ANALYSIS RULES OF THUMB

- The model should focus on the requirements that are visible within the problem or business domain
- High level of abstraction
- Each element of an analysis model should add to an overall understanding of software requirements and provide insight into the information domain, functionand behavior of the system
- Delay consideration of infrastructure and other non-functional models until design phase
- Minimize coupling throughout the system - interconnectedness between classes and functions should be low
- Keep the model as simple as it can be

ANALYSIS MODELING APPROACHES

- **Structured Analysis :** This considers data and processes that transforms the data as separate entities.
- Data objects are modeled in a way that defines their attributes and relationships.
- Processes that manipulate data objects are modeled in a manner that shows how they transform data as data objects flow through the system. Flow oriented modeling is predominantly used.

- **Object Oriented Analysis :** This approach focuses on the definition of classes and the manner in which they collaborate with one another to effect customer requirements.
- UML and Unified process are predominantly object oriented

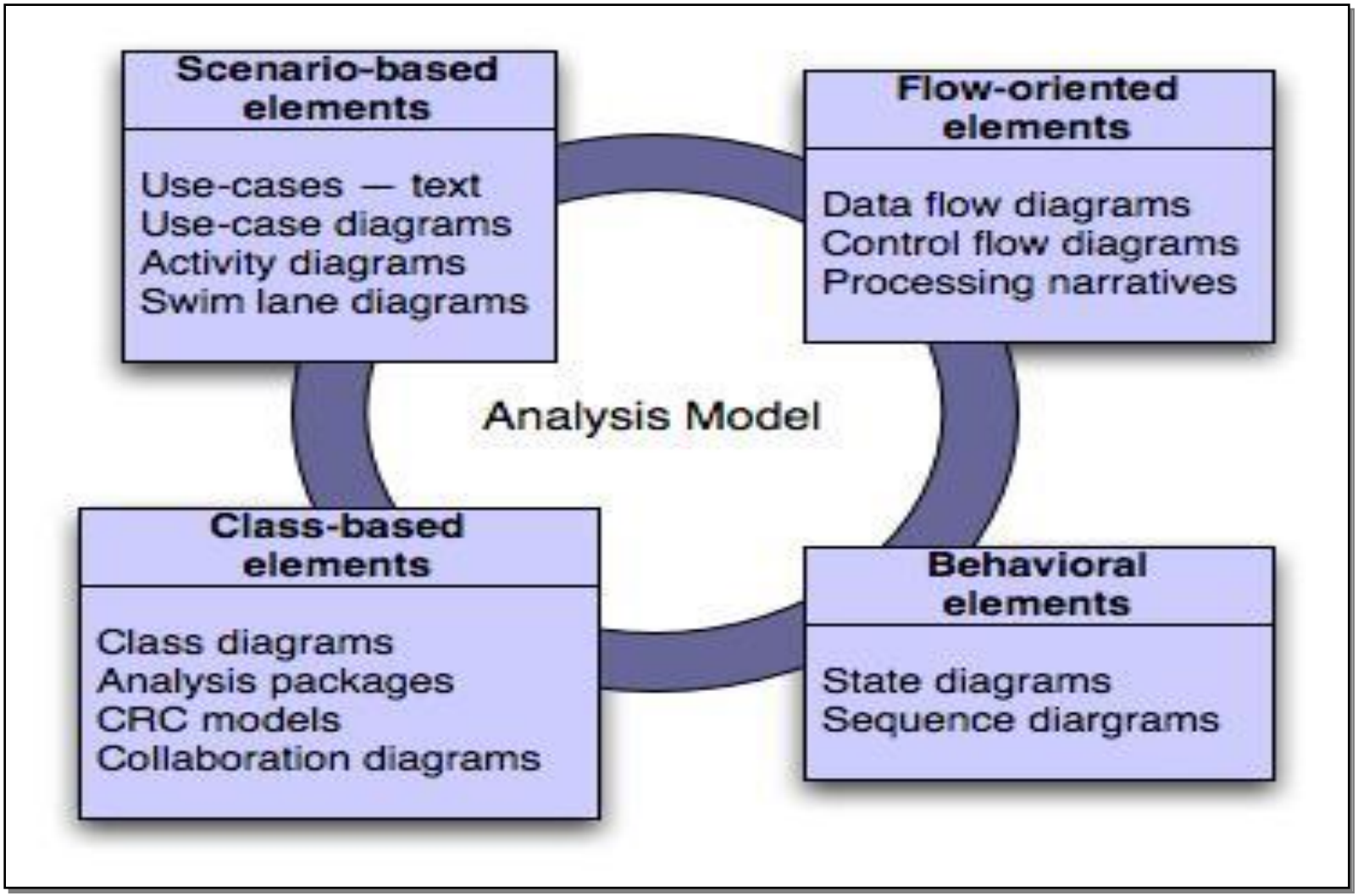
Analysis Model

- Analysis Modeling leads to the derivation of each of the modeling elements shown in the following figure

- Only those modeling elements that add value to the model should be be used

- The specific content of each element may differ from project to project

Analysis Model



Elements of the analysis model

Scenario-Based Modeling

- For the building of analysis and design models, it is essential for software engineers **to understand how end users and other actors want to interact with the system**
- Analysis Modeling with UML begins with the creation of scenarios in the form of use-cases, activity diagrams and swim-lane diagrams
-

Grammatical Parse

- 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 a **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…

