Software Engineering

Unit II : Requirement Engineering

- **Requirement Engineering :** Initiating the Process, Eliciting Requirements, Building the Requirements Model, Negotiating, Validating Requirements
- **Requirements Analysis :** Scenario Based Analysis
- **Requirements Modeling Strategies :** Flow Oriented Modeling, Class based Modeling
- **SRS :** Software Requirements Specifications

REUIREMENTS ENGINEERING

- What is it ?
- Who Does it?
- Why it is Important?
- It is not easy
- Bridge to Design & Construction
- Communication + Analysis

REQUIREMENT ENGINEERING TASKS

1. Inception : Basic Understanding
2. Elicitation : Clear Understanding
3. Elaboration : Refinement
4. Negotiation : Settlement of Conflicts
5. Specification : SRS
6. Validation : FTR Team Validates
7. Management : Changed Reqt. Mgmt.

INCEPTION

INITIATING THE REQUIREMENT ENGG. PROCESS

How Requirements are originated?

- Business Need is identified
- New market or service is discovered
- Working description of the project

Steps to initiate Requirement Engineering

1. Identifying the stakeholders
2. Recognizing multiple view points
3. Working towards collaboration
4. Asking the first questions

Identifying the stakeholders

Stakeholder :

- Anyone who gets benefited directly or indirectly from the system
- Stakeholders can be Business operations Managers, Product managers, Marketing people, Internal or External Customers, End-users, Consultants, Product Engineers, Software Engineers, Support & Maintenance Engineers and many others .
- At inception, the requirement engineer prepares a list of people who will contribute input as requirements are elicited.

Recognizing multiple view points

- Each stakeholder explores the system requirements from his point of view.
- As information from multiple view point is collected, emerging requirements may be inconsistent or conflicting.
- Categorize the information collected

Working Towards Collaboration

- Customers and other stakeholders should collaborate among themselves and software engineering practitioners for the success of the project.
- requirements are to be sorted out as commonly agreed and inconsistent or conflicting requirements.

Asking the First Questions

- Questions asked at the inception of the project should be “context free” with the following objectives:
  - to establish communication between software practitioners and the customer
  - to identify all the stakeholders
  - to get the better understanding of the problem
  - to help to “break the ice” and initiate the communication

Asking the First Questions – Contd.

First Set helps to identify all stakeholderswho will have interest in the software solution and it identifies measurable benefits of successful implementation.

- Who is behind the request for this work?
- Who will use the solution?
- What will be the economic benefit of a successful solution?
- Is there any other source for the solution you need?

Asking the First Questions – Contd.

Next Set helps software team to gain a better understanding of the problem

- How would you categorize "good output" that will be generated by a successful solution?
- What problems this solution will address?
- Can you show me the business environment in which the solution will be used?
- Will special performance issues or constraints affect the way the solution is approached?

Asking the First Questions – Contd.

Final Set of questions focuses on the effectiveness of the communication abilityitself.

- Are you the right person to answer these questions
- Are your answers official?
- Are my questions relevant to the problem that you have?
- Am I asking too many questions?
- can anyone else provide additional info?
- Should I be asking you anything else?

ELICITING REQUIREMENTS

Elicitation format consists of :

- Collaborative Requirement Gathering
- Quality Function Deployment
- User Scenarios (i.e Use Cases)
- Elicitation Work Products

Collaborative Requirement Gathering

Guidelines:

- Team of stakeholders & developers work together
- Meetings are conducted between them
- Rules for preparation & participation
- Formal Agenda – Topics, date, Time, Venue in advance
- A Facilitator must be appointed
- Definition Mechanism is to be used
- Goal must be to identify the problems/issues

Collaborative Requirement Gathering

Sequence of Events:

1. Product Request (O/P of Inception Phase)
2. A Facilitator is chosen
3. Meeting is called – Product request & Reqt. List given
4. Attendees are asked to identify objects used/created, processes, functions, constraints, performance etc.
5. Meeting day – Justification for the new system
6. Presentation by each participant – Definition mechanism Used
7. Combined list of requirements is prepared by a group
8. Facilitator shortens/lenghens/modifies the list to develop a consensus list identifying objects, services, constraints & performance

Collaborative Requirement Gathering

Sequence of Events:

9. Team is divided in smaller groups to prepare specifications a group of related requirements.
10. Presentation by each team – addition / deletion / further elaborations are done. New objects, services, constraints performances may get added to the original list
11. Issues list is prepared- to be discussed in nest meeting
12. Each attendee makes a list of validation criteria for the system/product
13. Draft Document is prepared.

QUALITY FUNCTION DEPLOYMENT

- QFD is the technique that translates needs of the customer into technical requirements of the software.
- QFD concentrates on maximizing customer satisfaction
- QFD identifies what is valuable to the customer and then deploys these values throughout the engineering process.
- QFD used interviews, surveys, analysis of historical data & creates "Customer Voice Table"
- QFD identifies three types of requirements
  1. Normal Requirement
  2. Expected Requirements
  3. Exciting Requirements
- QFD Concepts:
  1. Function Deployment
  2. Information Deployment
  3. Task / Behavior Deployment
  4. Value analysis

USER SCENARIOS

The Scenario often called as “Use-Cases” or “user stories”, provide a description of how the system will be used. As requirements are gathered, an overall vision of system functions & features begins to materialize. However it is difficult to move into more technical software engineering activities until the software team understands how these functions & features will be used by different classes of end-users. To accomplish this developers & users create a set of scenarios that helps to identify a thread of usage for the system to be constructed.

ELICITATION WORK PRODUCTS

- Statement of need and feasibility
- Bounded statement of scope
- A list of all stakeholders
- Description of System technical environment
- List of requirements /constraints (Function wise)
- Set of usage scenarios
- Any prototype developed to define requirements

Elaboration-

- info obtain during inception n elicitation is expanded and refined .
- RE activity focuses on developing a model of S/W functions , features , constraints.
- User scenarios , they are parsed into class

  Diagrams n other UML diagrams r produced.

  Analysis model is one produced at end.

NEGOTIATING REQUIREMENTS

- Customer is asked to balance the functionality, performance and other system or product characteristics against cost & time to market

- The intent of negotiations is to develop a project plan

- The best negotiations strive for “win-win” result.

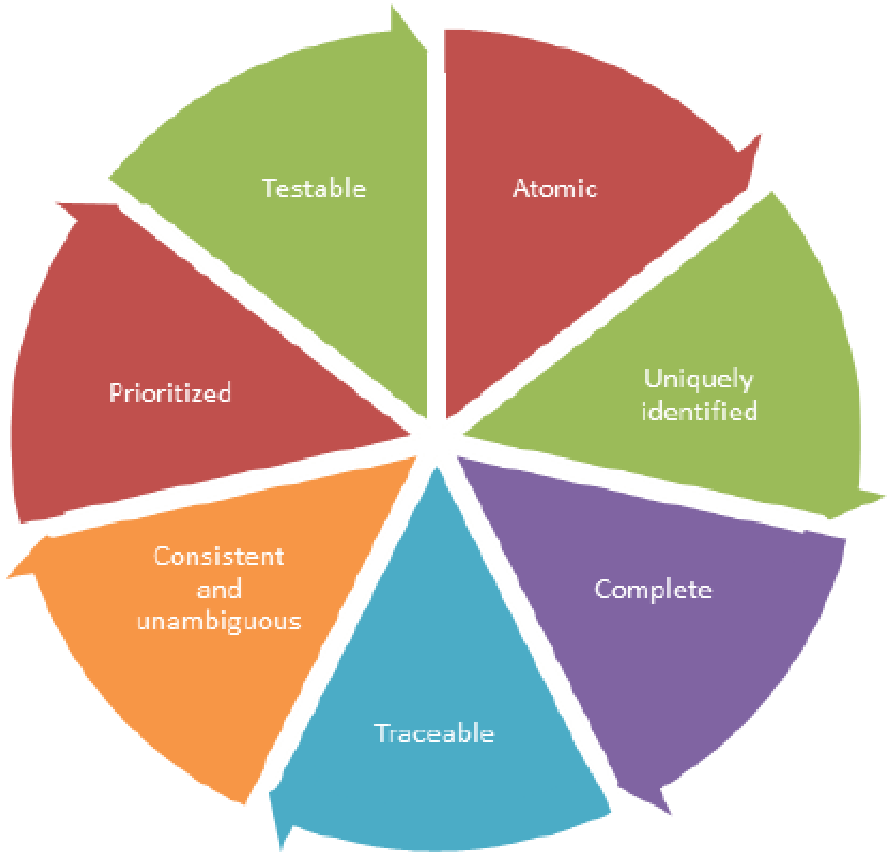
VALIDATING REQUIREMENTS

- As each element of Analysis Model is created, it is examined for consistency, omissions & ambiguity.
- The requirements represented by the model are priotorized by the customer & grouped into requirement packages that will be implemented as software increments & delivered to customer.

