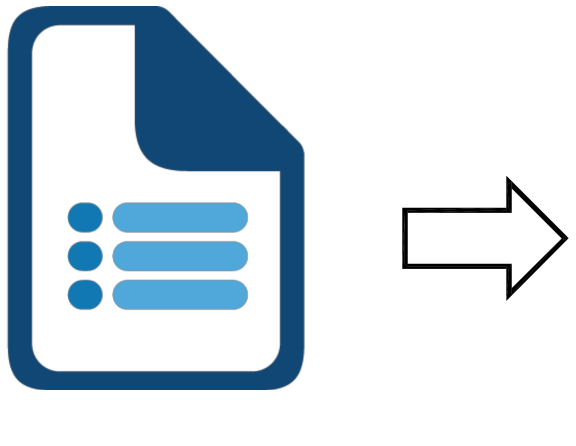
- Further define requirements
- Expand and refine requirements obtained from inception & elicitation
- Creation of User scenarios, extract analysis class and business domain entities

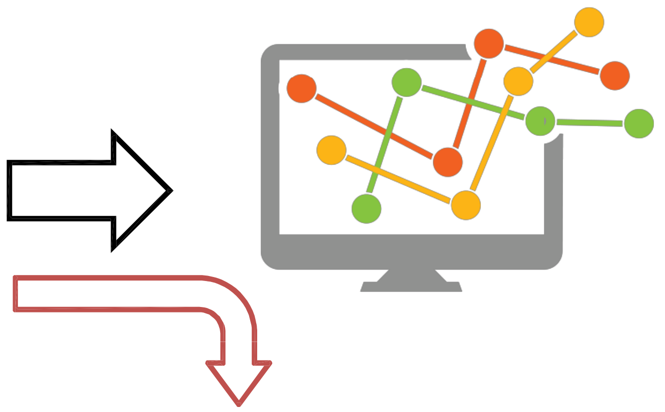
The requirement analysis model

System Description

Analysis Model

Design Model



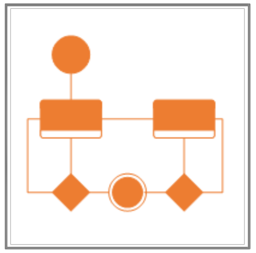


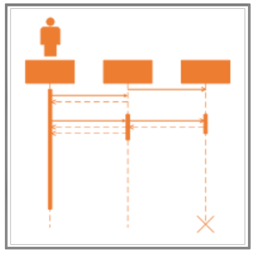
Purpose	
	Describe what the customer wants built
	Establish the foundation for the software design
	Provide a set of validation requirements

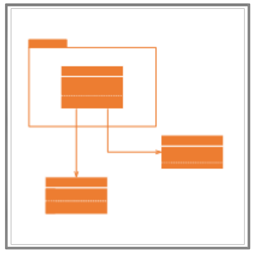
System Information

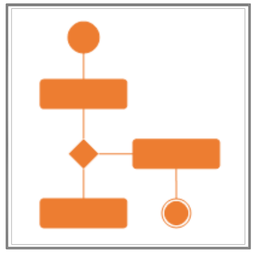
System Function

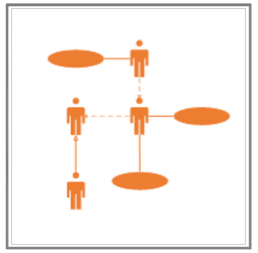
System Behaviors



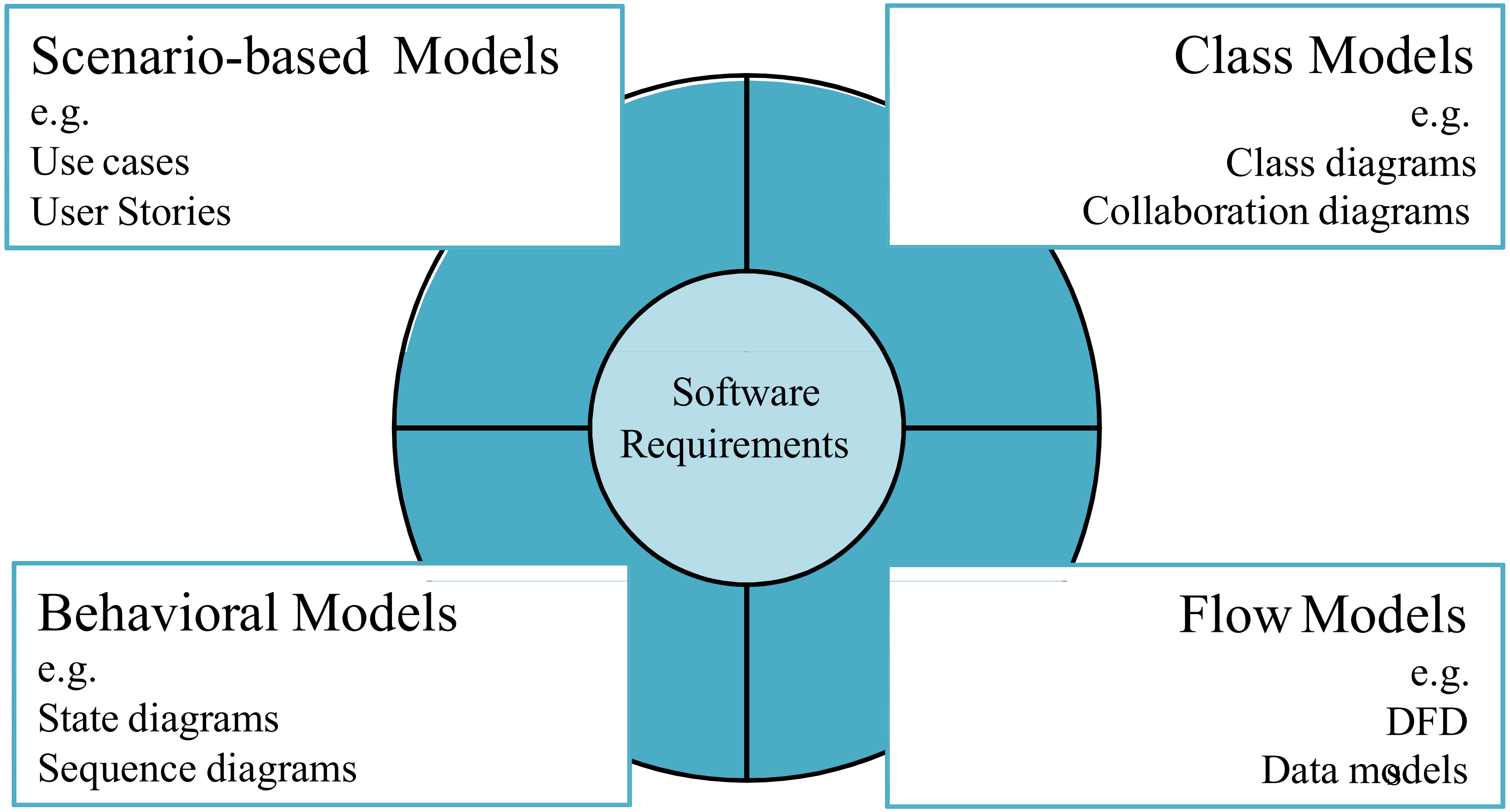








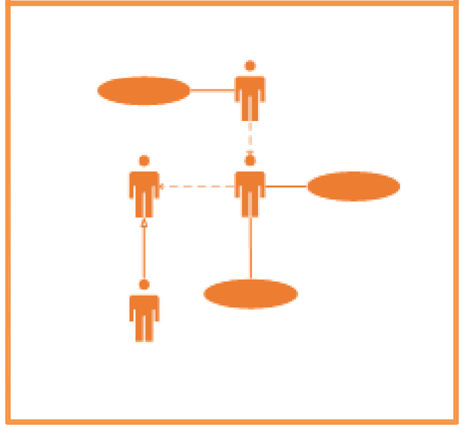
Elements of the Requirements Model



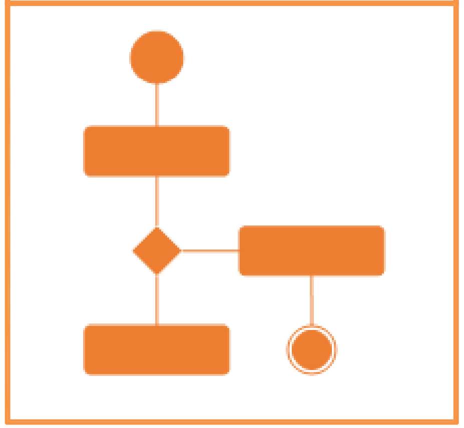
Elements of the Requirements Model

- Scenario-based elements
  - Describe the system from the user's point of view using scenarios that are depicted (stated) in use cases and activity diagrams
- Class-based elements
  - Identify the domain classes for the objects manipulated by the actors, the attributes of these classes, and how they interact with one another; which utilize class diagrams to do this.