Use cases

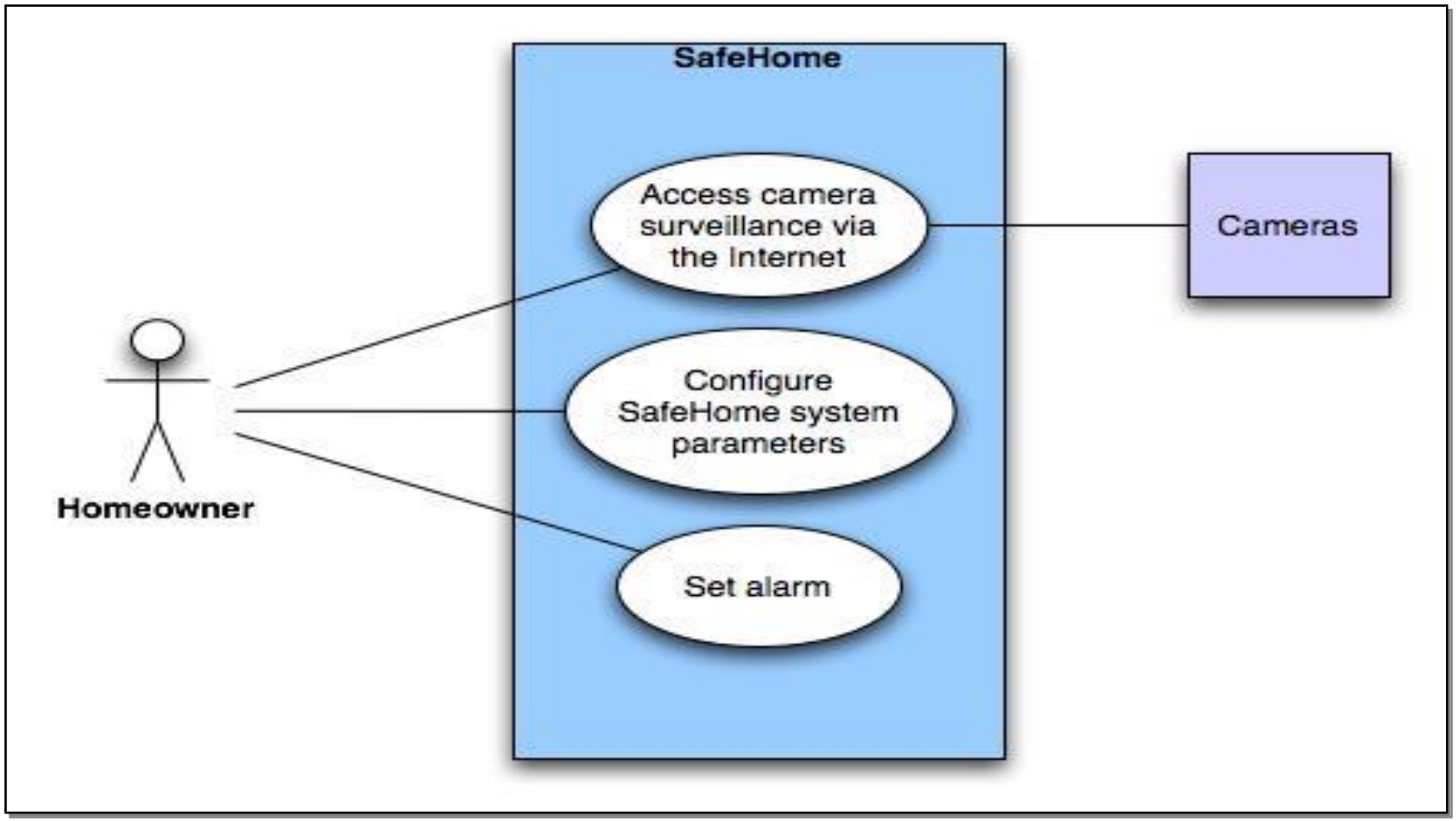
- Use cases can be represented as a text narrative or as an ordered sequence of user actions or by using a template
- Example, consider the use case Access Camera Surveillance – Display Camera Views (ACS – DCV) of the Safe Home Security System

Scenario-Based Modeling

- **Use case : (ACS – DCV) – User actions**

1. The homeowner logs on to the safe home products web-site.
2. The homeowner enters his / her user-id
3. The homeowner enters two passwords
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” [or “all camera”] button
7. The system displays the floor plan [or Thumbnail snapshots of all cameras”]
8. The homeowner selects the camera icon
9. The homeowner selects the “view” button
10. The system displays a viewing window that is identified by camera-id

Scenario-Based Modeling - Use-case Diagram



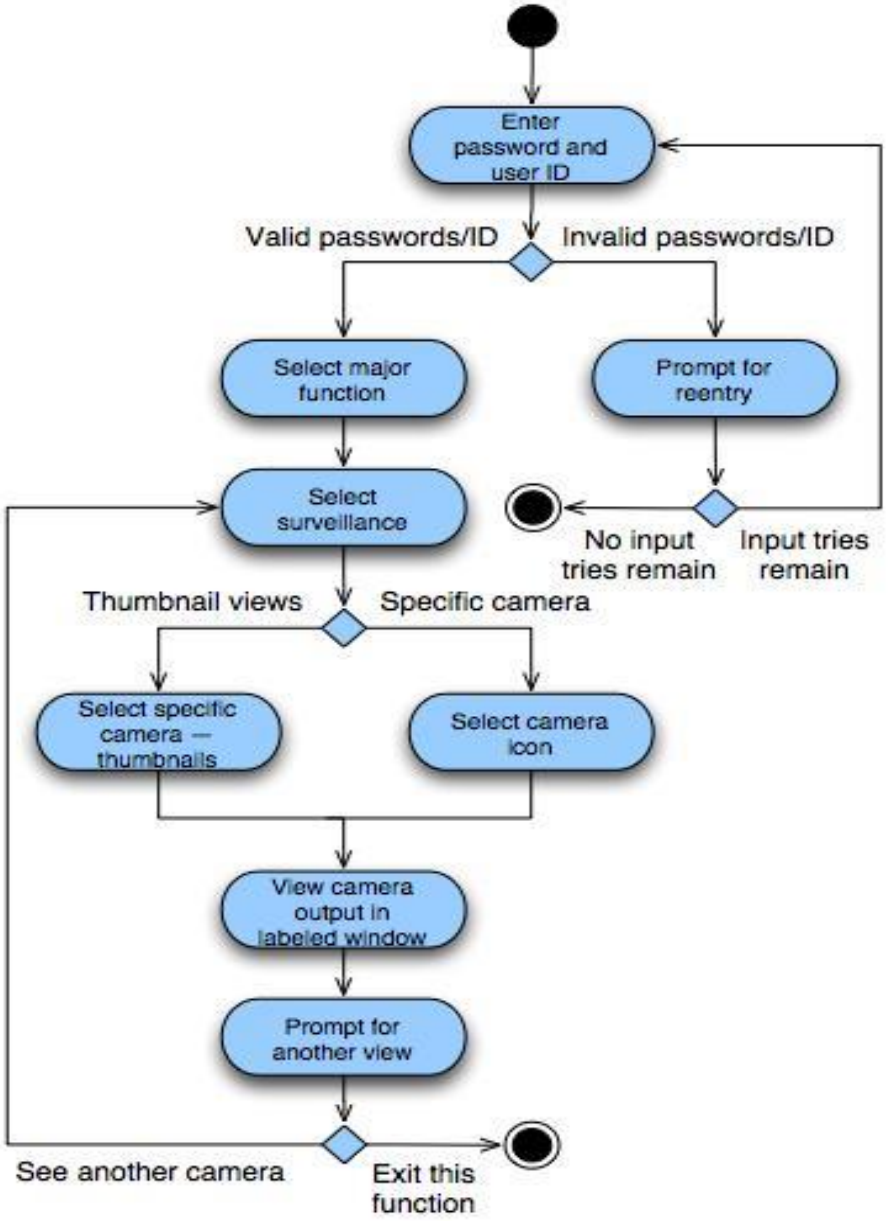
Use-case diagram for surveillance function