The review of Analysis model by FTR addresses:

- consistency of each requirement with the overall objective
- proper specifications of requirements
- necessity of requirement
- Is each requirement bounded & unambiguous?
- source of each reqt.
- check for conflicting reqt. * Info., function & Behavior.
- technical feasibility of reqt. * Model partitioning
- testability of reqt. * Usage of reqt. Pattern.

Requirement Analysis



SDLC –Framework activities

- C=>P=>M=>C=>D(Test & Deliver)
- Communication
- Planning
- (Requirement Engg=>SRS(…+Analysis –info,behavior,data flow))
- Modeling(Analysis(UML & Design(Architecture,Interface, Components))
- Coding
- Deployment

BUILDING THE ANALYSIS MODEL

- However the written word is wonderful vehicle for communication, but it is not necessarily the best way to represent the requirement for computer software
- Analysis Modeling uses combination of text & diagrammatic forms to depict requirements of data, functions & behavior in a way that is easy to understand & more importantly, straightforward to review for correctness, completeness & consistency
- A software Engineer (or a Systems Analyst) builds the model using requirements elicited from the customer
- It is very important activity because to validate software requirements , you need to view them from a number of different points of view. Analysis model represents requirements in " multiple dimensions" , thereby increasing the probability that errors will be found, that inconsistency will surface and that omissions will be uncovered

ANALYSIS RULES OF THUMB

- The model should focus on the requirements that are vissible within the problem or business domain
- The level of abstraction should be relatively high
- 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 MODELIMG 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

