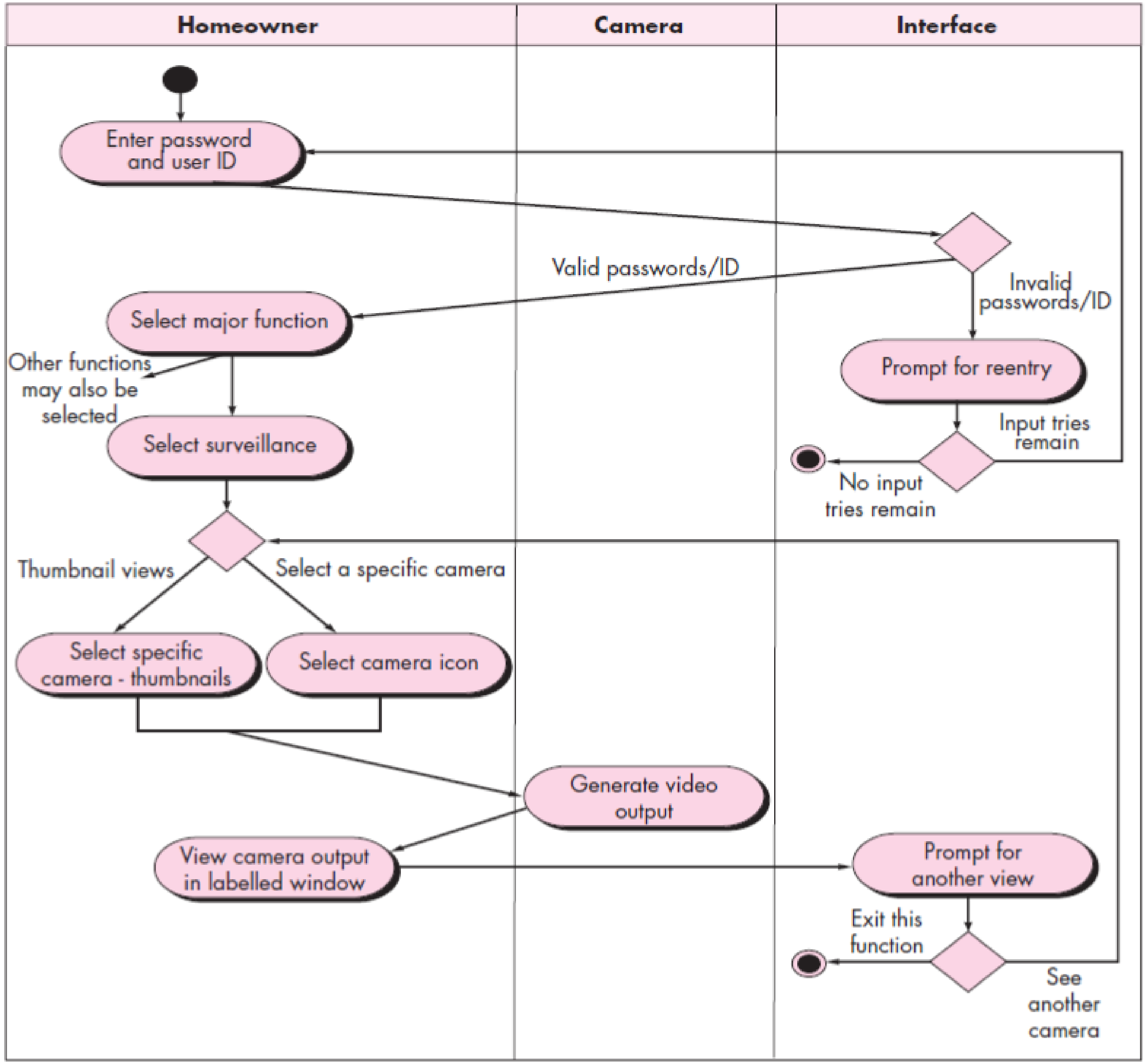
Swimlane Diagrams

- The UML swimlane diagram is a useful variation of the activity diagram and allows you to represent the flow of activities described by the use case and at the same time indicate which actor (if there are multiple actors involved in a specific use case) or analysis class has responsibility for the action described by an activity rectangle.
- Responsibilities are represented as parallel segments that divide the diagram vertically, like the lanes in a swimming pool.
- Three analysis classes—Homeowner, Camera, and Interface—have direct or indirect responsibilities in the context of the activity diagram represented in below Figure.
- The activity diagram is rearranged so that activities associated with a particular analysis class fall inside the swimlane for that class.
- For example, the Interface class represents the user interface as seen by the homeowner.
- The activity diagram notes two prompts that are the responsibility of the interface—“prompt for reentry” and “prompt for another view.”
- These prompts and the decisions associated with them fall within the Interface swimlane.
- However, arrows lead from that swimlane back to the Homeowner swimlane, where homeowner actions occur.