Alternative Actions

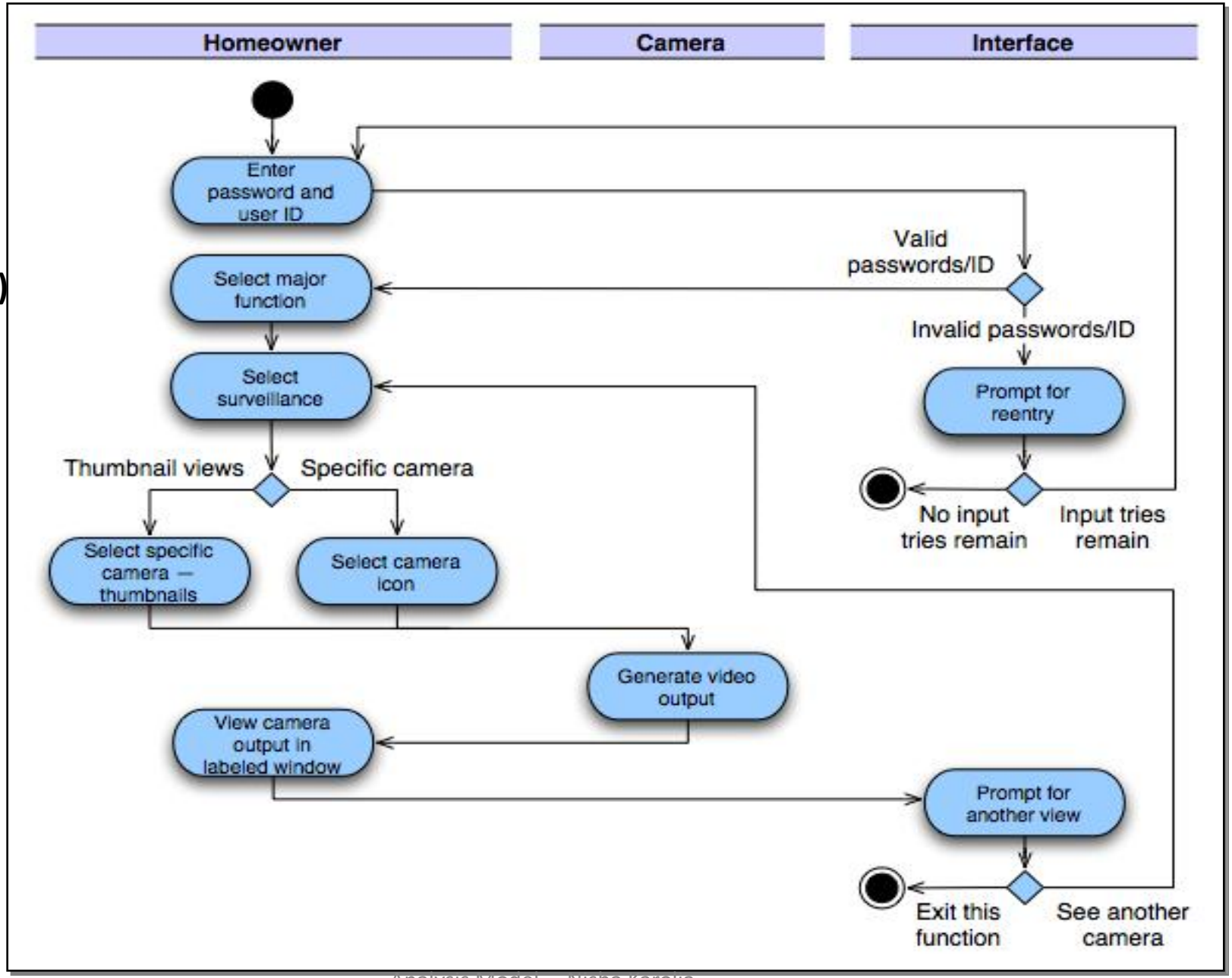
- Can the actor take some other action at this point?
- Is it possible that the actor will encounter some error condition at this point? - Exceptions
- Is it possible that the actor will encounter behavior invoked by some event outside the actor's control?