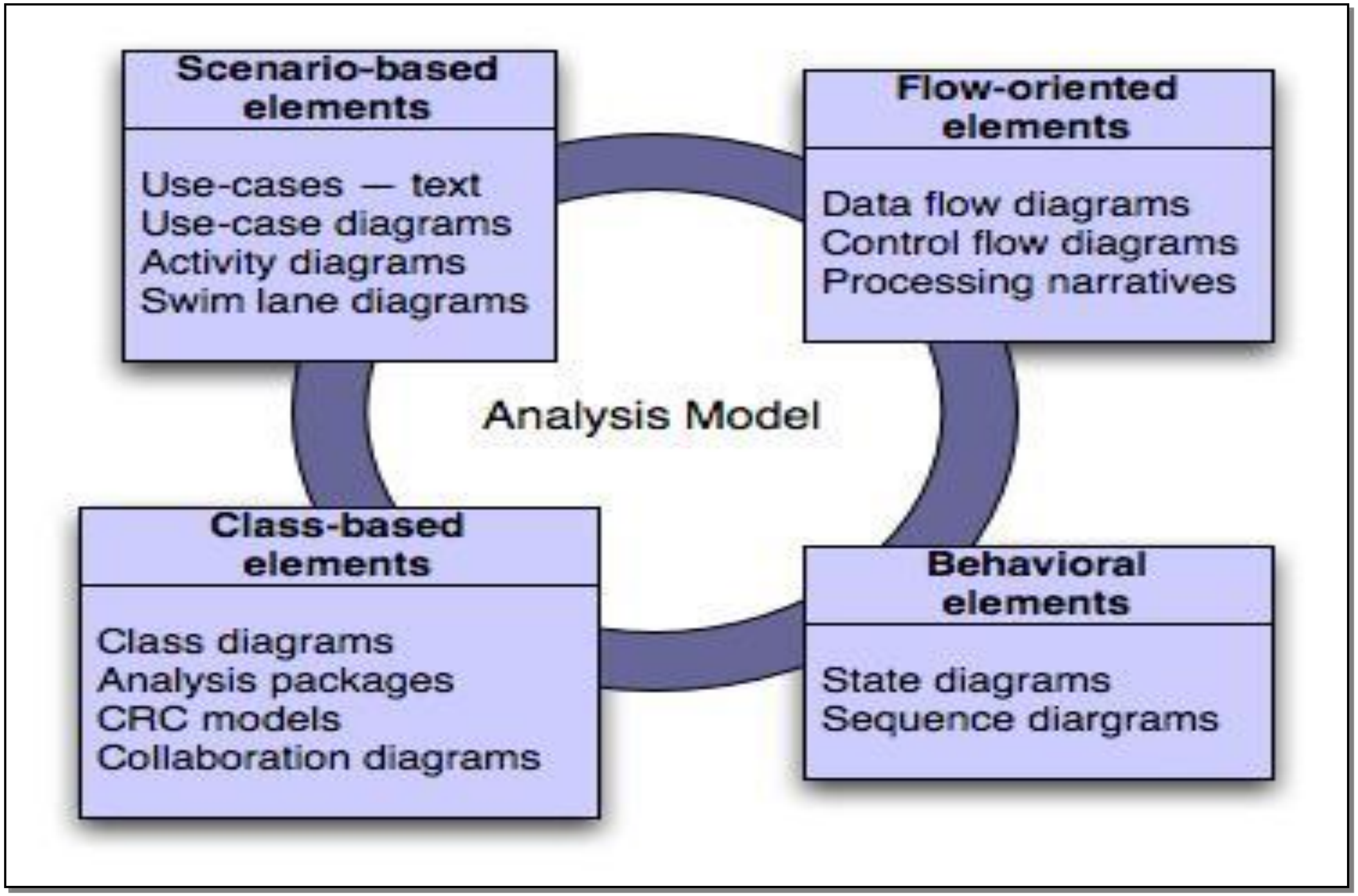
Analysis Model

- 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

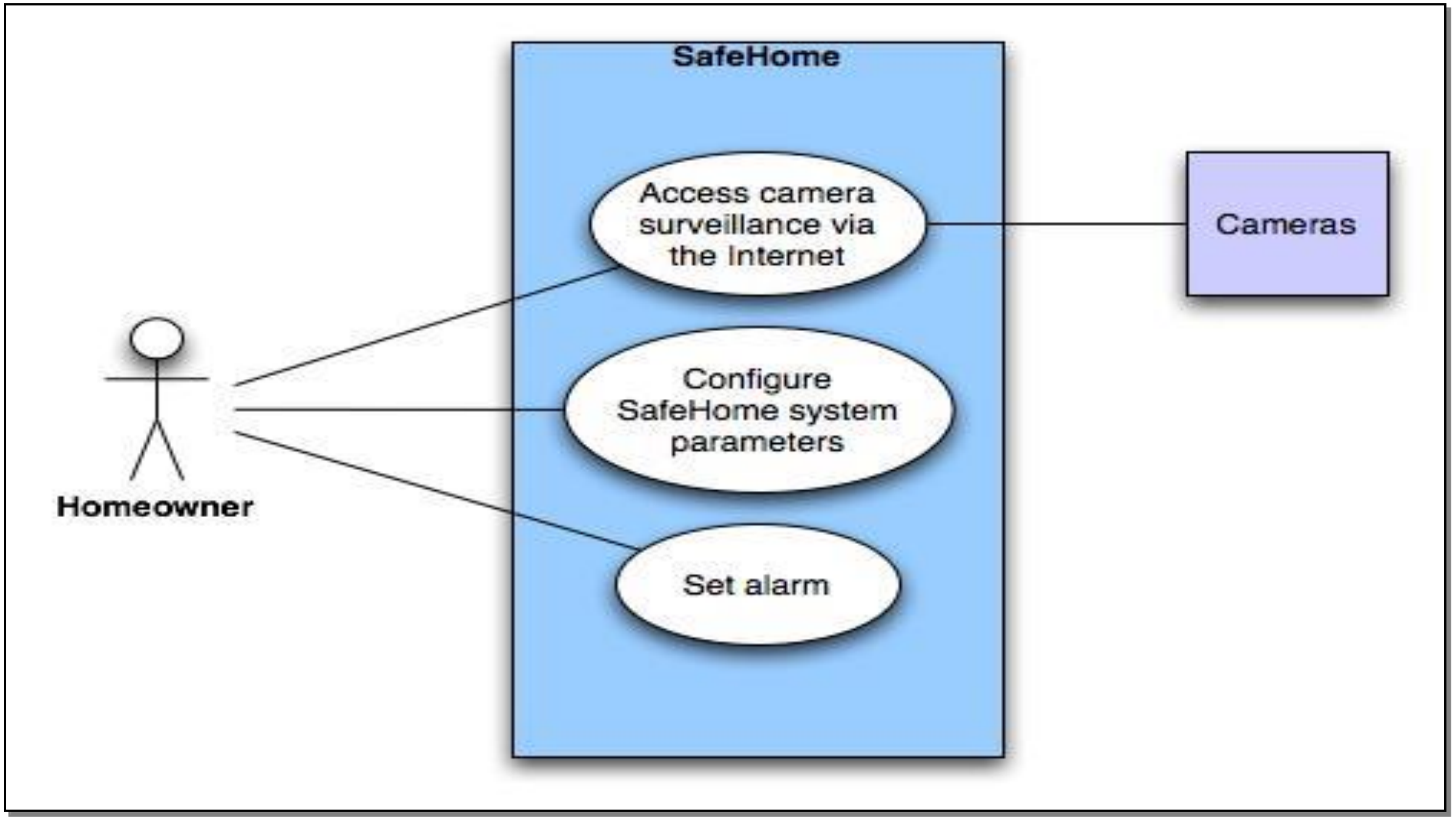
Scenario-Based Modeling

- 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
- Use cases can be represented as a text narrative or as a 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


SafeHome
Access camera surveillance via the Internet
Configure SafeHome system parameters
Set alarm
Cameras
Homeowner

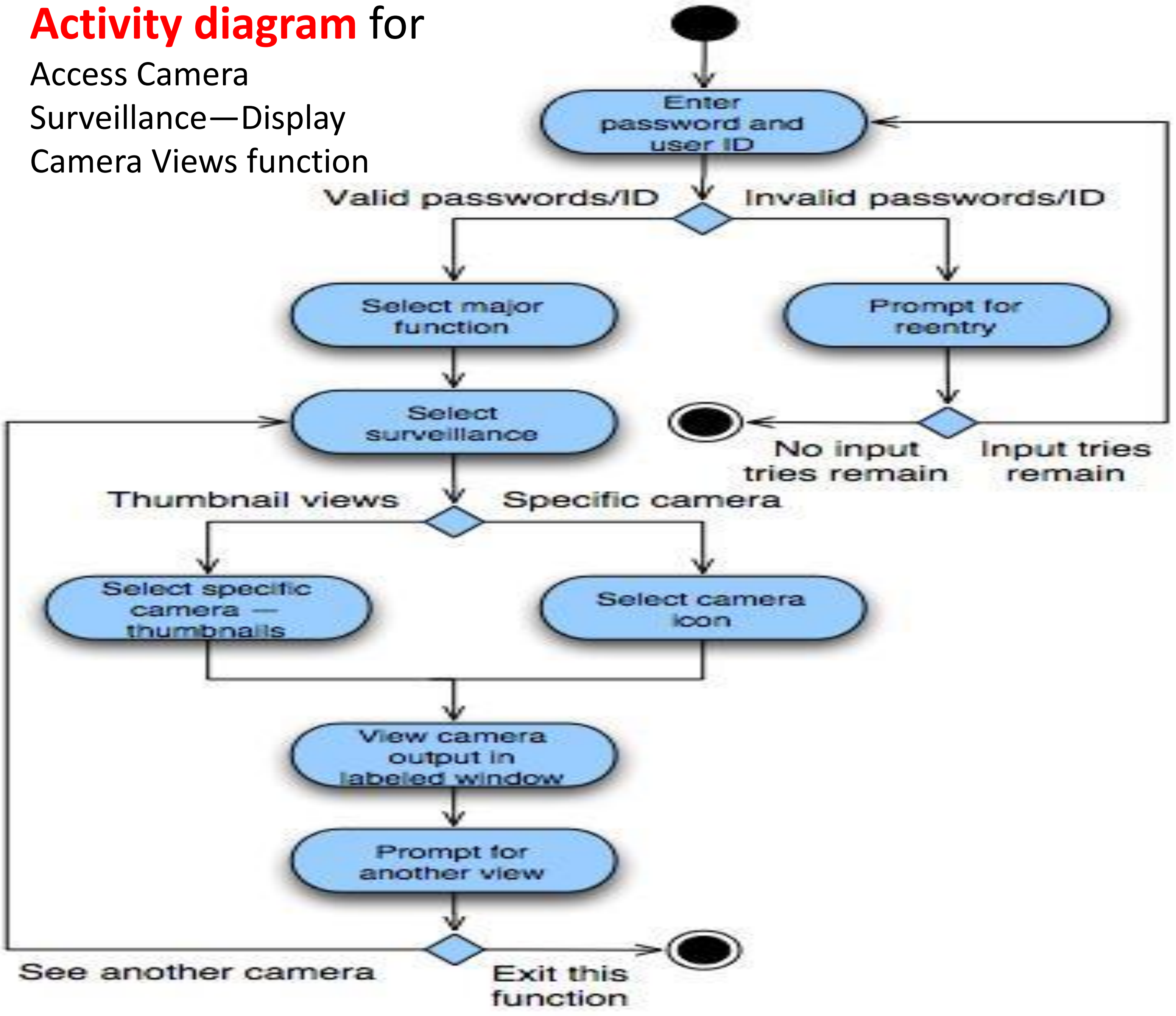
Use-case diagram for surveillance function