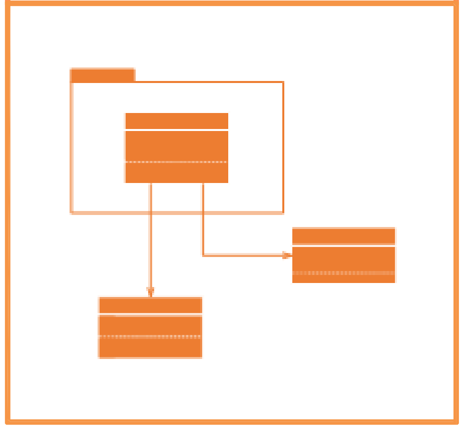
Use Case Diagram



Activity Diagram

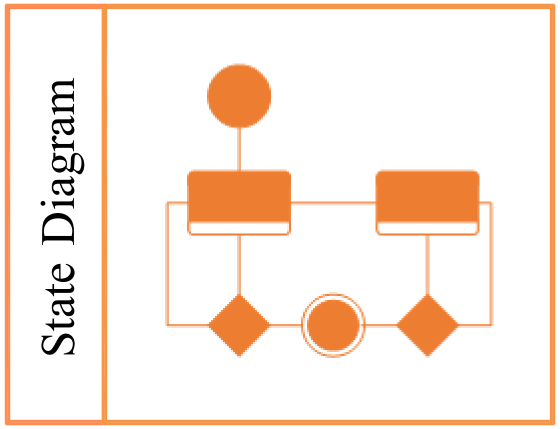


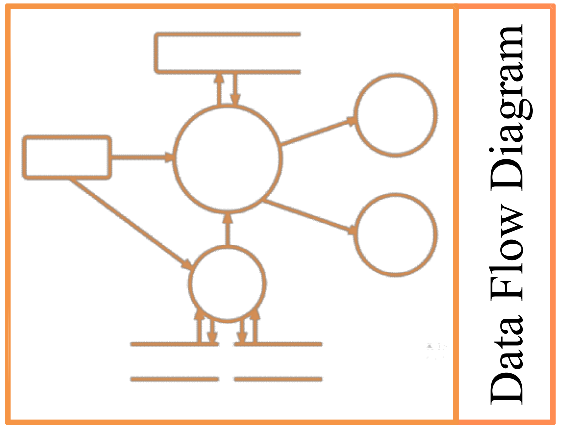
Class Diagram



Elements of the Requirements Model

- Behavioral elements
  - Use state diagrams to represent the state of the system, the events that cause the system to change state, and the actions that are taken as a result of a particular event.
  - This can also be applied to each class in the system.
- Flow-oriented elements
  - Use data flow diagrams to show the input data that comes into a system, what functions are applied to that data to do transformations, and what resulting output data are produced.





Analysis rule of Thumb

- ☐ Make sure all points of view are covered
- ☐ Every element should add value
- ☐ Keep it simple
- ☐ Maintain a high level of abstraction
- ☐ Focus on the problem domain
- ☐ Minimize system coupling
- ☐ Model should provides value to all stakeholders

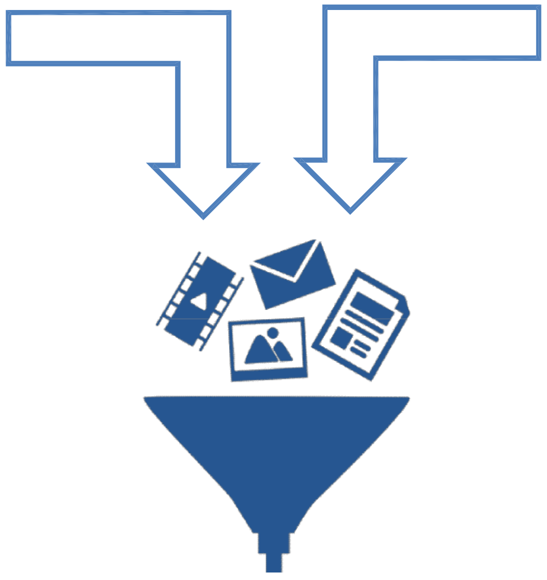
Analysis Modeling Approaches

Structured Analysis

- Models data elements
  - Attributes
  - Relationships
- Models processes that transform data

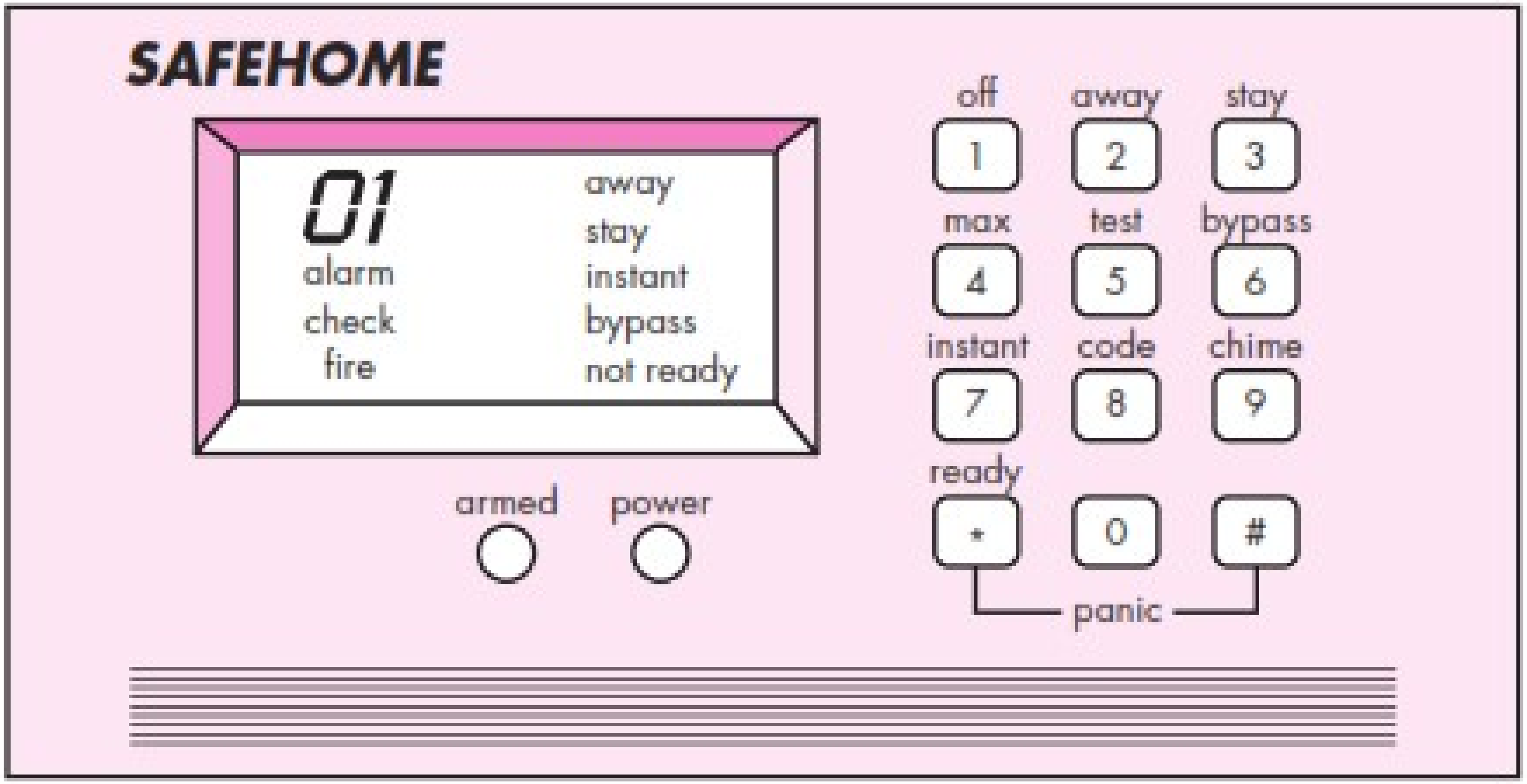
Object Oriented Analysis

- Models analysis classes
  - Data
  - Processes
- Models class collaborations



Techniques from both approaches are typically used in practice.