Activity diagram for

Access Camera Surveillance—Display Camera Views function



Swimlane diagram (ACS-DCV)



Flow-Oriented Modeling

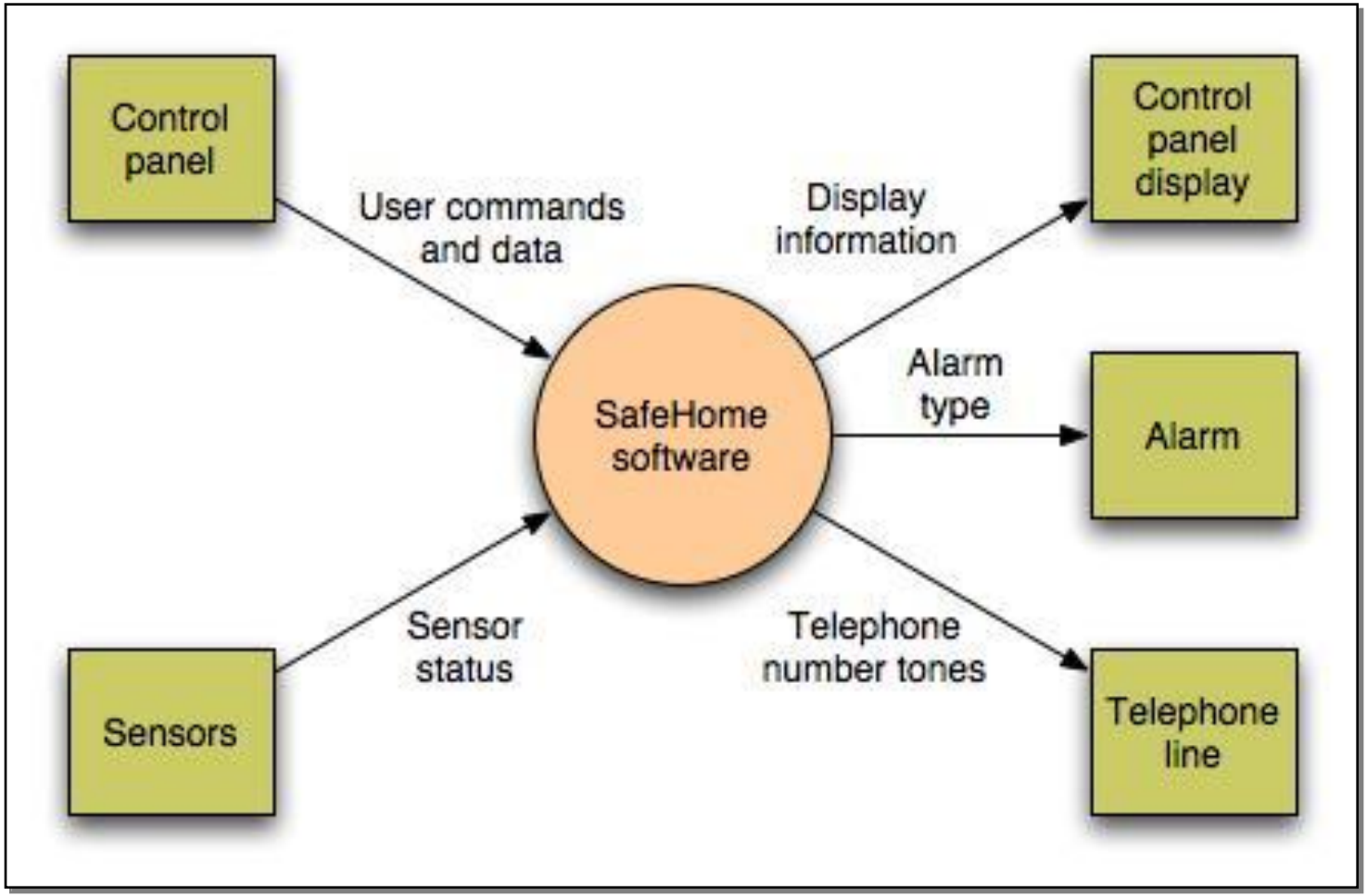
- **Data Flow Diagram**
- DFD takes "input-process-output" view of a system i.e. Data Objects flow into the software & they are transformed by processing elements & resultant data objects flow out of the software
- Data objects are represented by labeled arrows .
- Transformations are represented by bubbles (circles).
- Primary external entities are represented by boxes, they either produce information for use by system or consume information generated by the system.

Flow-Oriented Modeling

Guidelines :

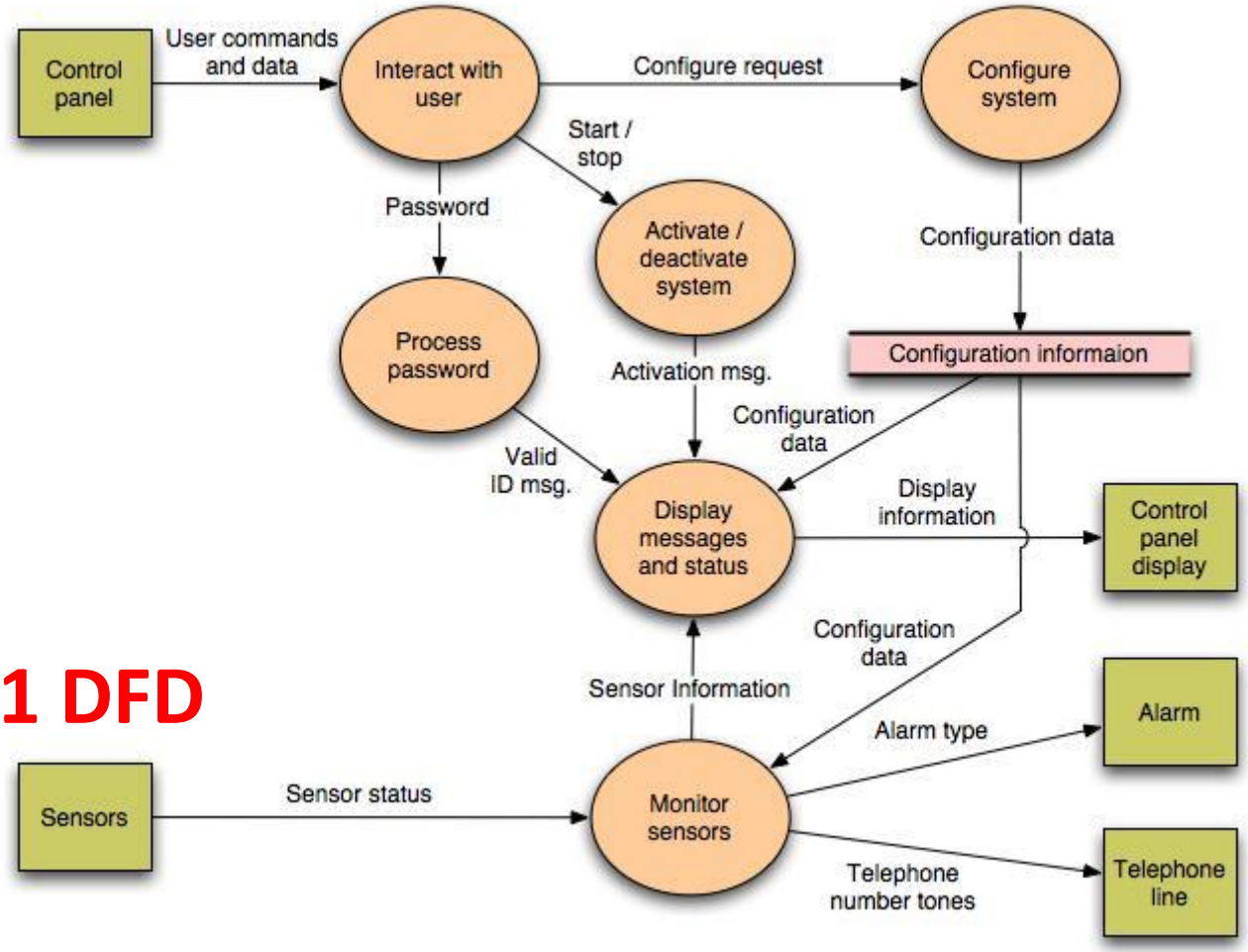
- Depict the system as single bubble in level 0.
- Carefully note primary input and output.
- Refine by isolating candidate processes and their associated data objects and data stores.
- Label all elements with meaningful names.
- Maintain information conformity between levels.
- Refine one bubble at a time.