Alternative Actions

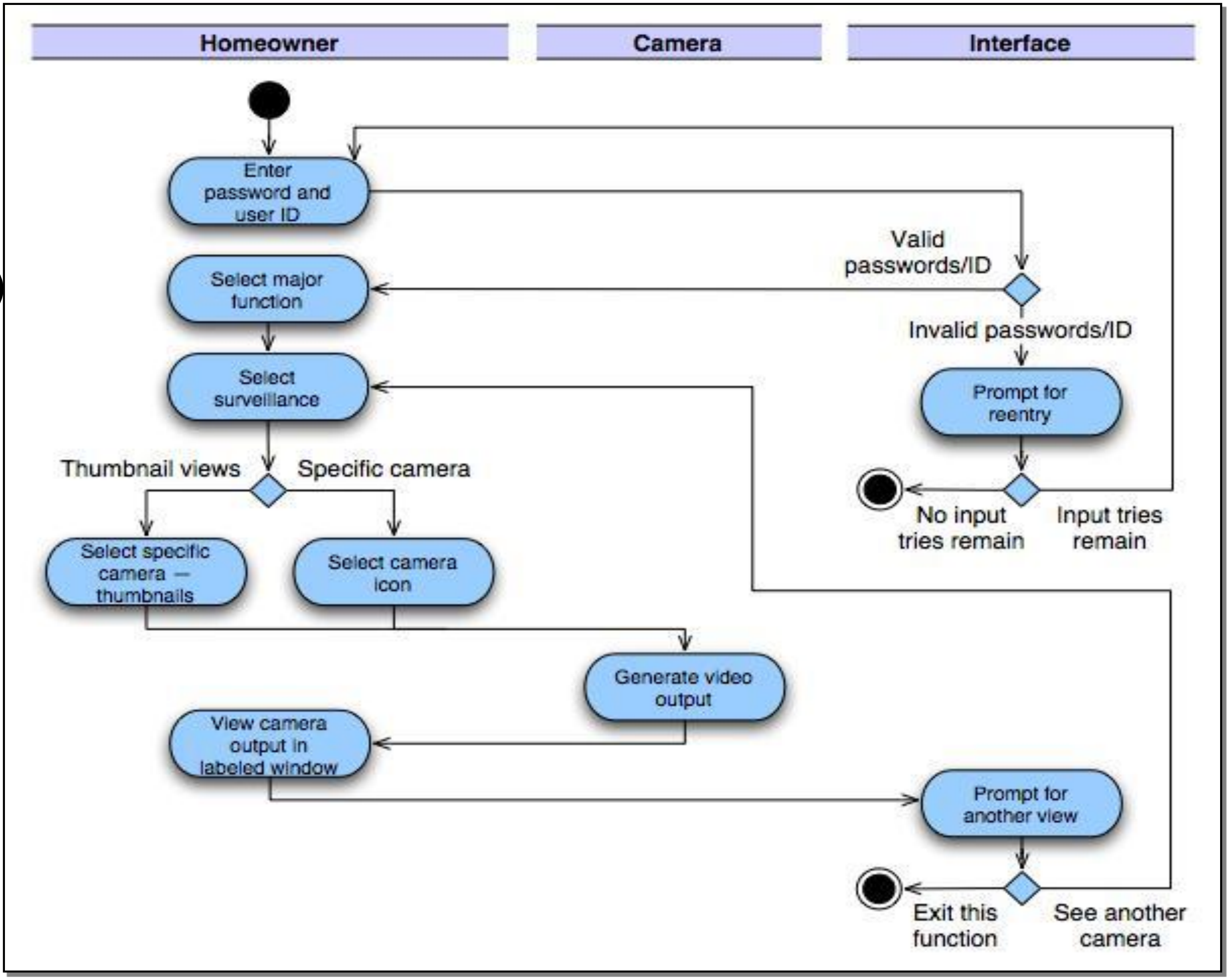
- Can the actor take some other action at this point?
- Is it possible that the actor will encounter some error condition at this point? - Exceptions
- Is it possible that the actor will encounter behavior invoked by some event outside the actor’s control?

Activity diagram for

Access Camera Surveillance—Display Camera Views function



Swimlane diagram (ACS-DCV)



Flow-Oriented Modeling

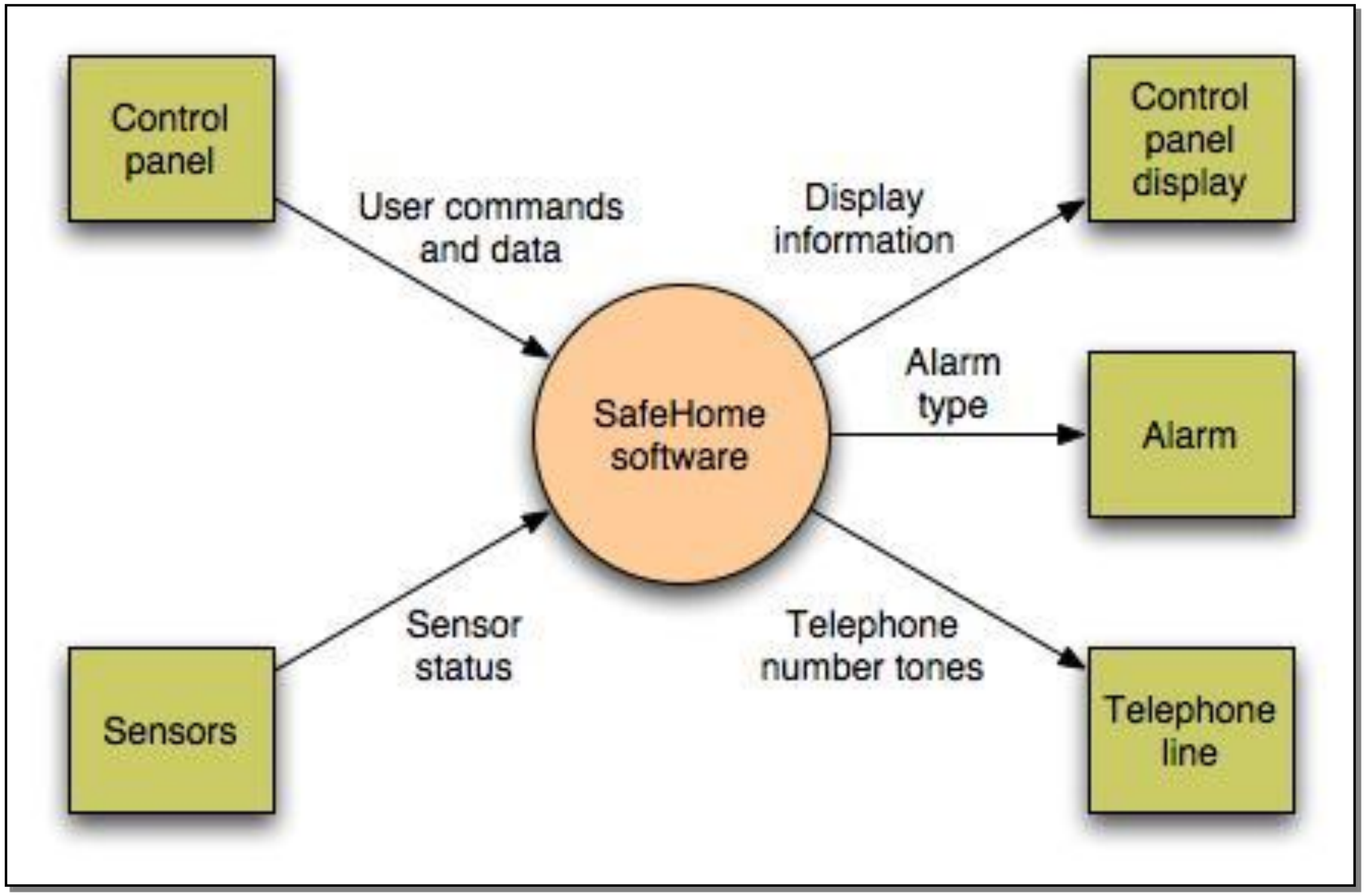
- Data Flow Modeling continues to be one of the most widely used analysis technique today. Although Data Flow Diagrams (DFDs) & other related diagrams and information are not a formal part of UML, they can be used to complement UML diagrams & provide additional insight into the system requirement and flow
- 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