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



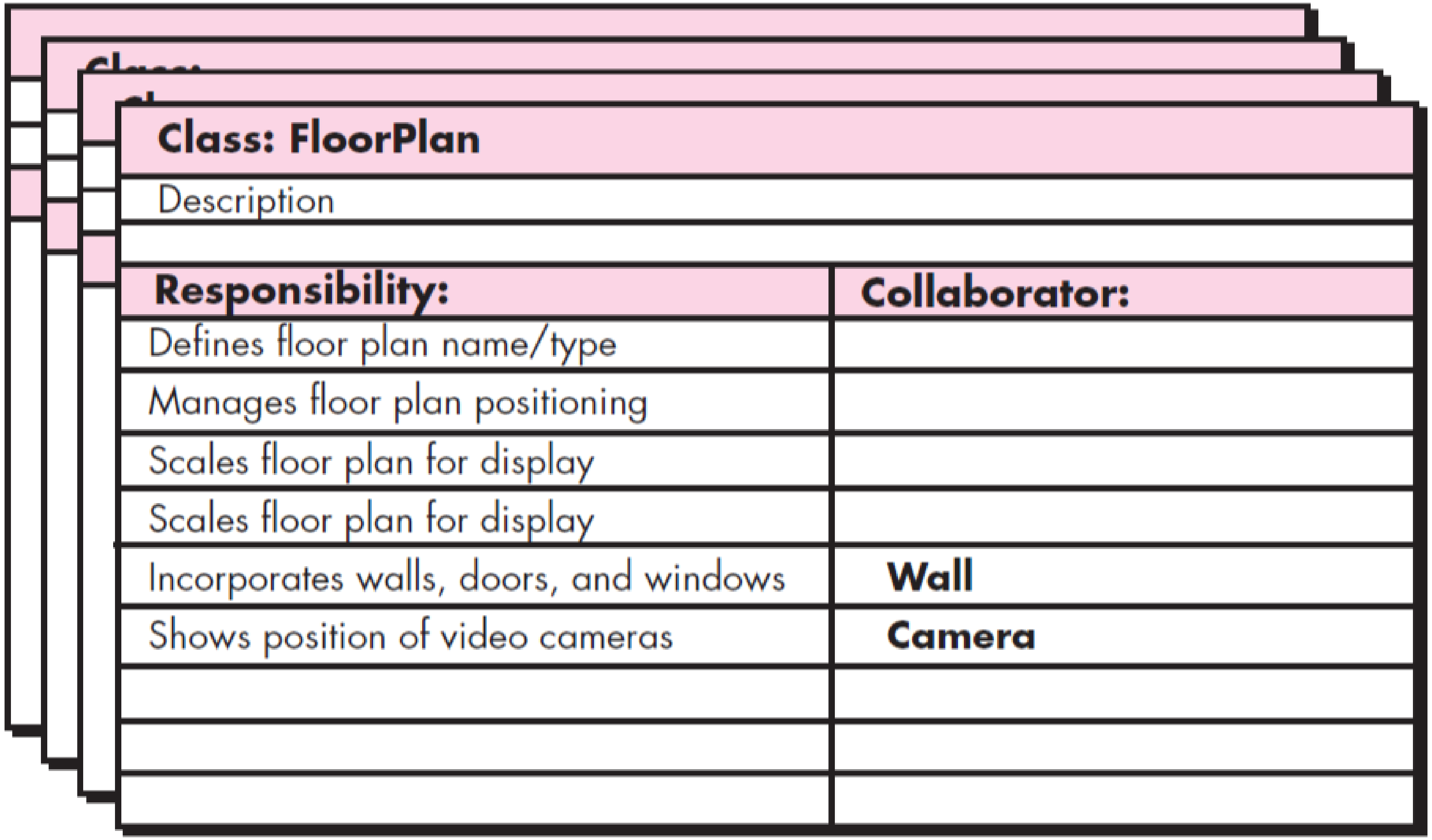
Class-Responsibility-Collaborator (CRC) Modeling

- Class-responsibility-collaborator (CRC) modeling [Wir90] provides a simple means for identifying and organizing the classes that are relevant to system or product requirements.
- Ambler describes CRC modeling in the following way:

➢ A CRC model is really a collection of standard index cards that represent classes. The cards are divided into three sections. Along the top of the card you write the name of the class. In the body of the card you list the class responsibilities on the left and the collaborators on the right.