Safe Home Control Panel



Use-Cases

- A collection of user scenarios that describe the thread of usage of a system
- Each scenario is described from the point-of-view of an "actor"
  - Actor: a person or device that interacts with the software
- Each scenario answers the following questions:
  - Who is the primary actor, the secondary actor (s)?
  - What are the actor's goals?
  - What preconditions should exist before the story begins? main
  - What tasks or functions are performed by the actor? extensions
  - What might be considered as the story is described?

A **Primary actor** is one having a goal requiring the assistance of the system

A **Secondary actor** is one from which System needs assistance

Scenerio Based Model :Use-Cases

- What variations in the actor's interaction are possible?
- What system information will the actor acquire, produce, or change?
- Will the actor have to inform the system about changes in the external environment?
- What information does the actor desire from the system?
- Does the actor wish to be informed about unexpected changes?

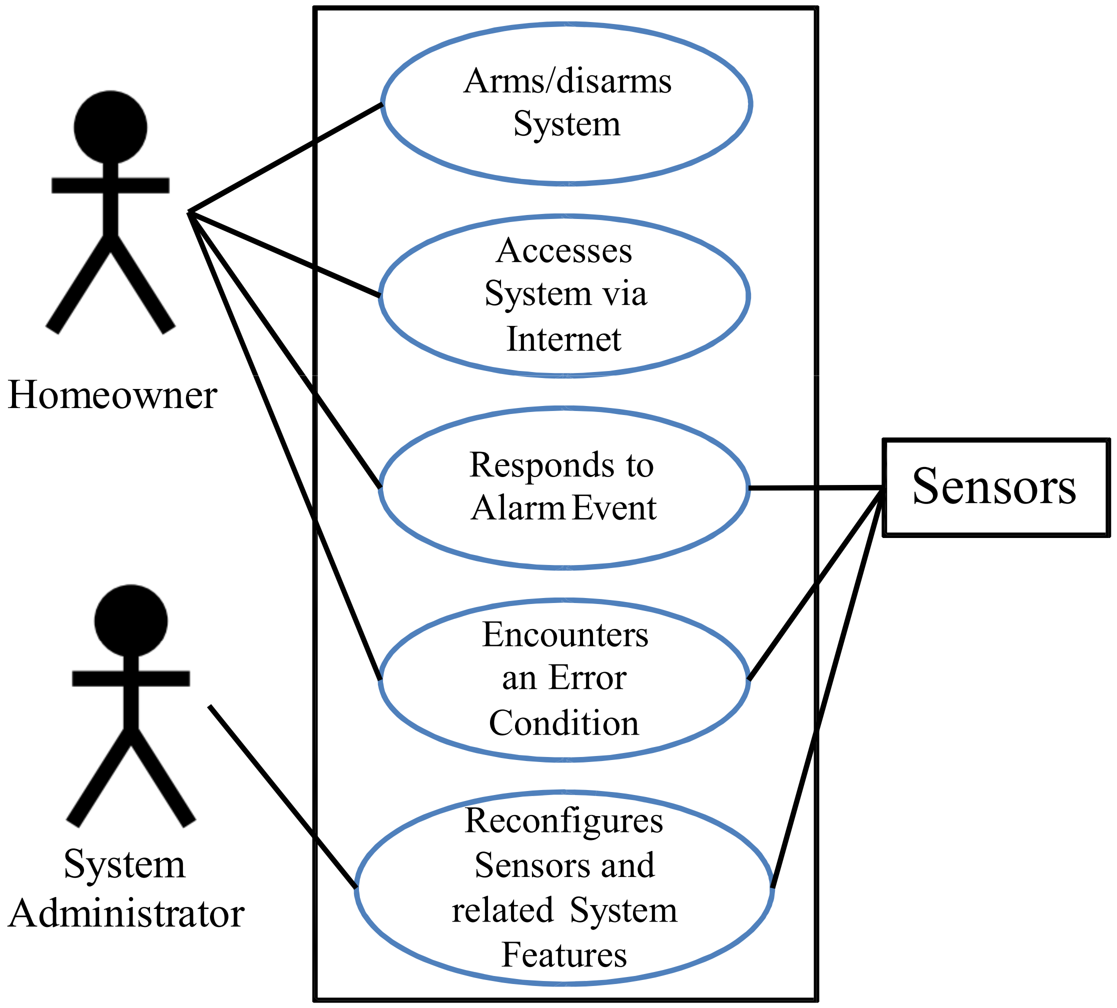
Thus

Use Case captures who (actor) does what (interaction) with the system, for what purpose (goal), without dealing with system internals

Use-Case Diagram

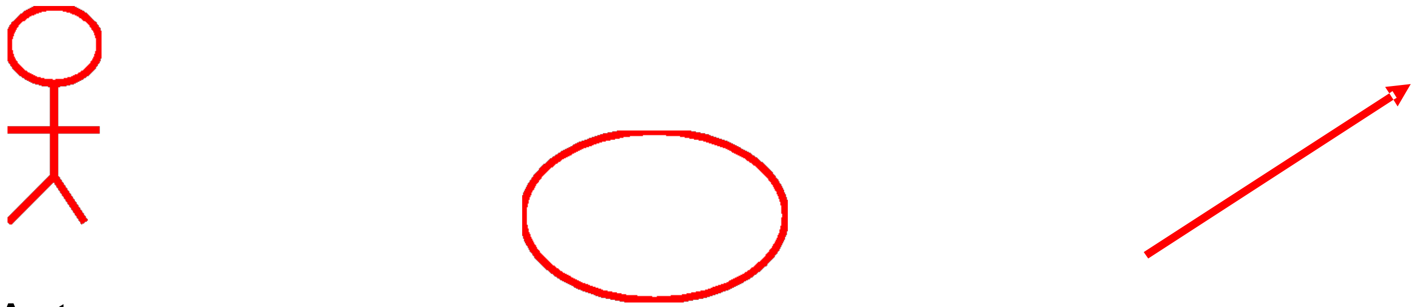
- It is referred as the diagram used to describe a set of actions (use cases) that some system should perform in collaboration with system users.

Use Case diagram for SafeHome home security function


Homeowner
System Administrator
Arms/disarms System
Accesses System via Internet
Responds to Alarm Event
Encounters an Error Condition
Reconfigures Sensors and related System Features
Sensors

Use-Case Diagram

-- represents what happens when actor interacts with a system
-- captures functional aspect of the system