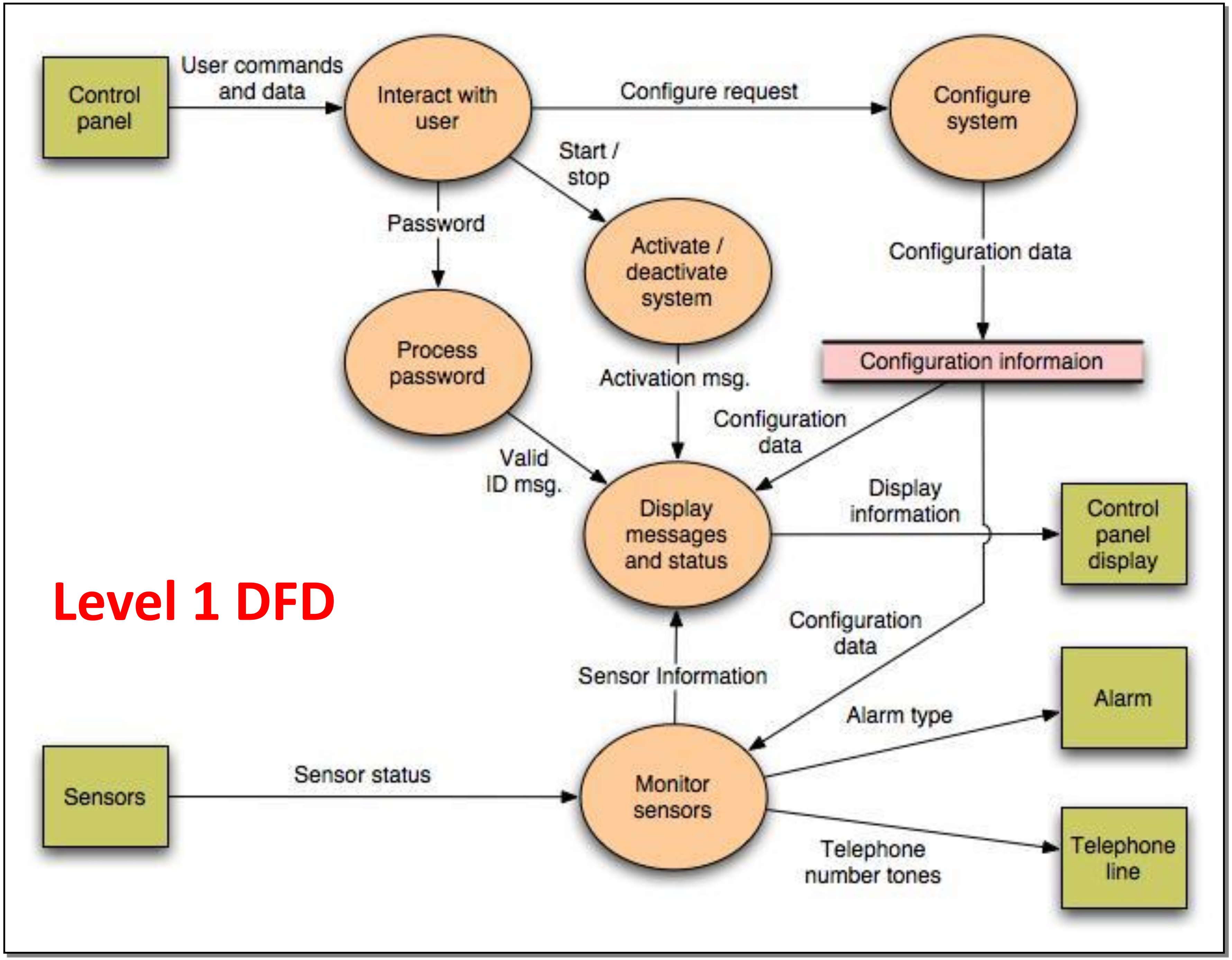
Guidelines :

- 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.

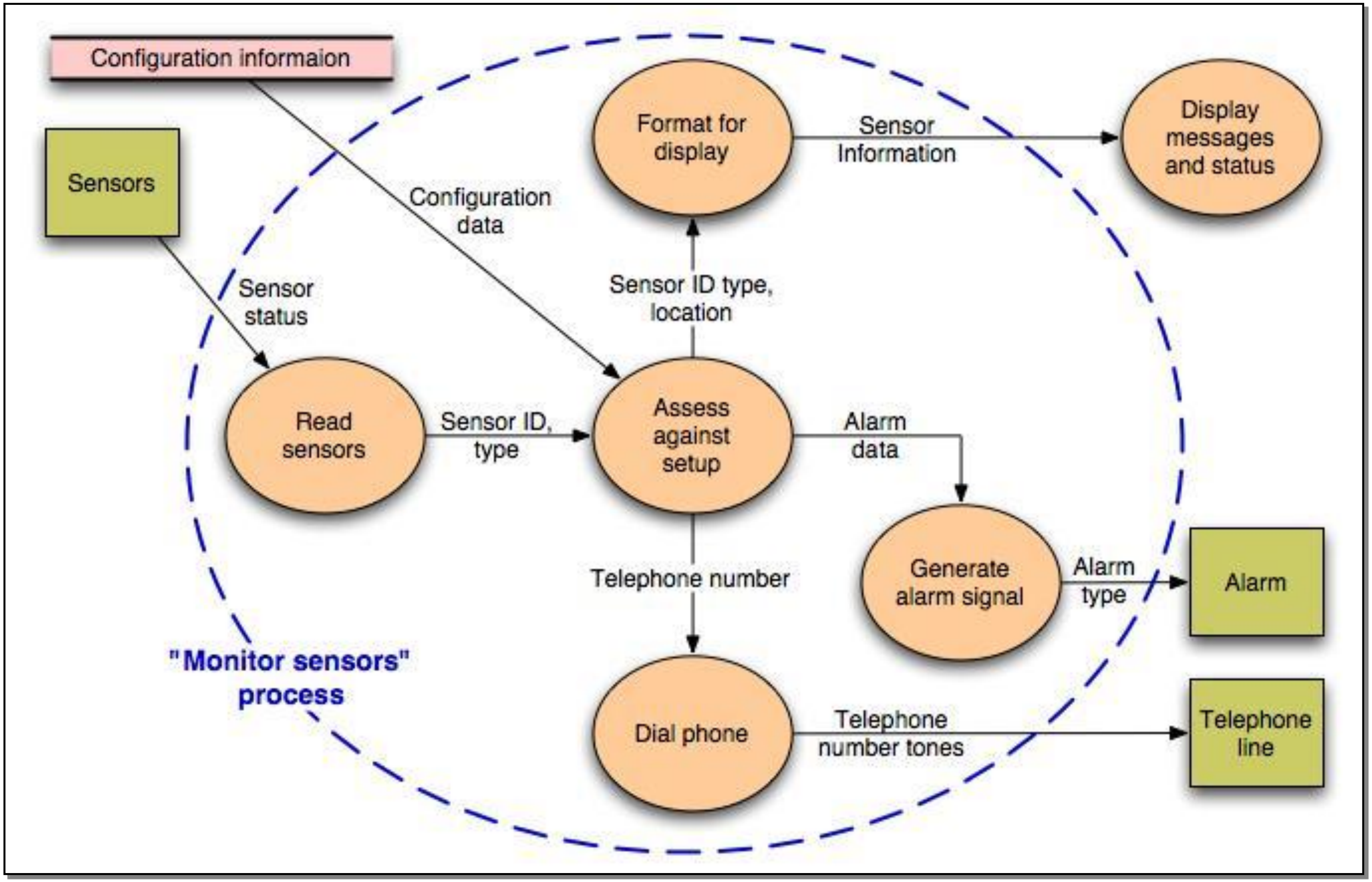
Data Flow Diagram



Context-level DFD for *SafeHome* security function


Control panel
User commands and data
Interact with user
Configure request
Configure system
Start / stop
Password
Activate / deactivate system
Configuration data
Configuration informaion
Process password
Activation msg.
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 1 DFD



Level 2 DFD that refines the monitor sensors process

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…

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 :