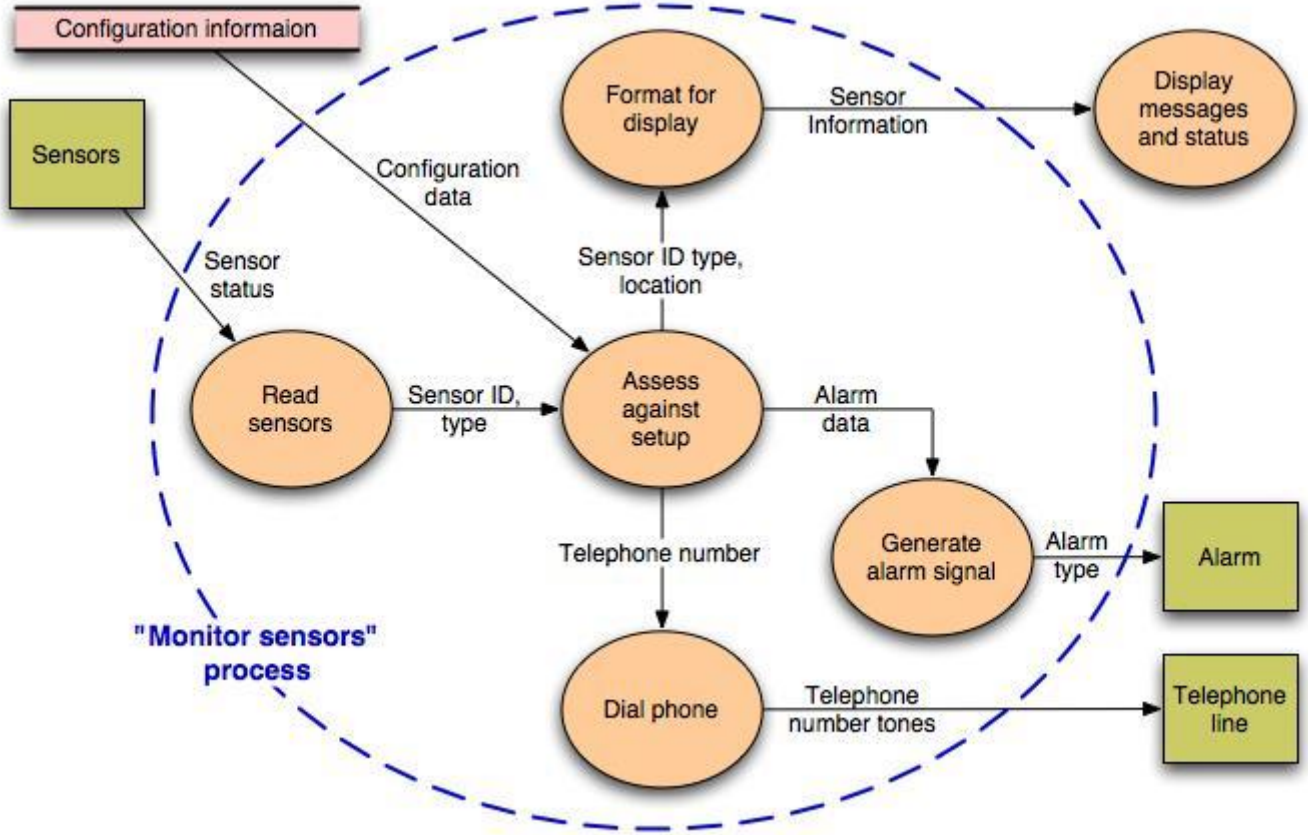
Data Flow Diagram



Context-level DFD for *SafeHome* security function

Level 1 DFD


Control panel
User commands and data
Interact with user
Configure request
Configure system
Start / stop
Password
Activate / deactivate system
Configuration data
Process password
Activation msg.
Configuration informaion
Configuration data
Valid ID msg.
Display messages and status
Display information
Control panel display
Configuration data
Sensor Information
Alarm type
Alarm
Sensors
Sensor status
Monitor sensors
Telephone number tones
Telephone line



Level 2 DFD that refines the monitor sensors process

Flow-Oriented Modeling

Creation of Control Flow Model – Need :

- For many types of applications, data model & DFD are sufficient to obtain meaningful insight into Software Requirements
- However large class of applications are driven by “events” rather than data, produce control information rather than displays or reports and process information with heavy concern for time & performance Such applications need Control Flow Modeling in addition to data flow modeling
- Event or control item can be implemented as a boolean value (0 or 1 OR True or False)

Flow-Oriented Modeling

Creation of Control Flow Model – Guidelines :