Actor

Use Case

Relationship between actors and use case and/or between the use cases

-- Actors appear outside the rectangle

--Use cases within rectangle providing functionality

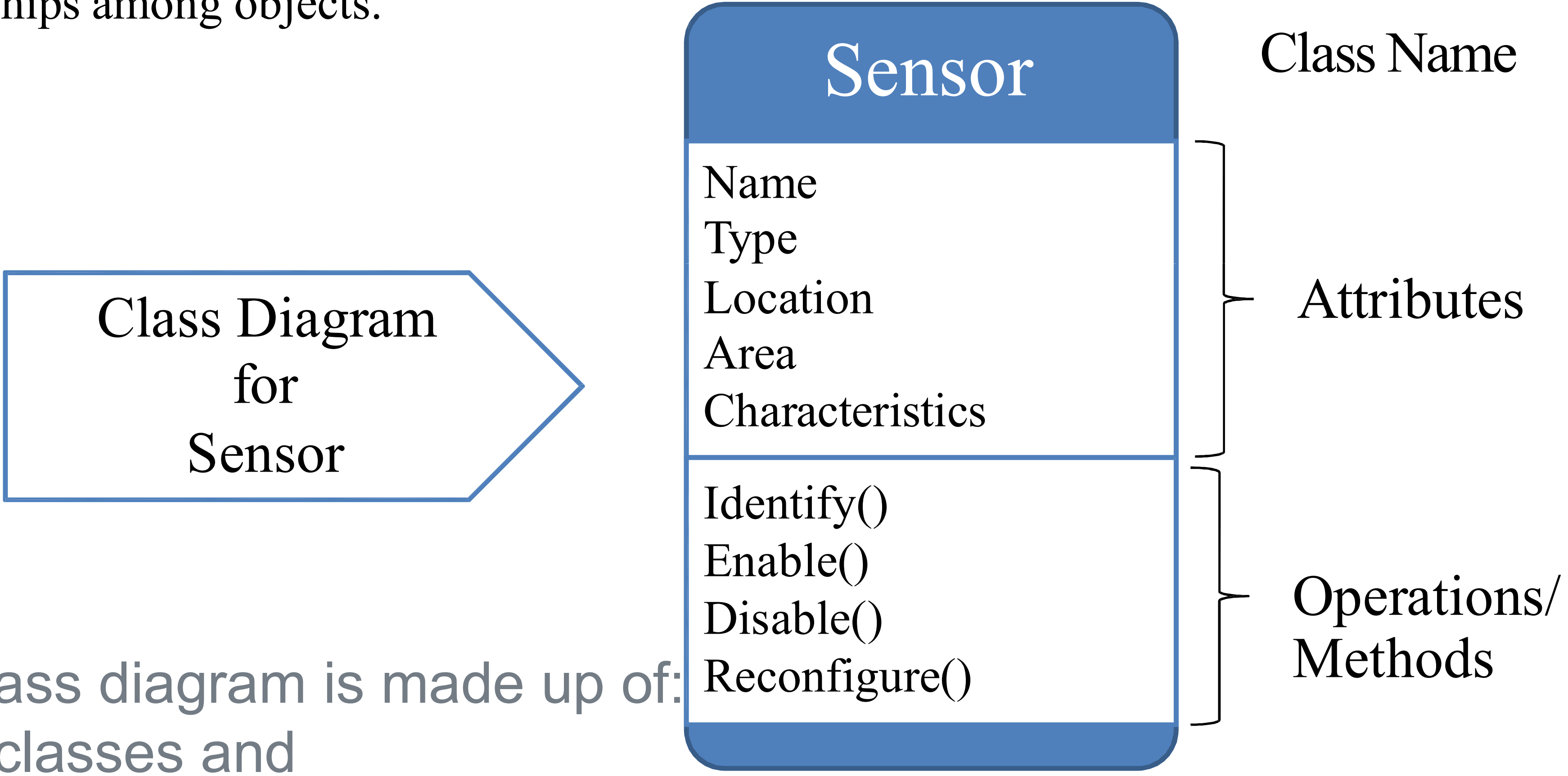
--Relationship association is a solid line between actor & use cases

Use-Case

1	Use Case Title	Login
2	Abbreviated Title	Login
3	Use Case Id	1
4	Actors	Librarian , Members, Asst. Librarian
5	Description: To interact with the system, LMS will validate its registration with this system. It also defines the actions a user can perform in LMS.	
5.1	Pre Conditions: User must have proper client installed on user terminal	
5.2	Task Sequence 1. System show Login Screen 2. User Fill in required information. Enter user name and password 3. System acknowledge entry	
5.3	Post Conditions: System transfer control to user main screen to proceed further actions	
5.4	Exception: If no user found then system display Invalid user name password error message and transfer control to Task Sequence no.1	
6	Modification history: Date 08-01-2018	
7	Author: Pradyumansinh Jadeja Project ID LMS	

Class Model: Class Diagram

In software engineering, a class diagram in the **Unified Modeling Language (UML)** is **a type of static structure diagram** that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.



A UML class diagram is made up of:

- A set of classes and
- A set of relationships between classes

Class Notation

A class notation consists of three parts:

1.Class Name

1. The name of the class appears in the first partition.

2.Class Attributes

1. Attributes are shown in the second partition.
2. The attribute type is shown after the colon.
3. Attributes map onto member variables (data members) in code.

3.Class Operations (Methods)

1. Operations are shown in the third partition. They are services the class provides.
2. The return type of a method is shown after the colon at the end of the method signature.
3. The return type of method parameters is shown after the colon following the parameter name.
4. Operations map onto class methods in code

Class Notation

A class notation consists of three parts:

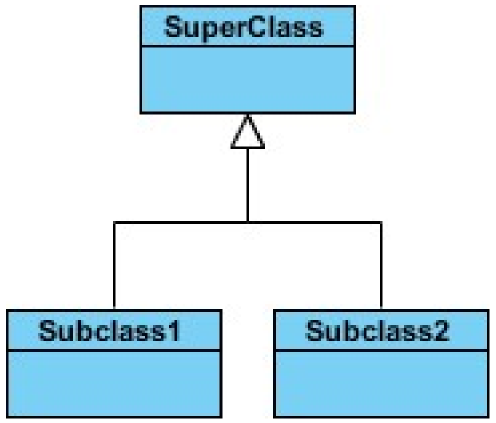
1. **Class Name**
    1. The name of the class appears in the first partition.
2. **Class Attributes**
    1. Attributes are shown in the second partition.
    2. The attribute type is shown after the colon.
    3. Attributes map onto member variables (data members) in code.
3. **Class Operations** (Methods)
    1. Operations are shown in the third partition. They are services the class provides.
    2. The return type of a method is shown after the colon at the end of the method signature.
    3. The return type of method parameters is shown after the colon following the parameter name.
    4. Operations map onto class methods in code

Class Relationship

A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:

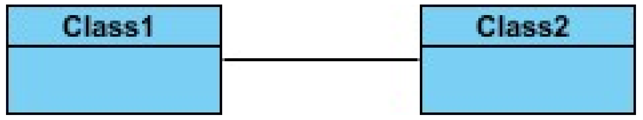
Inheritance (or Generalization):

- Represents an "is-a" relationship.
- An abstract class name is shown in italics.
- SubClass1 and SubClass2 are specializations of Super Class.
- A solid line with a hollow arrowhead that point from the child to the parent class



Simple Association:

- A structural link between two peer classes.
- There is an association between Class1 and Class2
- A solid line connecting two classes

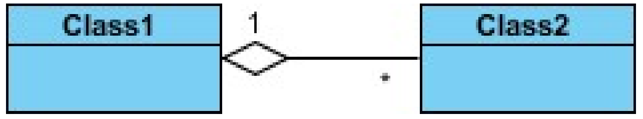


Class Relationship

Aggregation:

A special type of association. It represents a "part of" relationship.