- In reality, the CRC model may make use of actual or virtual index cards.
- The intent is to develop an organized representation of classes.
- Responsibilities are the attributes and operations that are relevant for the class.
- Stated simply, a responsibility is “anything the class knows or does”.
- Collaborators are those classes that are required to provide a class with the information needed to complete a responsibility.
- In general, a collaboration implies either a request for information or a request for some action.
- A simple CRC index card for the FloorPlan class is illustrated in below Figure.
- The list of responsibilities shown on the CRC card is preliminary and subject to additions or modification.
- The classes Wall and Camera are noted next to the responsibility that will require their collaboration.

A CRC model index card



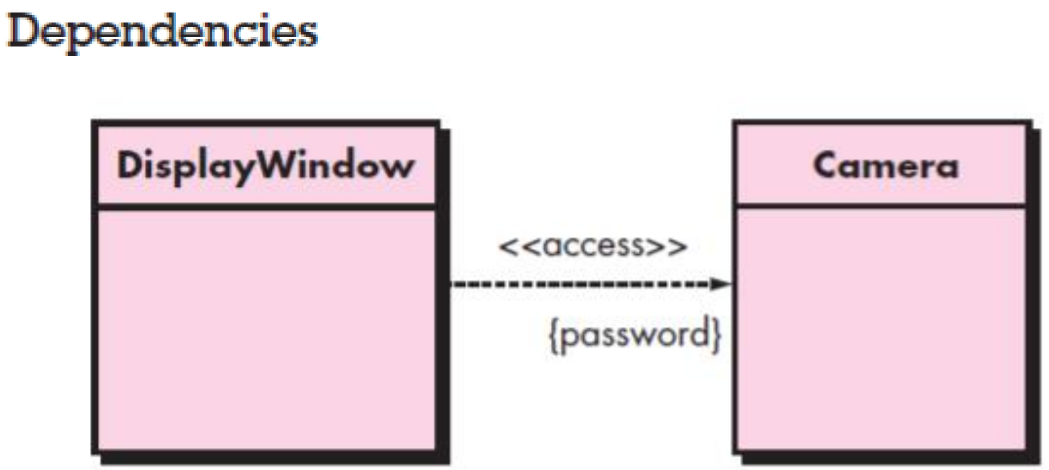
Class: FloorPlan

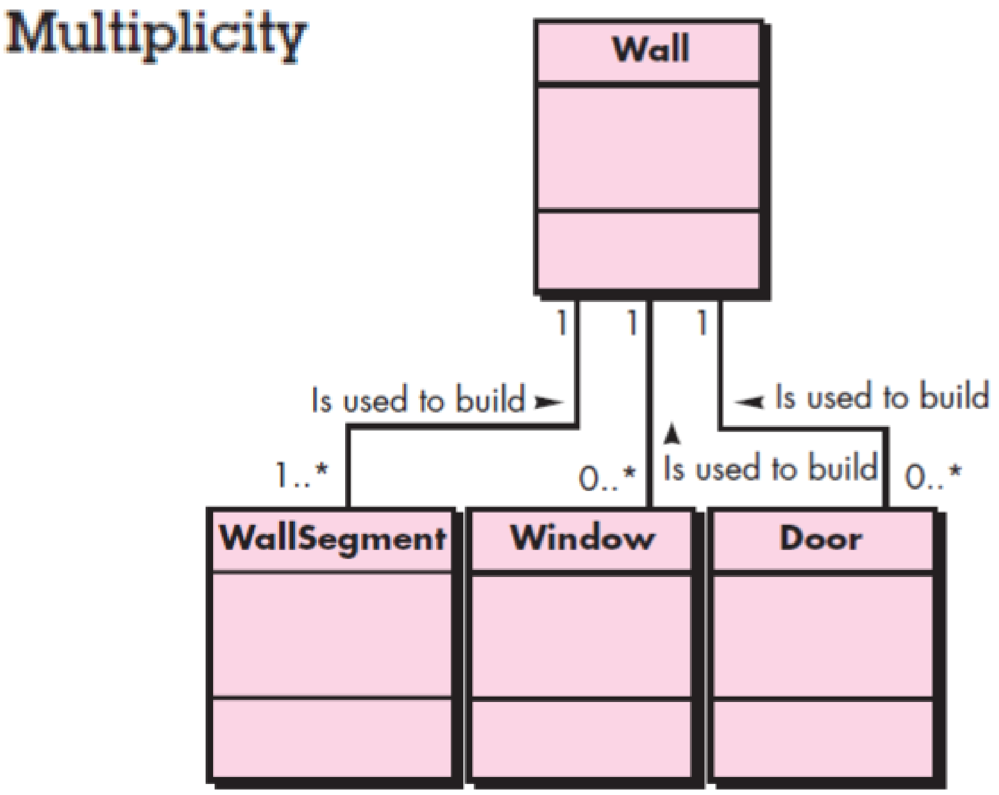
Description

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

- As an example, consider the SafeHome security function.
- As part of the activation procedure, the ControlPanel object must determine whether any sensors are open. A responsibility named determine-sensor-status() is defined. If sensors are open, ControlPanel must set a status attribute to "not ready." Sensor information can be acquired from each Sensor object. Therefore, the responsibility determinesensor- status() can be fulfilled only if ControlPanel works in collaboration with Sensor.

- Associations and Dependencies:





Data Flow Model

- The data flow diagram enables you to develop models of the information domain and functional domain.
- As the DFD is refined into greater levels of detail, you perform an implicit functional decomposition of the system.
- At the same time, the DFD refinement results in a corresponding refinement of data as it moves through the processes that embody the application.
- A few simple guidelines can aid immeasurably during the derivation of a data flow diagram:

1) The level 0 data flow diagram should depict the software/system as a single bubble.
2) Primary input and output should be carefully noted;
3) Refinement should begin by isolating candidate processes, data objects, and data stores to be represented at the next level;
4) All arrows and bubbles should be labeled with meaningful names;
5) Information flow continuity must be maintained from level to level,2 and
6) One bubble at a time should be refined.

- To illustrate the use of the DFD and related notation, we again consider the SafeHome security function.
- A level 0 DFD for the security function is shown in below Figure 1.
- The primary external entities (boxes) produce information for use by the system and consume information generated by the system.
- The labeled arrows represent data objects or data object hierarchies.
- For example, user commands and data encompasses all configuration commands, all activation/deactivation commands, all miscellaneous interactions, and all data that are entered to qualify or expand a command.