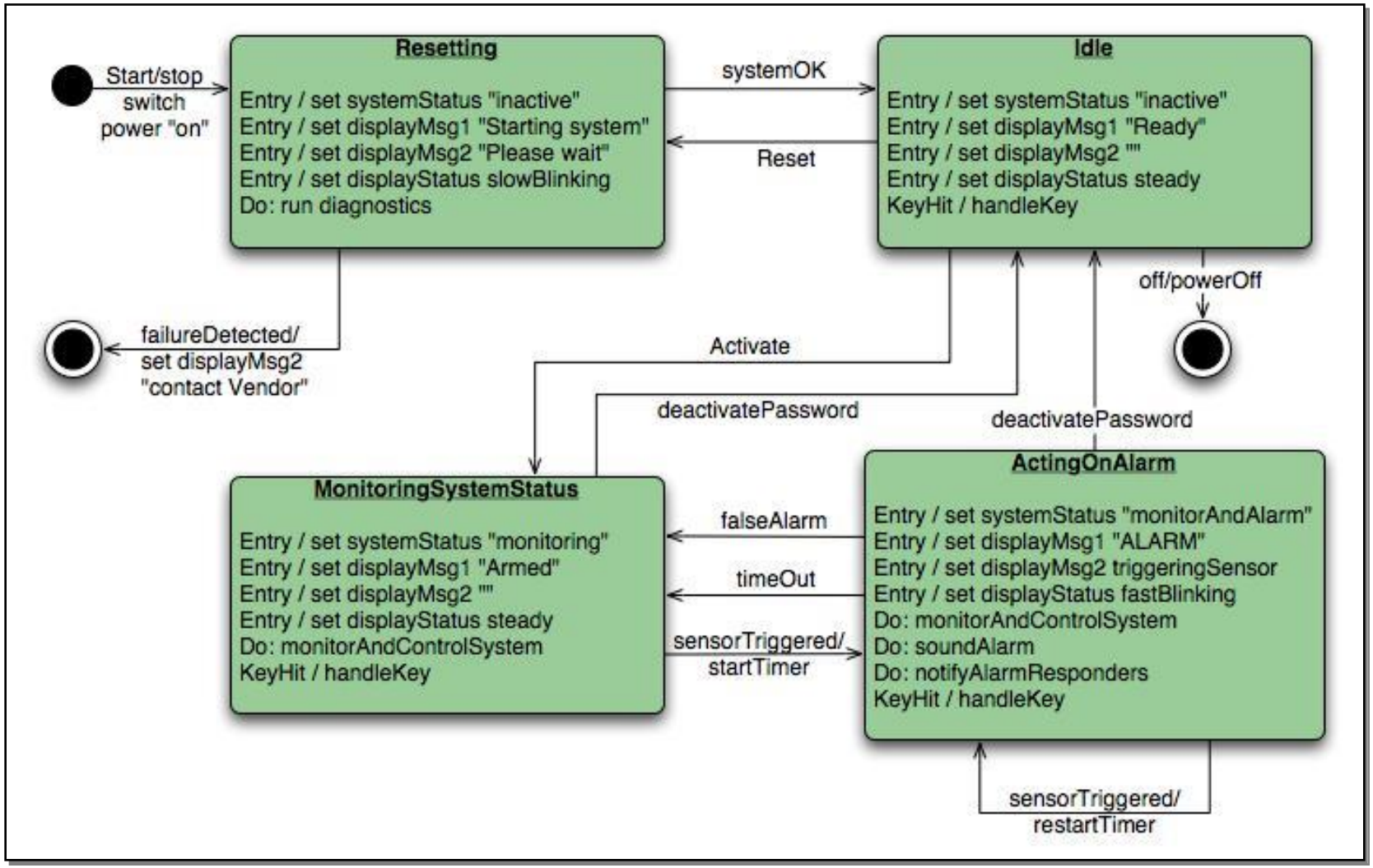
- List all sensors that are “read” by the software
- List all “interrupts conditions”
- List all “switches” that are actuated by an operator
- List all “data conditions” (empty, full, jammed)
- Review all control items w.r.t. noun/verb parse of processing narrative. They are possible I/O for control flow
- Describe the behavior of the system by identifying it’s states : identify how each stage is reached & define the transitions between states
- Focus on omissions – alternate way to reach or exit

Flow-Oriented Modeling

Control Specifications [CSPEC] :

- CSPEC represents the behavior of the system.
- It contains a state diagram that is sequential specification of behavior of the system. It may also contains Program Activation Table i.e. a combinatorial specification of behavior.

Control Flow Diagram



Flow-Oriented Modeling

Process Specifications [PSPEC] :

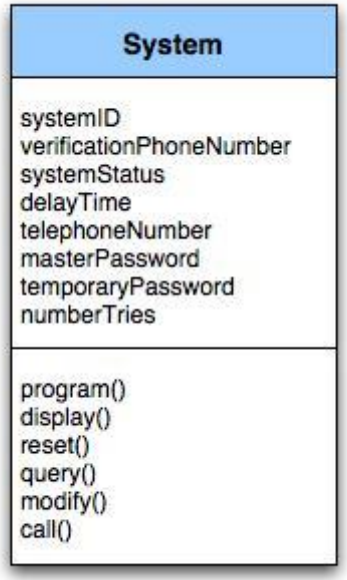
- PSPEC is used to describe all processes of DFD that appear at the final level of refinement. It contains narrative text and program design language (PDL) description of the process algorithm, mathematical equations, tables, diagrams or charts.
- By providing a PSPEC for each bubble in the data flow model a software engineer creates a “mini-spec” that can serve as a guide for design of the software component that will implement the process. For example, PSPEC for process password

Class-Based Modeling

Identifying Potential Analysis Classes :