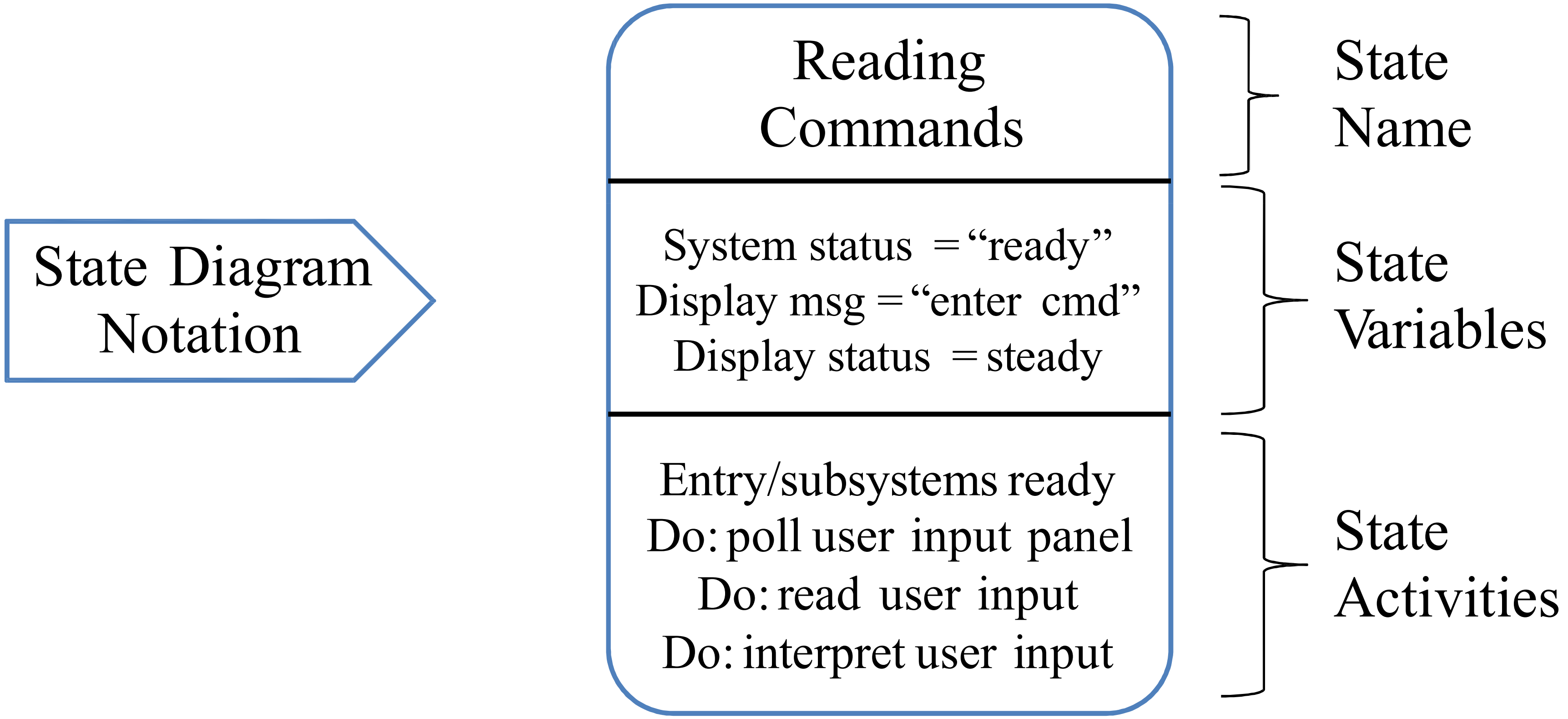
- Class2 is part of Class1.
- Many instances (denoted by the *) of Class2 can be associated with Class1.
- Objects of Class1 and Class2 have separate lifetimes.
- A solid line with an unfilled diamond at the association end connected to the class of composite



Behavioral Models : State Diagram

- It is used to describe the behaviour of systems.
- It requires that the system described is composed of a finite number of states.

A state diagram consists of states, transitions, events, and activities. You use state diagrams to illustrate the dynamic view of a system. They are especially important in modelling the behavior of an interface, class, or collaboration. State diagrams emphasize the event-ordered behavior of an object, which is especially useful in modelling reactive systems.

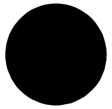
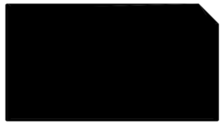
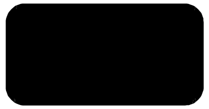
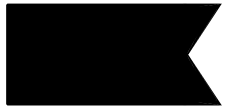
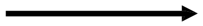
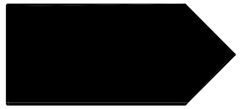
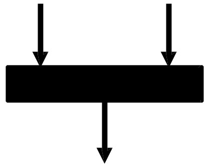
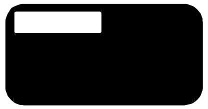
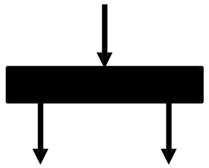
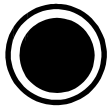
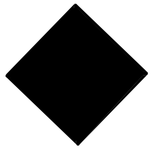


Activity & Swimlane Diagram

A **Swimlane diagram** is a type of **flowchart** that delineates who does what in a process. Using the metaphor of lanes in a pool, a **Swimlane diagram** provides clarity and accountability by placing process steps within the horizontal or vertical "**Swimlanes**" of a particular employee, work group or department.

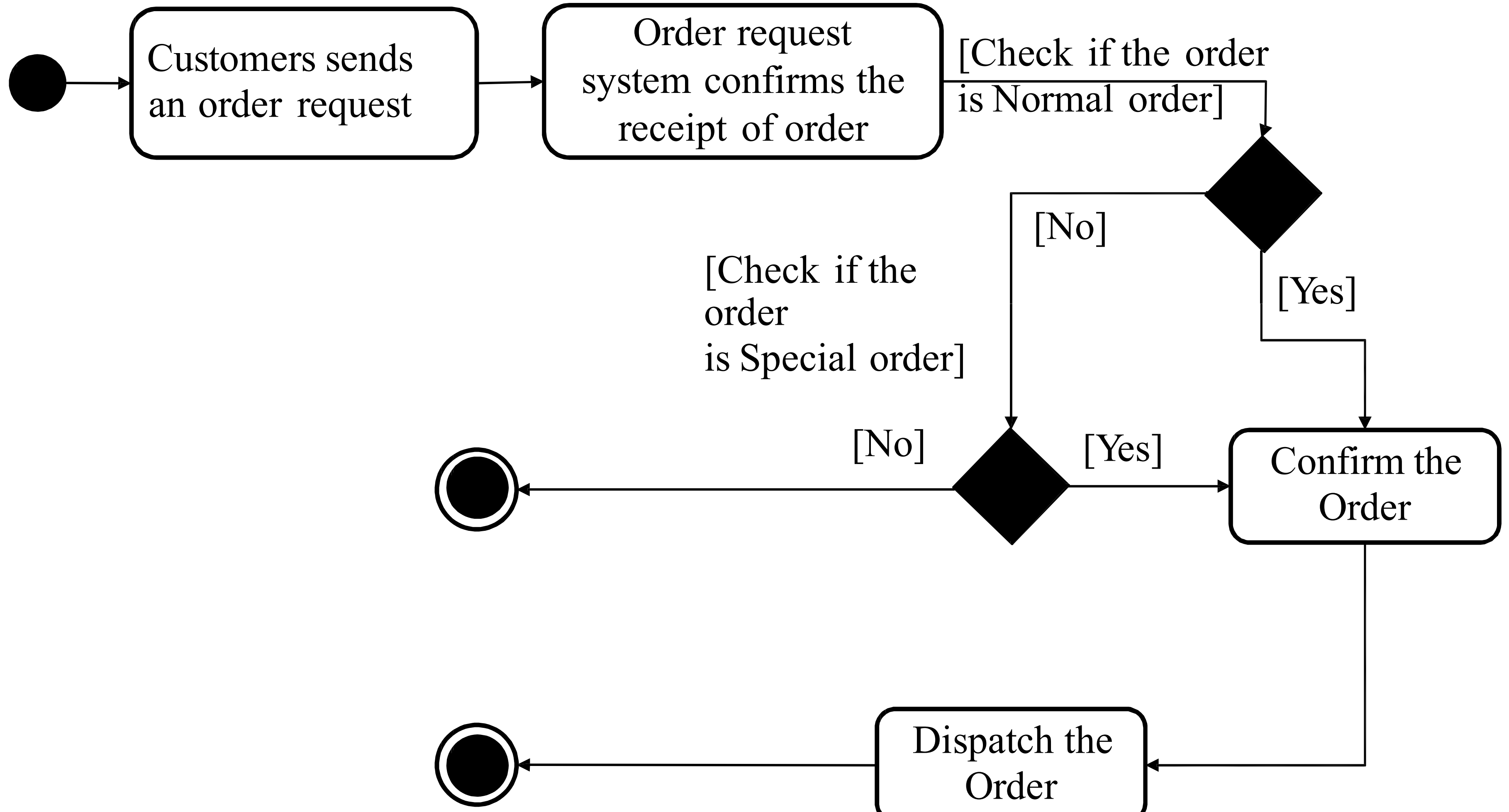
- Activity diagram is basically a flowchart to represent the flow from one activity to another activity
- The activity can be described as an operation of the system.
- A swimlane diagram is a type of activity diagram. Like activity diagram, it diagrams a process from start to finish, but it also divides these steps into categories to help distinguish which departments or employees are responsible for each set of actions
- A swim lane diagram is also useful in helping clarify responsibilities and help departments work together in a world where departments often don't understand what the other departments do

Activity Diagram Symbols

	Start		Note
	Activity		Receive Signal
	Connector		Send Signal
	Join		Option Loop
	Fork		End
	Decision		