Context-level DFD for the *SafeHome* security function

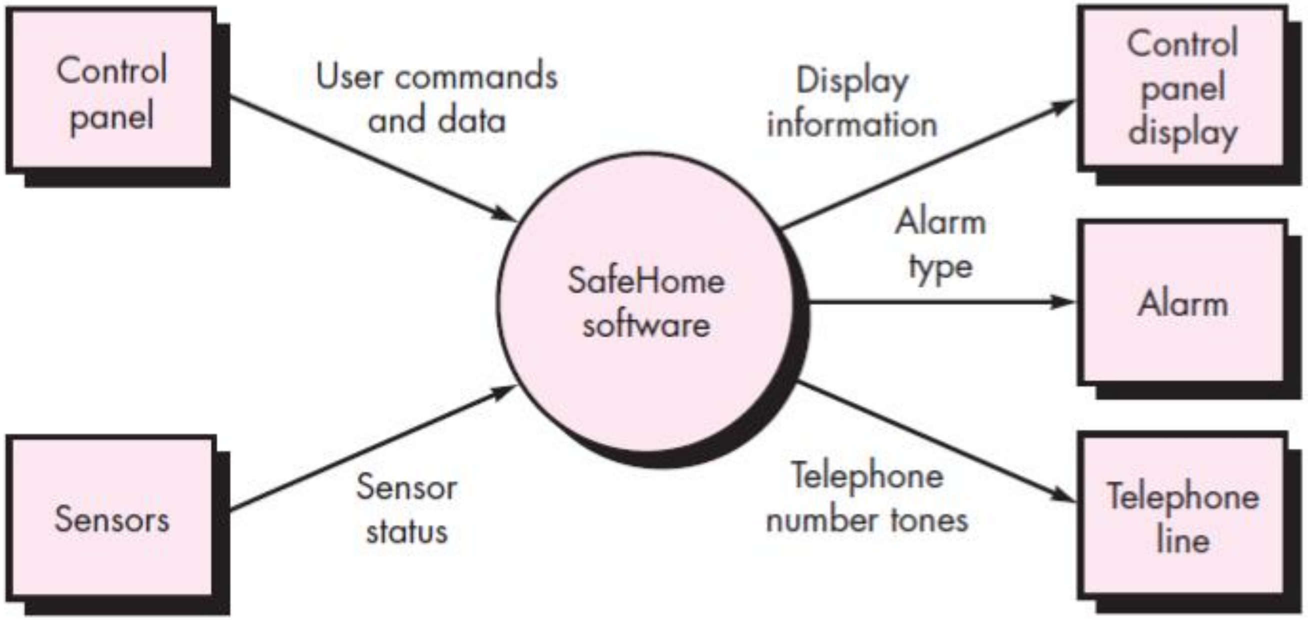
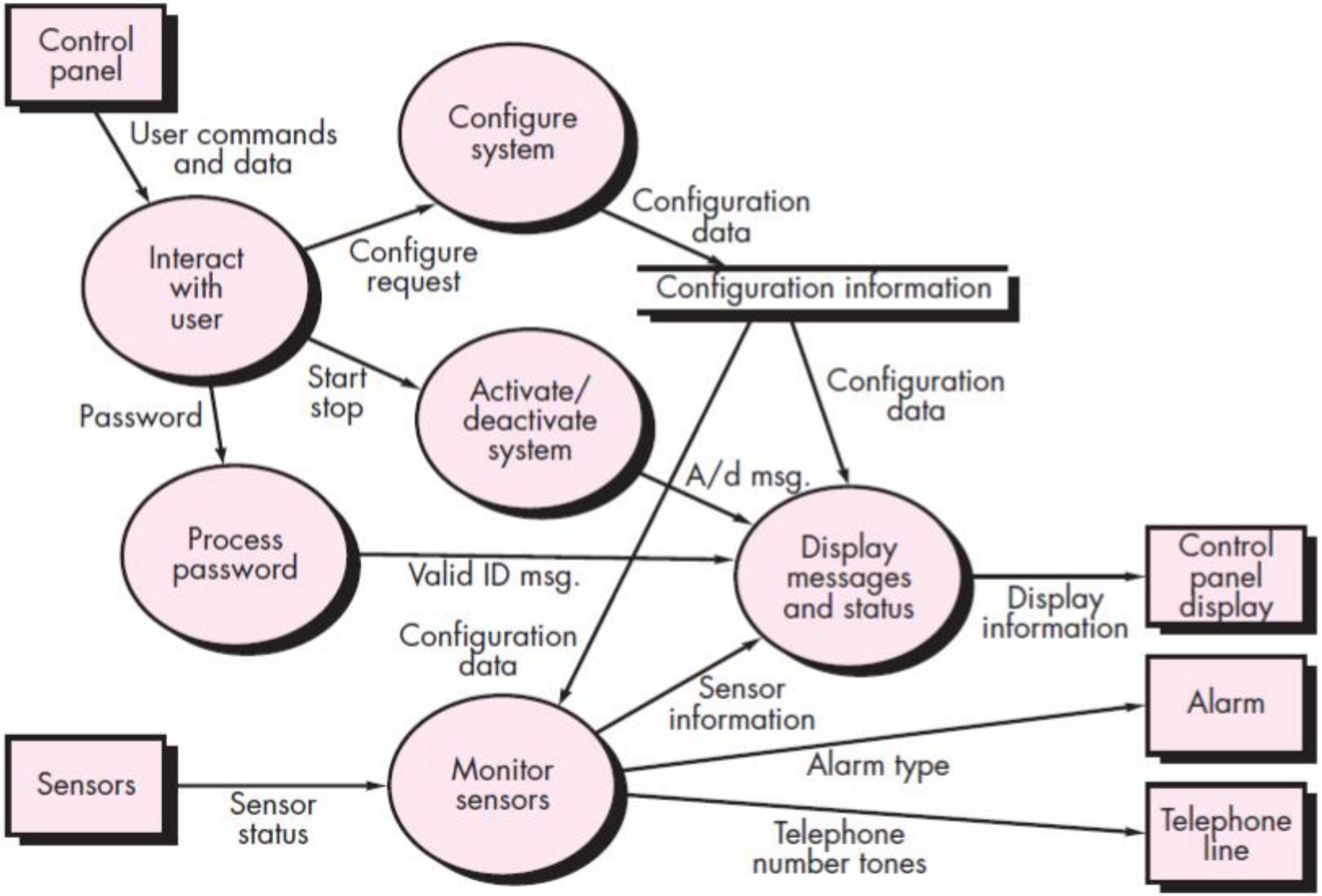