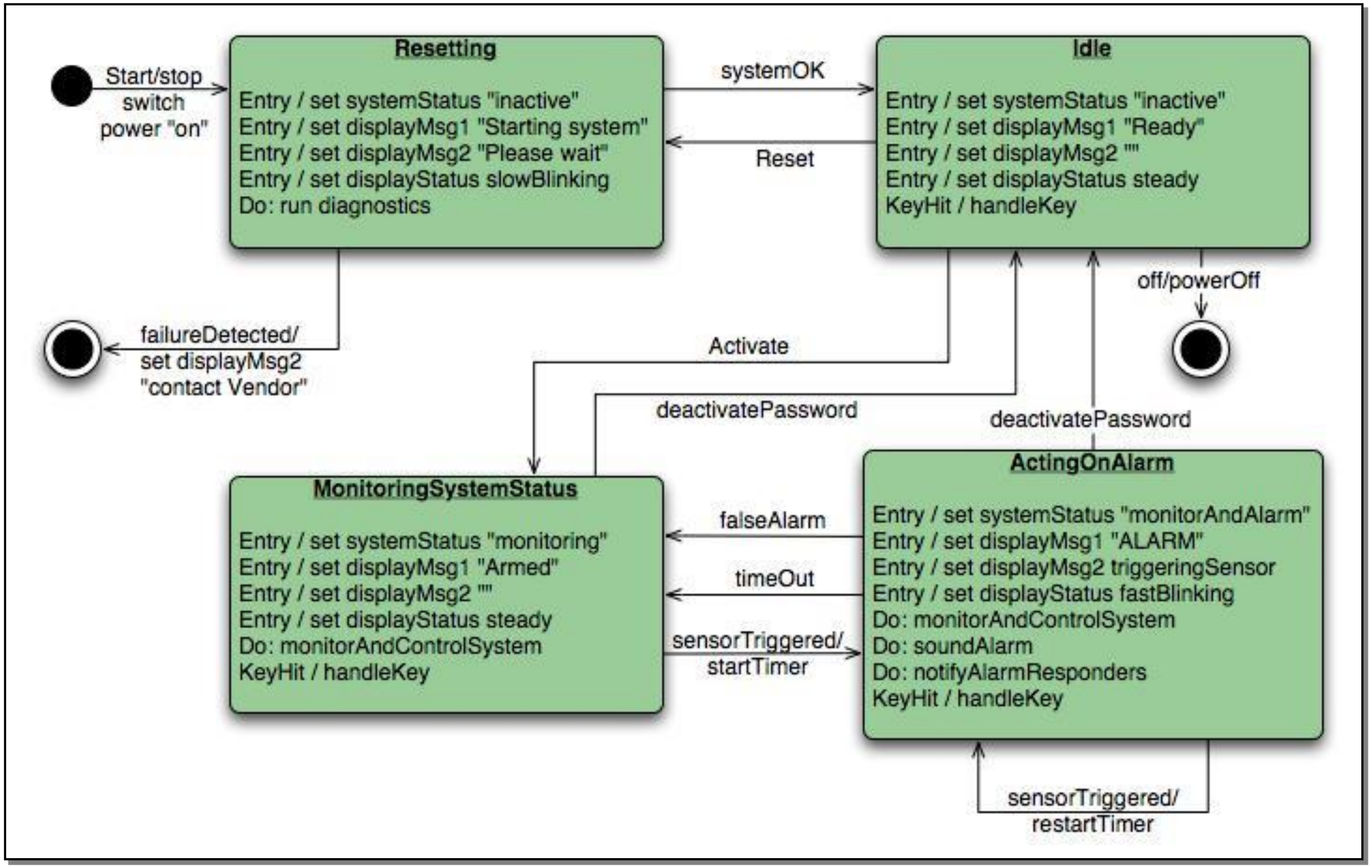
- List all sensors that are “read” by the software
- List all “interrupts conditions”
- List all “switches” that are actuated by an operator
- List all “data conditions” (empty, full, jammed)
- Review all control items w.r.t. noun/verb parse of processing narrative. They are possible I/O for control 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



Start/stop switch power "on"
Resetting
Entry / set systemStatus "inactive"
Entry / set displayMsg1 "Starting system"
Entry / set displayMsg2 "Please wait"
Entry / set displayStatus slowBlinking
Do: run diagnostics
systemOK
Reset
Idle
Entry / set systemStatus "inactive"
Entry / set displayMsg1 "Ready"
Entry / set displayMsg2 ""
Entry / set displayStatus steady
KeyHit / handleKey
off/powerOff
failureDetected/ set displayMsg2 "contact Vendor"
Activate
deactivatePassword
deactivatePassword
MonitoringSystemStatus
Entry / set systemStatus "monitoring"
Entry / set displayMsg1 "Armed"
Entry / set displayMsg2 ""
Entry / set displayStatus steady
Do: monitorAndControlSystem
KeyHit / handleKey
falseAlarm
timeOut
sensorTriggered/ startTimer
ActingOnAlarm
Entry / set systemStatus "monitorAndAlarm"
Entry / set displayMsg1 "ALARM"
Entry / set displayMsg2 triggeringSensor
Entry / set displayStatus fastBlinking
Do: monitorAndControlSystem
Do: soundAlarm
Do: notifyAlarmResponders
KeyHit / handleKey
sensorTriggered/ restartTimer

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 Selection Criteria

Inclusion of potential class in Analysis Model:
(Selection Characteristics)

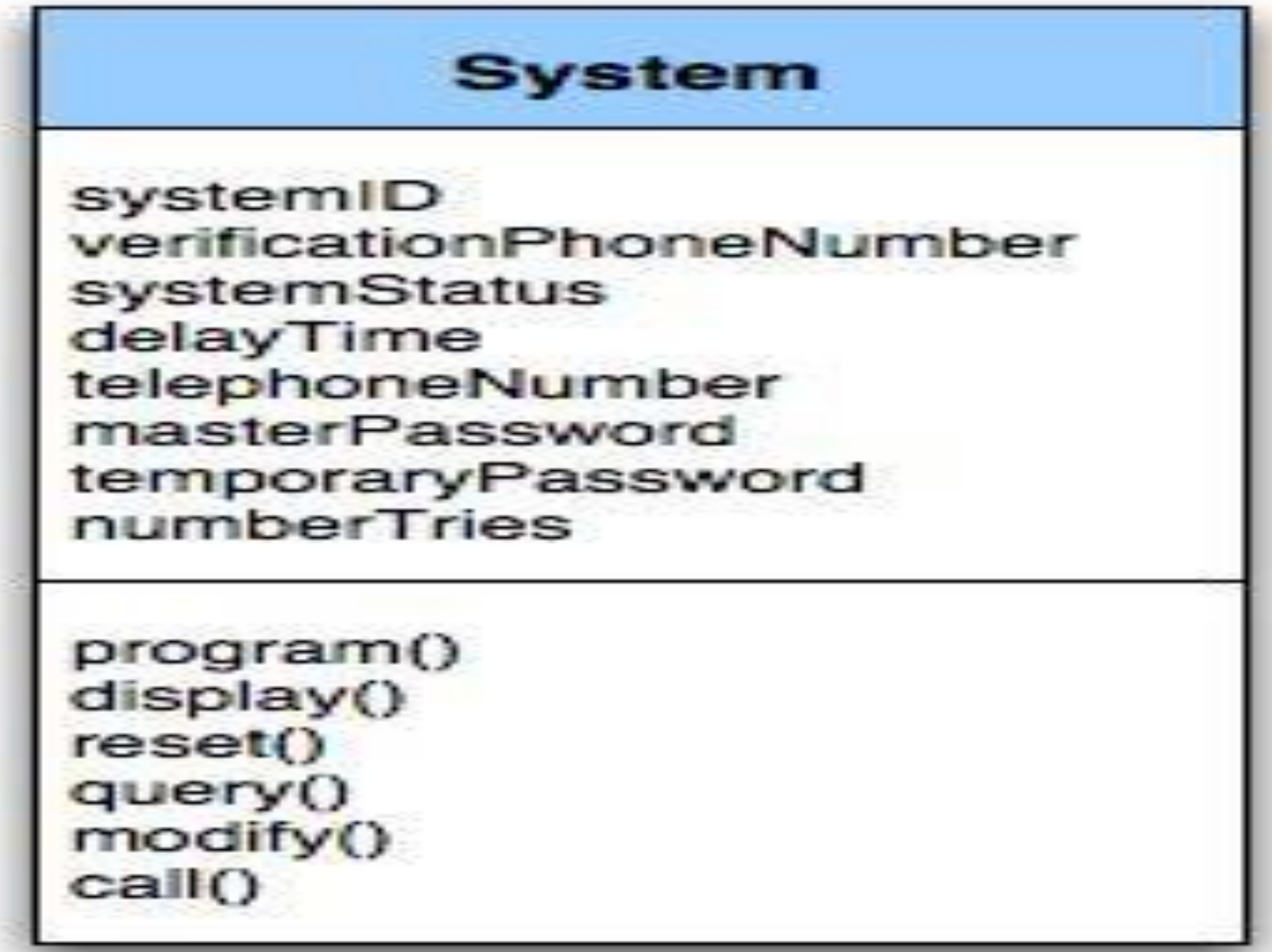
1. Retained information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential requirements

After performing Grammatic parse on narrative of Safe Home Security system following potential classes are identified