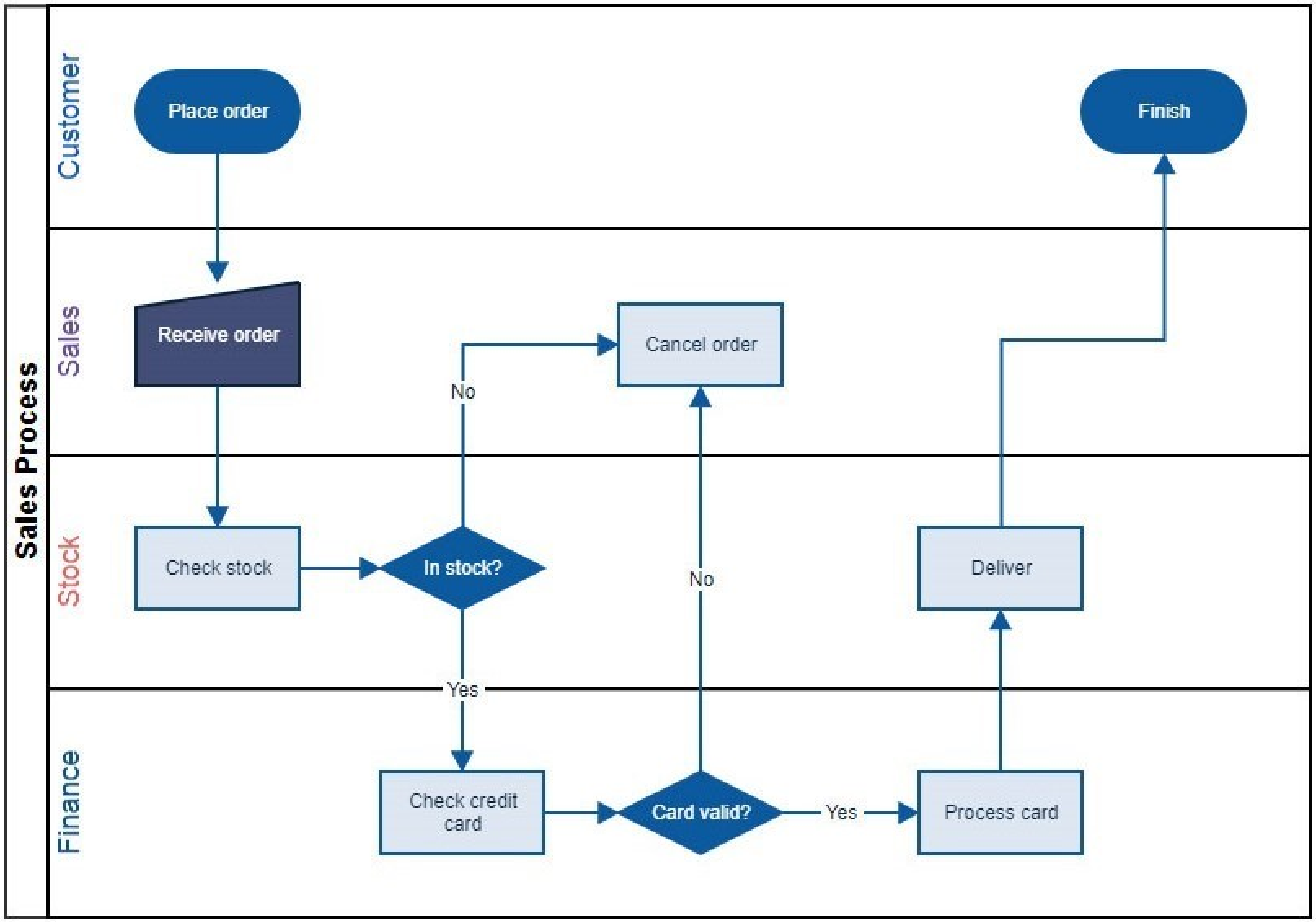
Activity diagram of order processing

Send order by the customer, Receipt of the order, Confirm the order, Dispatch the order

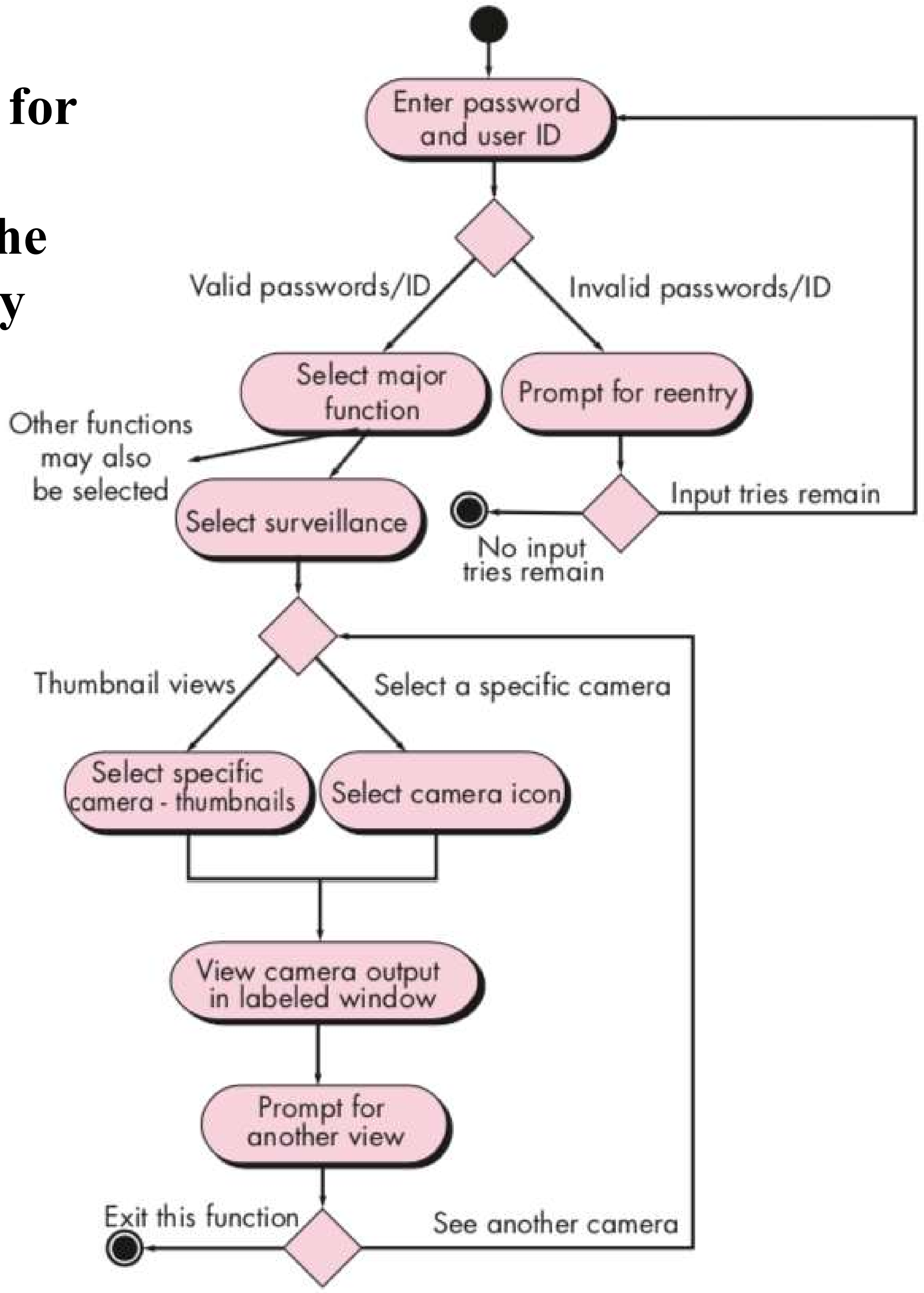


Customers sends an order request
Order request system confirms the receipt of order
[Check if the order is Normal order]
[No]
[Yes]
[Check if the order is Special order]
[No]
[Yes]
Confirm the Order
Dispatch the Order