Fig.1 Level 0

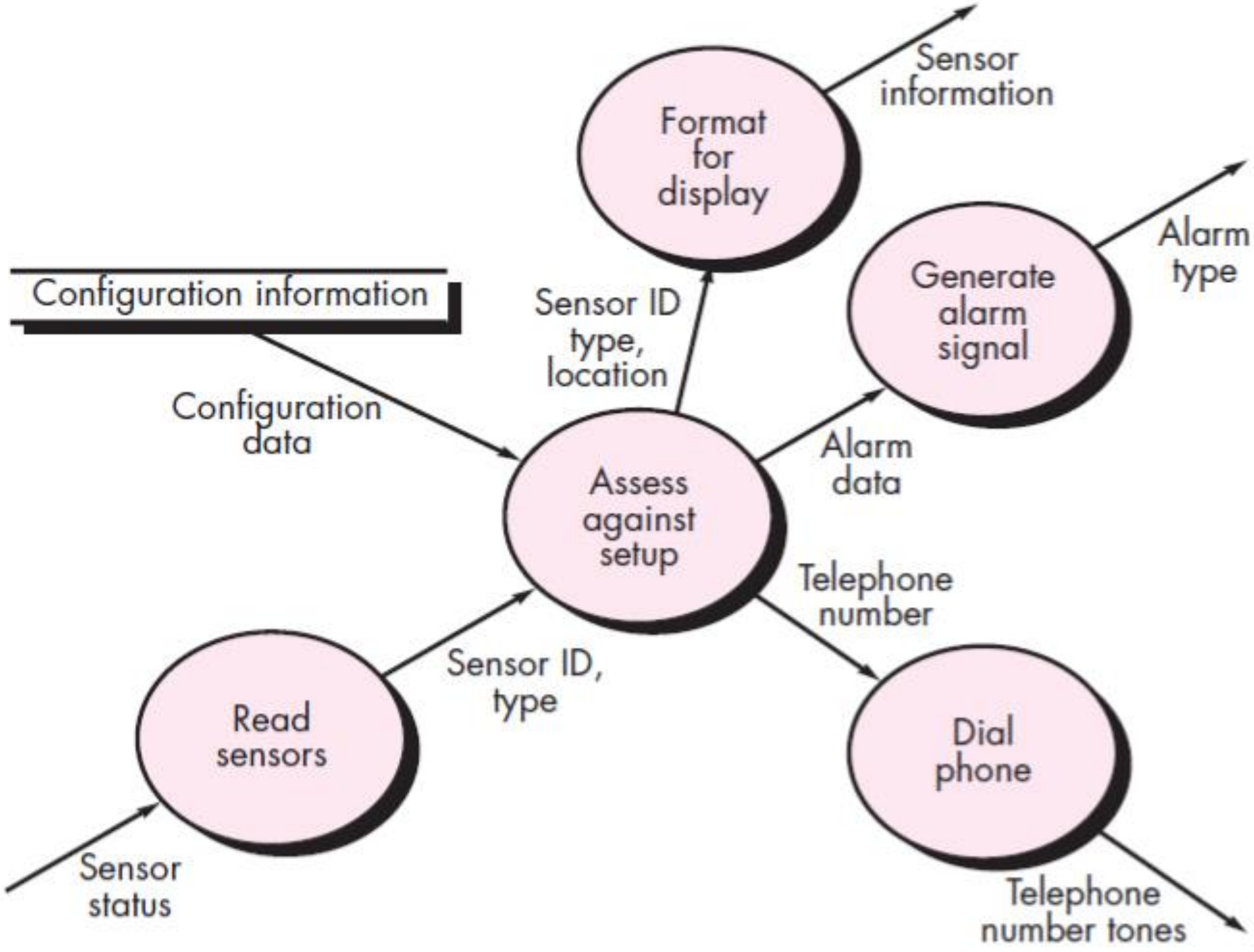
- The level 0 DFD must now be expanded into a level 1 data flow model.

Level 1 DFD for *SafeHome* security function



- The processes represented at DFD level 1 can be further refined into lower levels.
- For example, the process monitor sensors can be refined into a level 2 DFD as shown in Figure below.

Level 2 DFD that refines the *monitor sensors* process


Sensor information
Format for display
Alarm type
Generate alarm signal
Configuration information
Sensor ID type, location
Configuration data
Alarm data
Assess against setup
Telephone number
Sensor ID, type
Read sensors
Dial phone
Sensor status
Telephone number tones