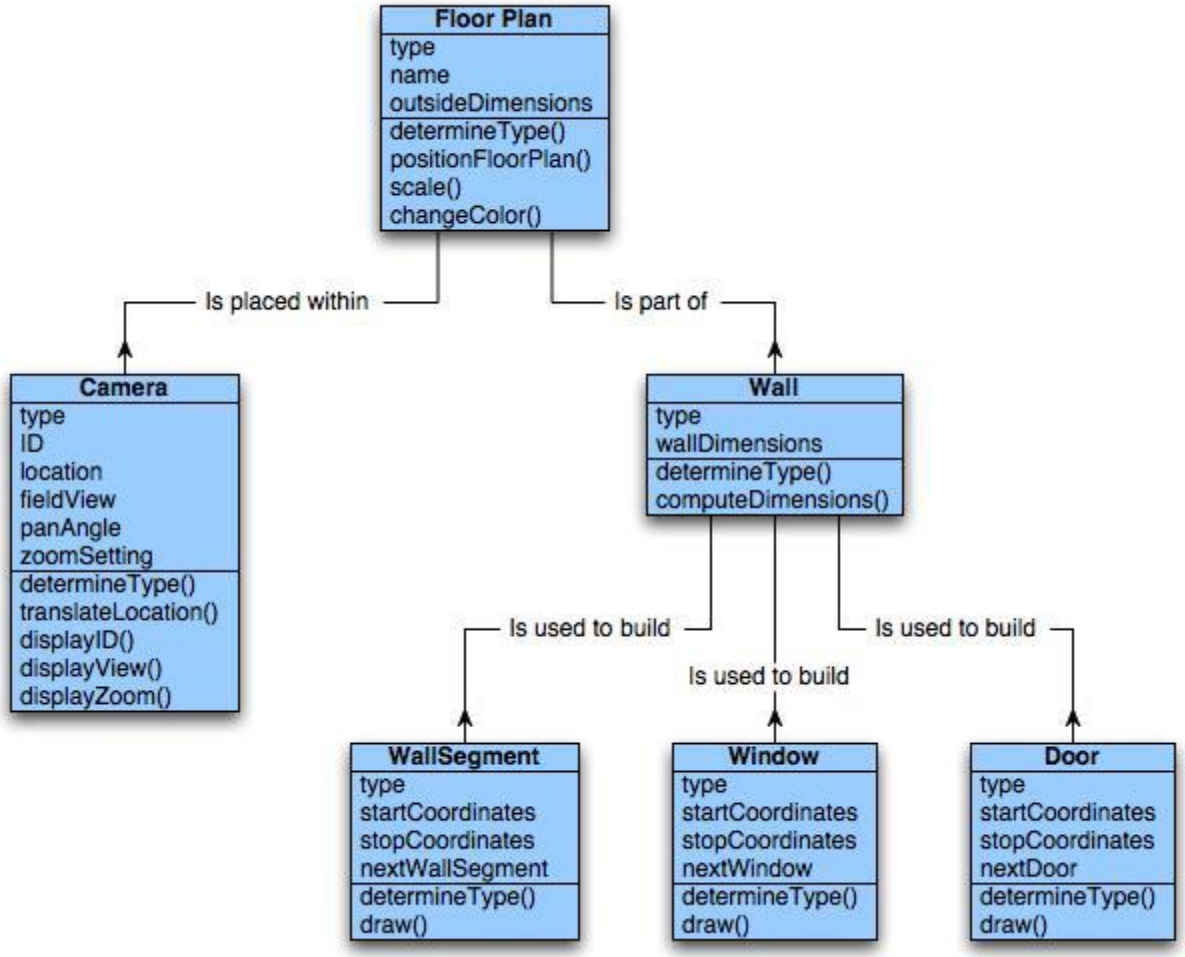
- External entities that produce or consume information
- Things that are part of the information domain
- Occurrences or events
- Roles played by people who interact with the system
- Organizational units
- Places that establish context
- Structures that define a class of objects

Class Diagram



Class diagram for the system class

Class Diagram

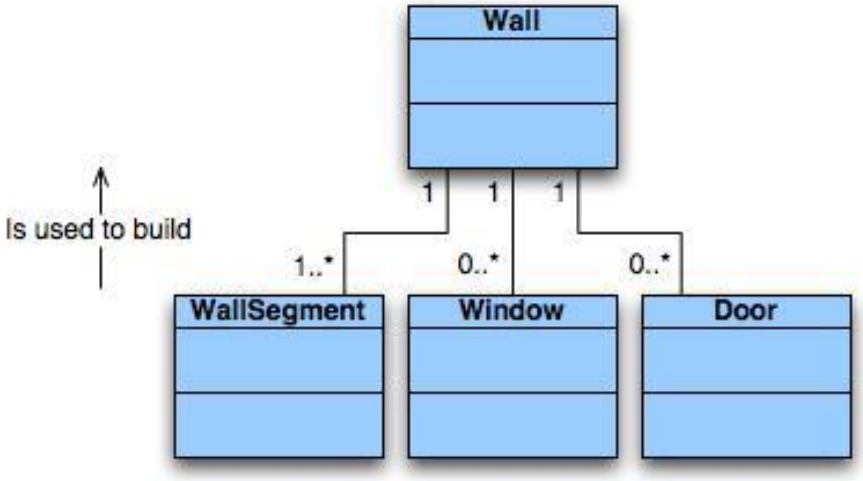


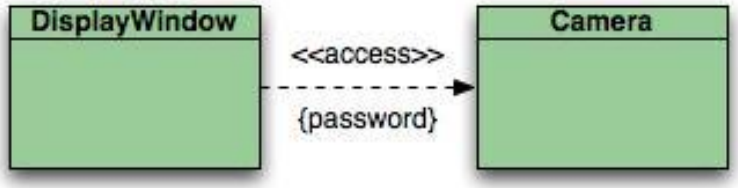
Class diagram for FloorPlan

Class Collaborations

- **Relationships between classes:**
  - is-part-of — used when classes are part of an aggregate class.
  - has-knowledge-of — used when one class must acquire information from another class.
  - depends-on — used in all other cases.