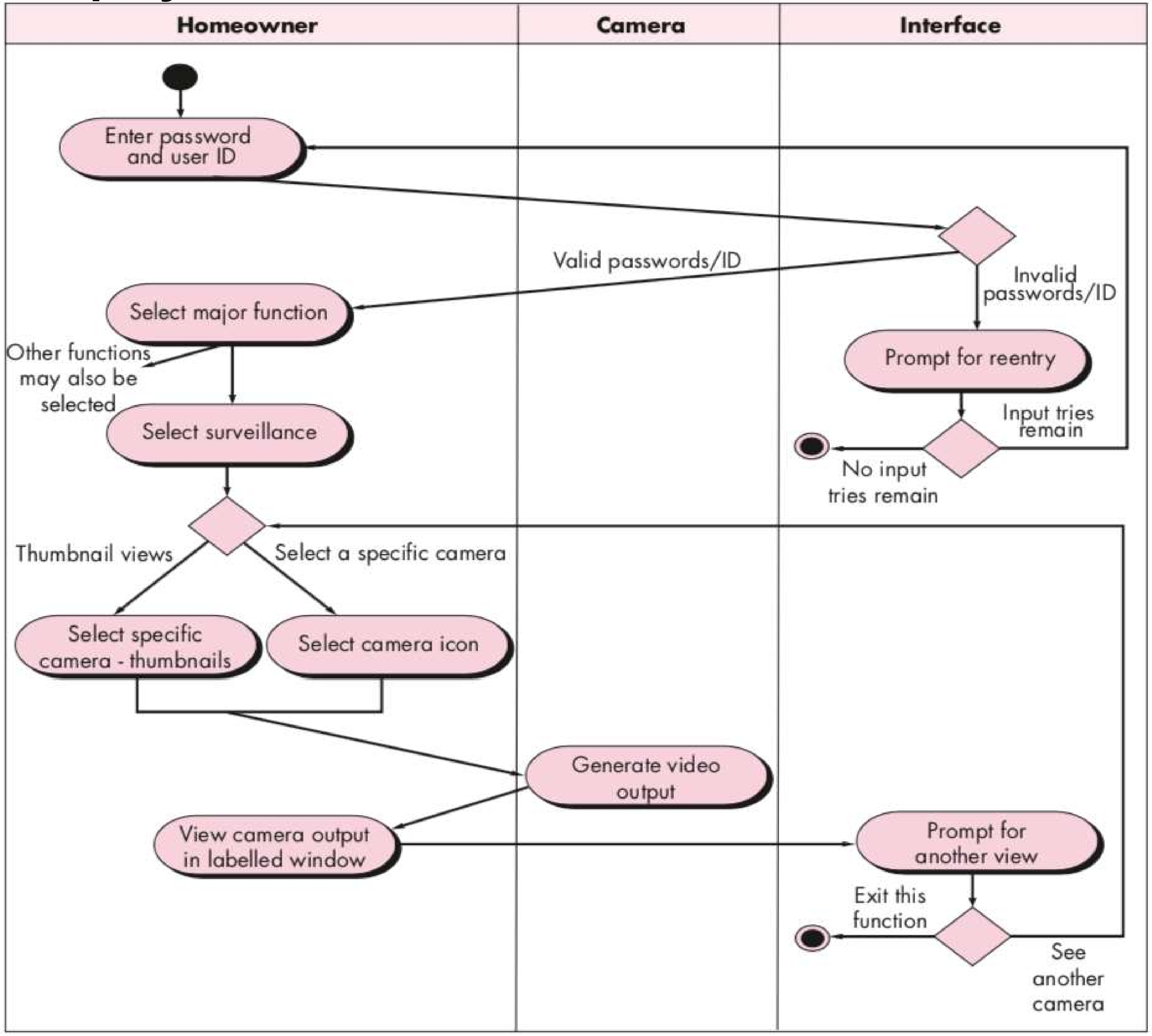
Swimlane diagram of order processing



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

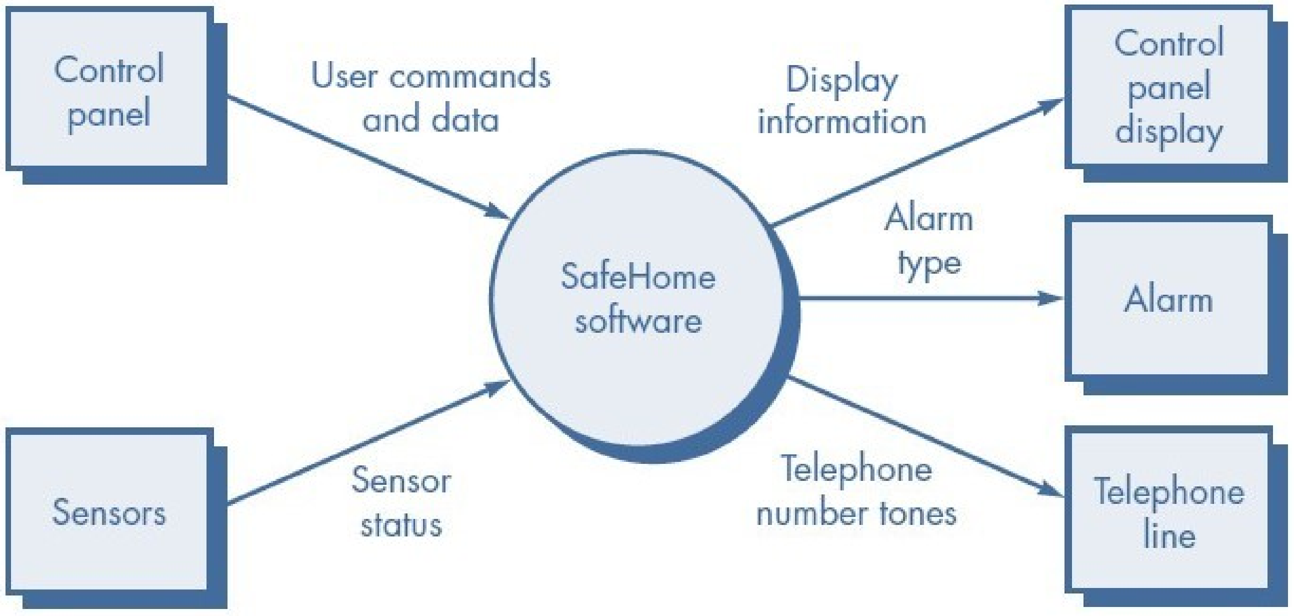


Swimlane diagram for Access camera surveillance via the Internet—display camera views function



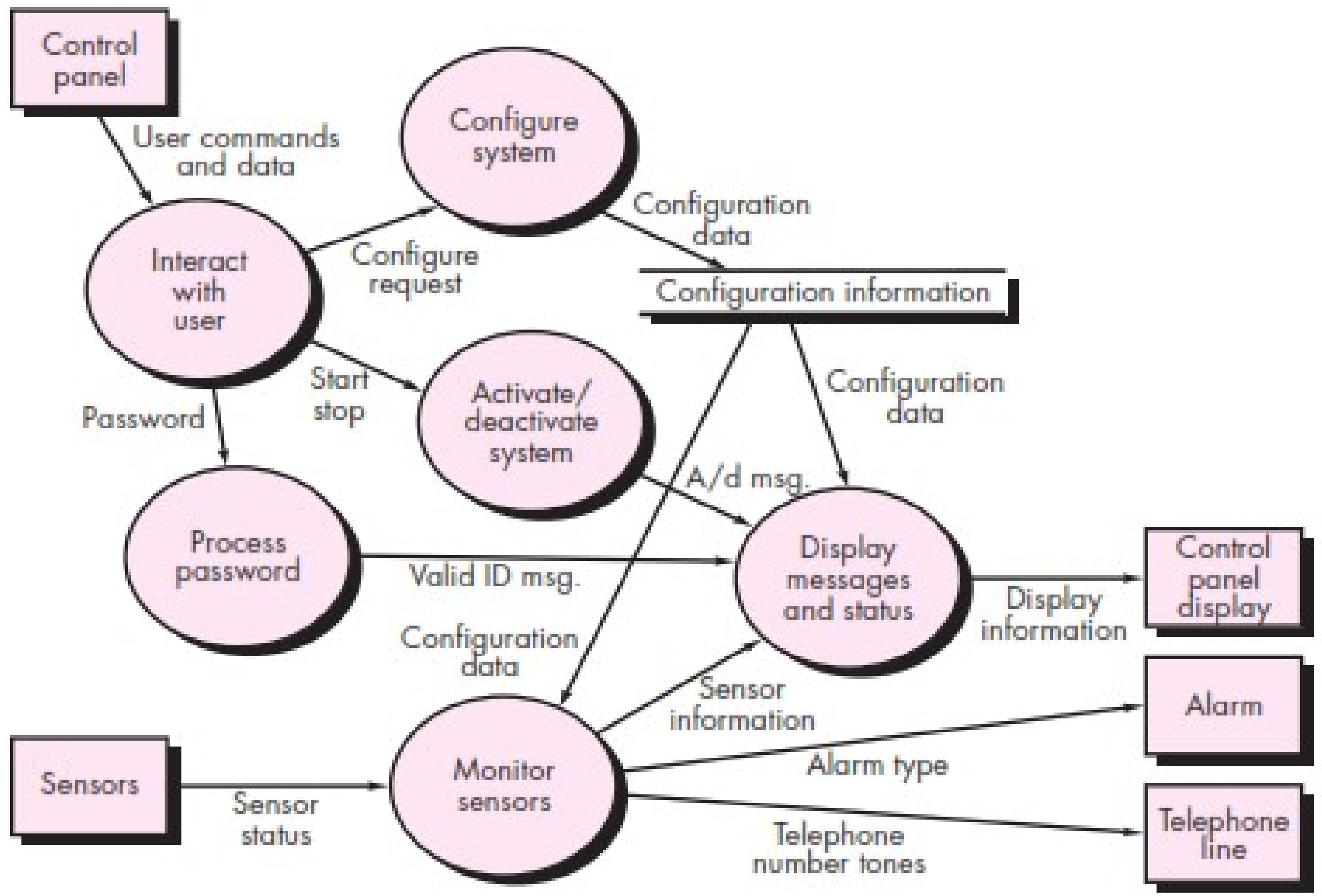
Data Flow Model: Data Flow Diagram (DFD)