Identifying Classes

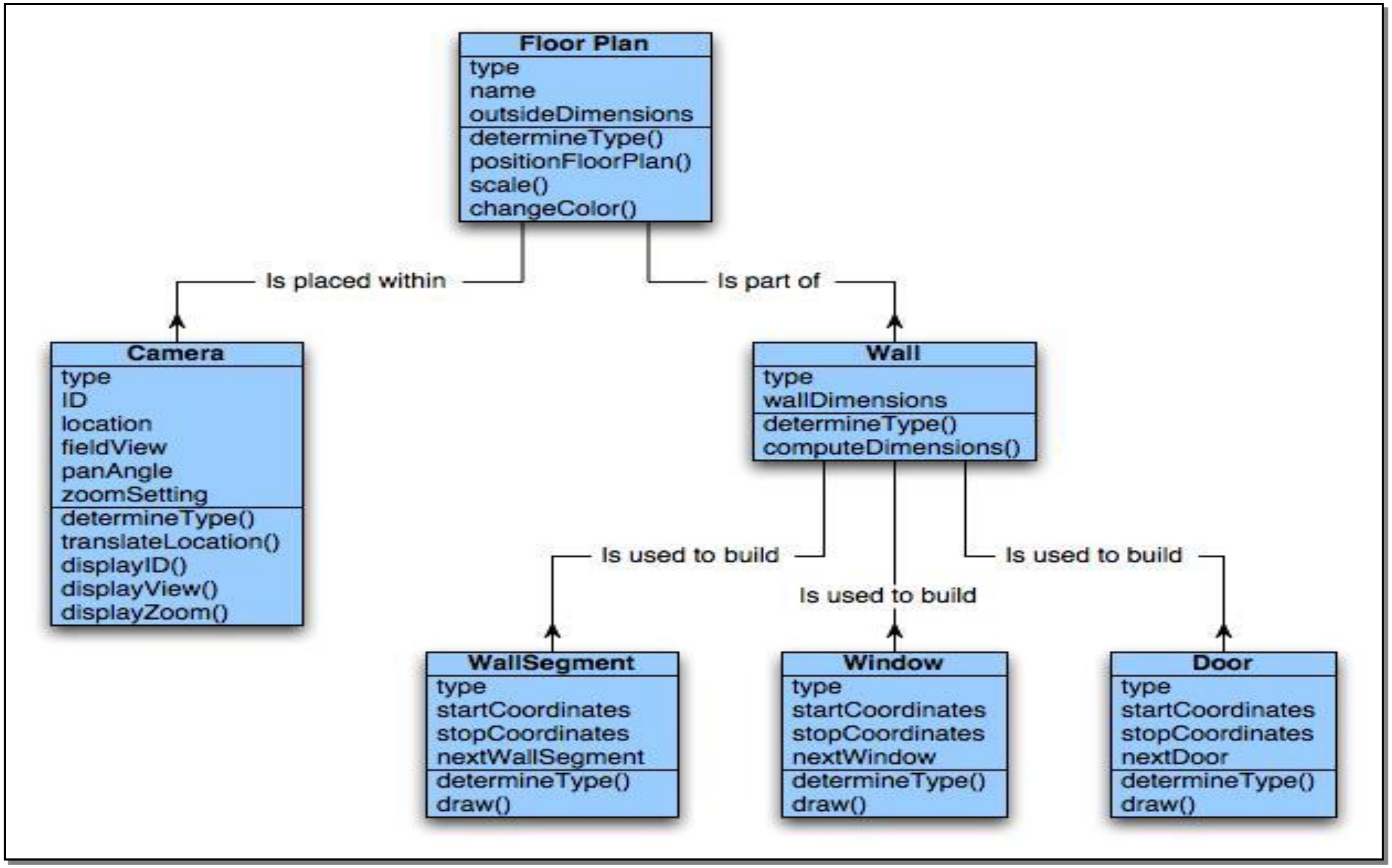
Potential class	**Classification**	**Accept / Reject**
homeowner	role; external entity	reject: 1, 2 fail
sensor	external entity	accept
control panel	external entity	accept
installation	occurrence	reject
(security) system	thing	accept
number, type	not objects, attributes	reject: 3 fails
master password	thing	reject: 3 fails
telephone number	thing	reject: 3 fails
sensor event	occurrence	accept
audible alarm	external entity	accept: 1 fails
monitoring service	organizational unit	reject: 1, 2 fail