Class Diagrams





Top: Multiplicity

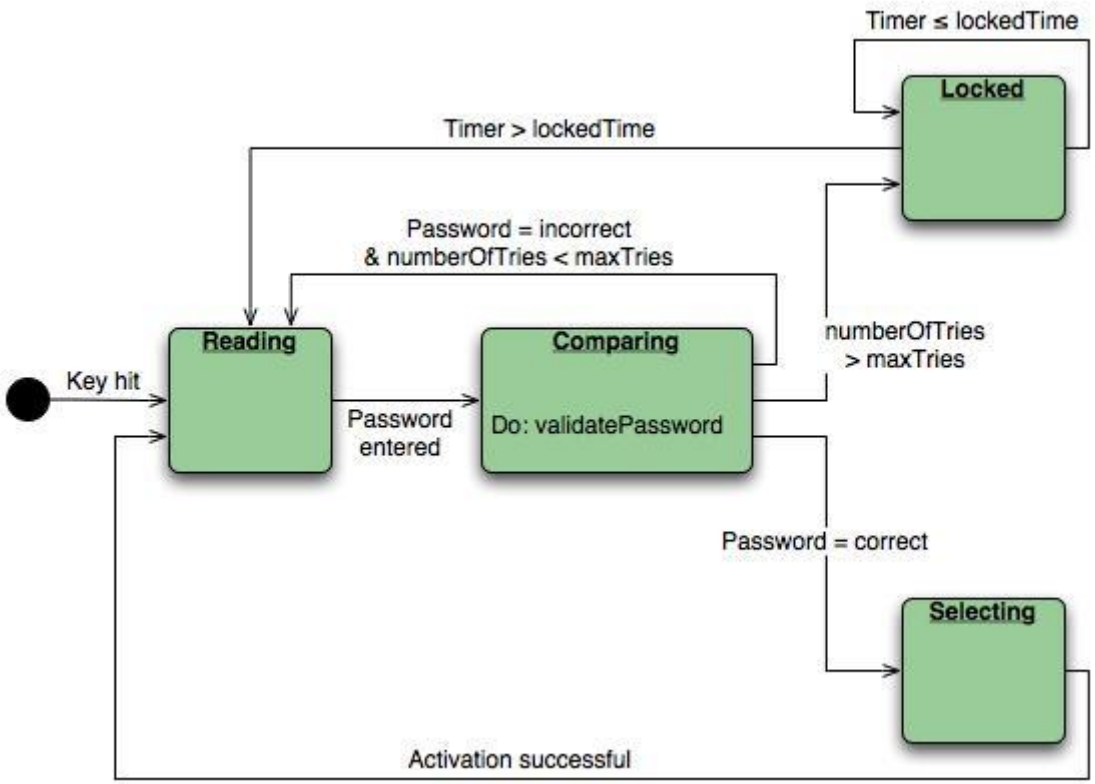
Bottom: Dependencies

Behavioral Modeling

Identifying Events :

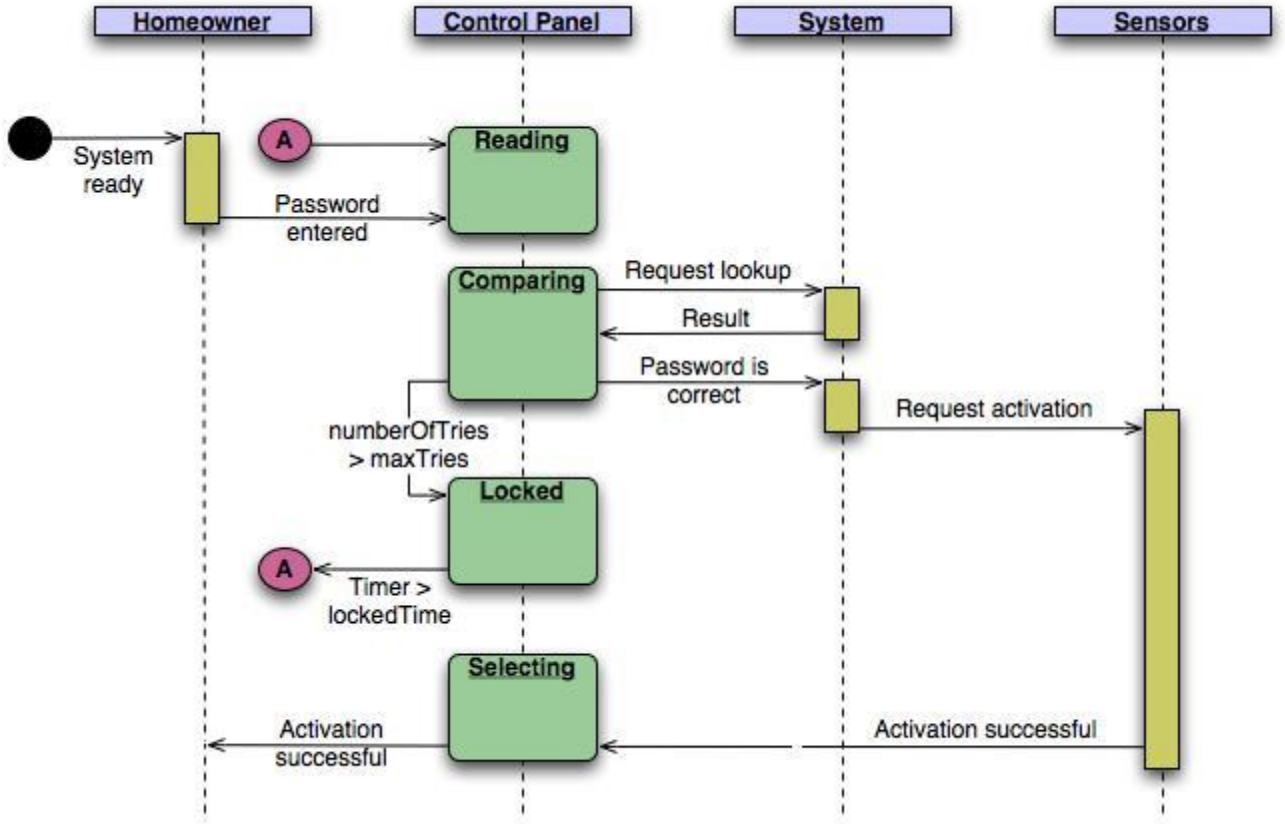
- A use-case is examined for *points of information exchange*.
- The homeowner uses the keypad to key in a four-digit password. The password is compared with the valid password stored in the system. If the password in incorrect, the control panel will beep once and reset itself for additional input. If the password is correct, the control panel awaits further action.

State Diagram



State diagram for the ControlPanel class

Sequence Diagram



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