- It is a graphical representation of the "flow" of data through an information system, modelling its process aspects
- It is often used as a preliminary step to create an overview of the system, which can later be elaborated

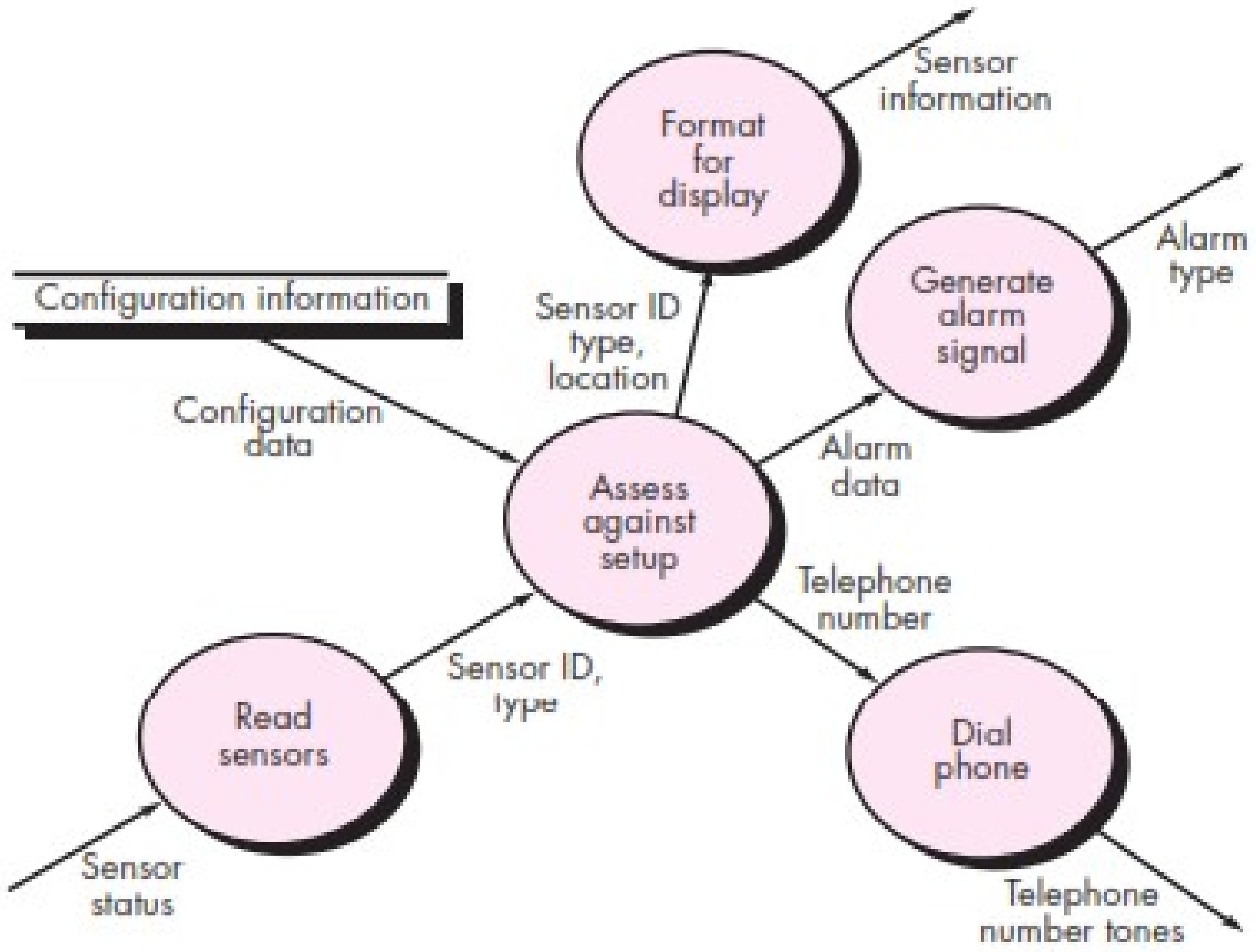


Level 0: Context-level DFD for the SafeHome security function

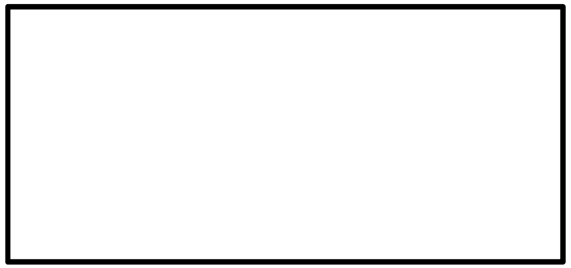
Data Flow Model: DFD Level 1

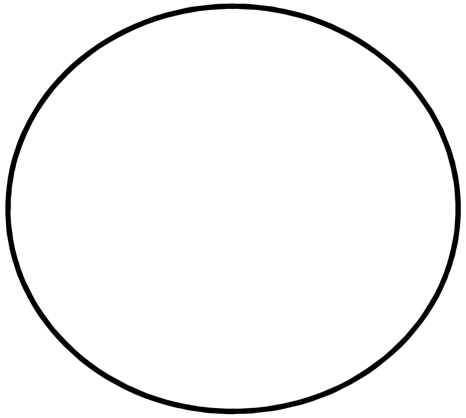


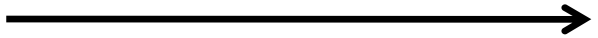
Data Flow Model: DFD Level 2 (Monitor Sensor)

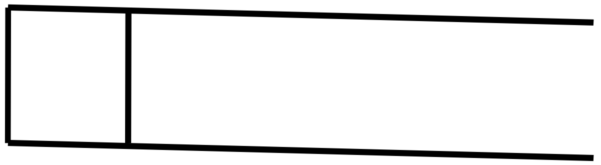


Data Flow Notations


Input / Output


Process/ Function


Flow


Data Store

Requirements Engineering Tasks

4 Negotiation

- Reconcile conflicts
- Agree on a deliverable system that is realistic for developers and customers

Negotiation is not a contest or a game. It works best when both parties win. There are many instances in which you and other stakeholders must negotiate functions and features, identified problems, Risk, priorities, and delivery dates. If the team has collaborated well, all parties have a common goal. Still, negotiation will demand compromise from all parties. Output of Negotiation process is consensuses.

Art of negotiation

Recognise that it is not competition
Map out a strategy
Listen actively
Focus of other party's concern
Don't get it personal
Be creative
Be ready to commit