Class Diagram



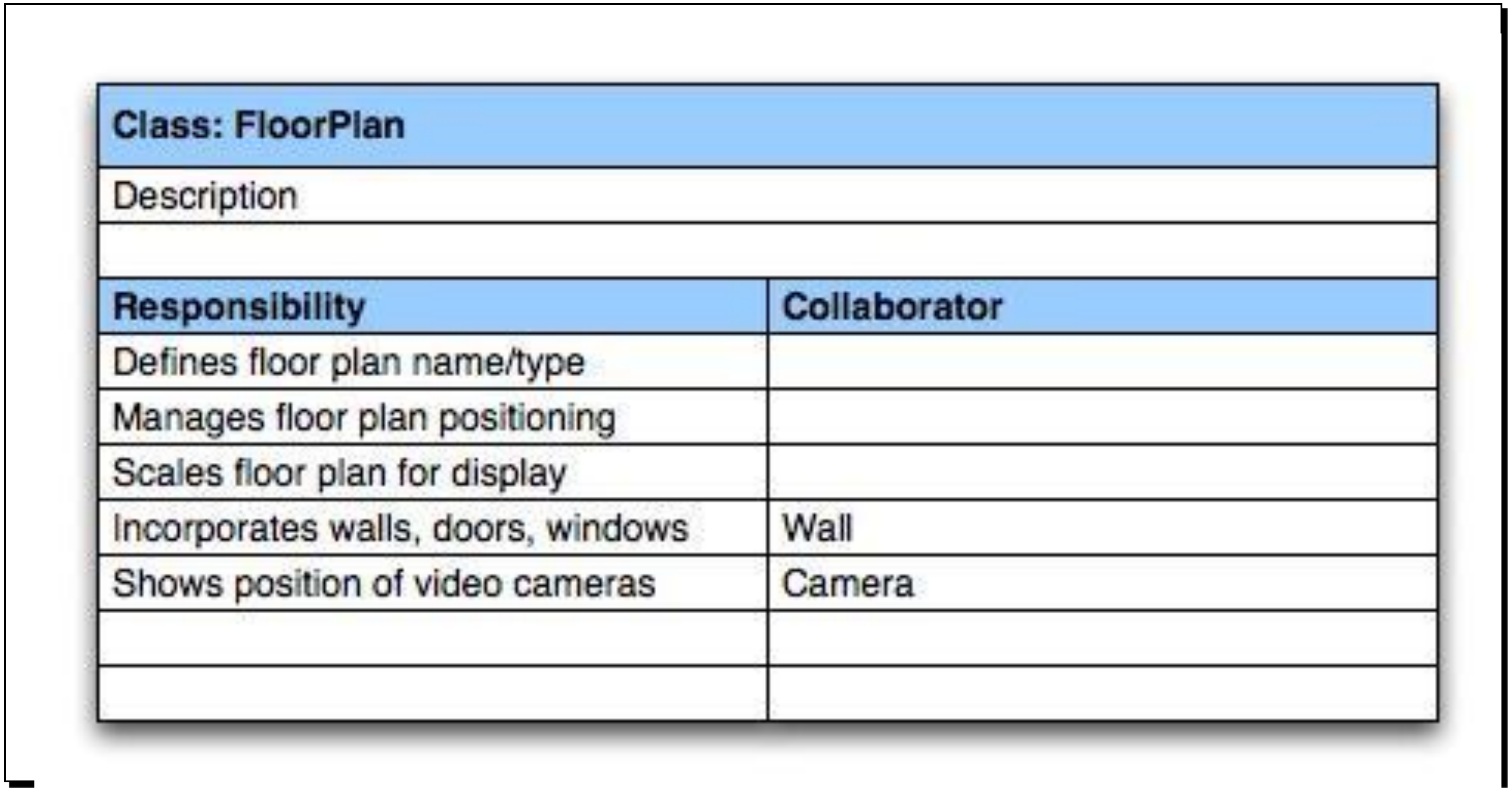
Class diagram for the system class

Class Diagram



Class diagram for FloorPlan

CRC Modeling



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

A CRC model index card for FloorPlan class

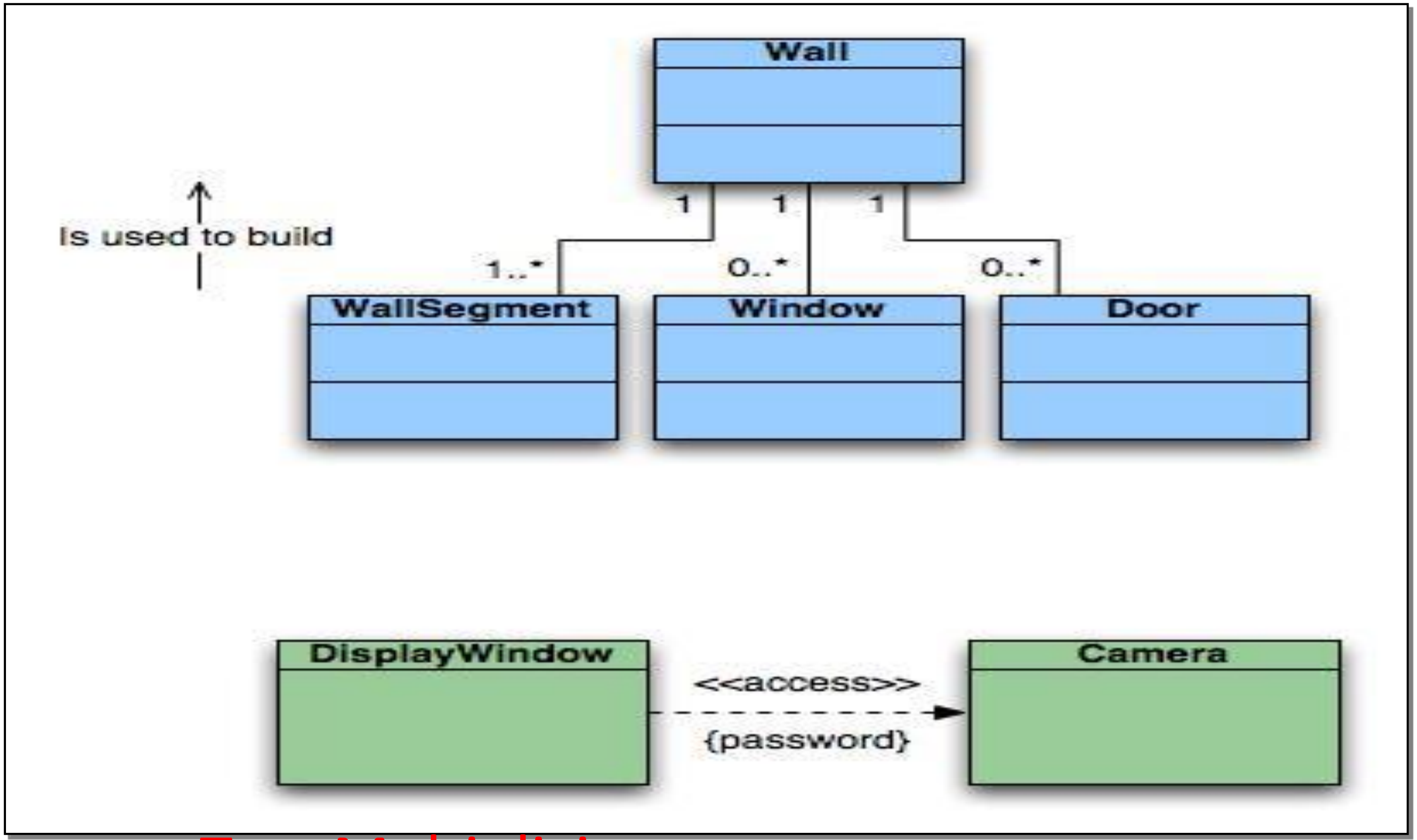
Class Responsibilities

- Distribute system intelligence across classes.
- State each responsibility as generally as possible.
- Put information and the behavior related to it in the same class.
- Localize information about one thing rather than distributing it across multiple classes.
- Share responsibilities among related classes, when appropriate.

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

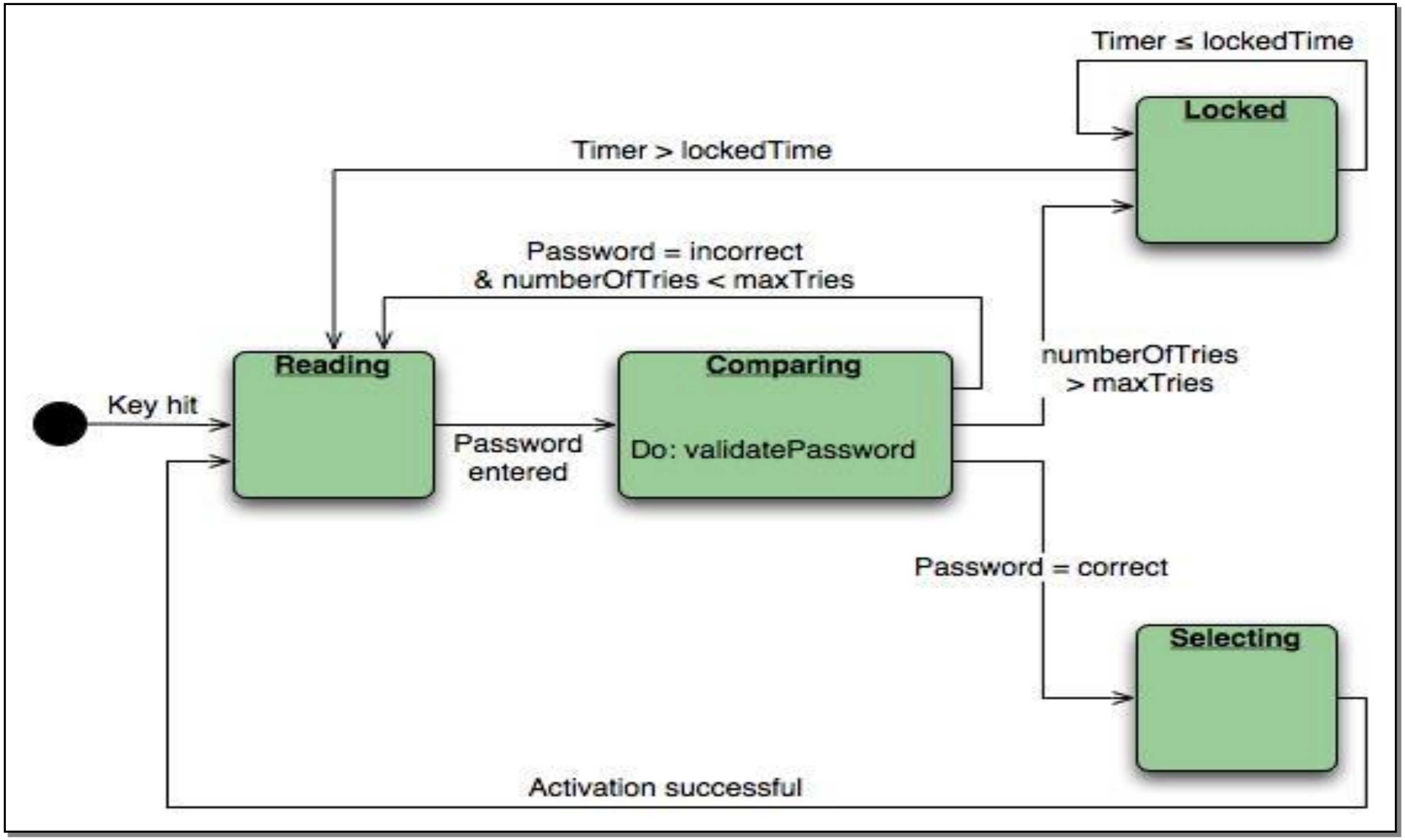
Bottom: Dependencies

Behavioral Modeling

Identifying Events :

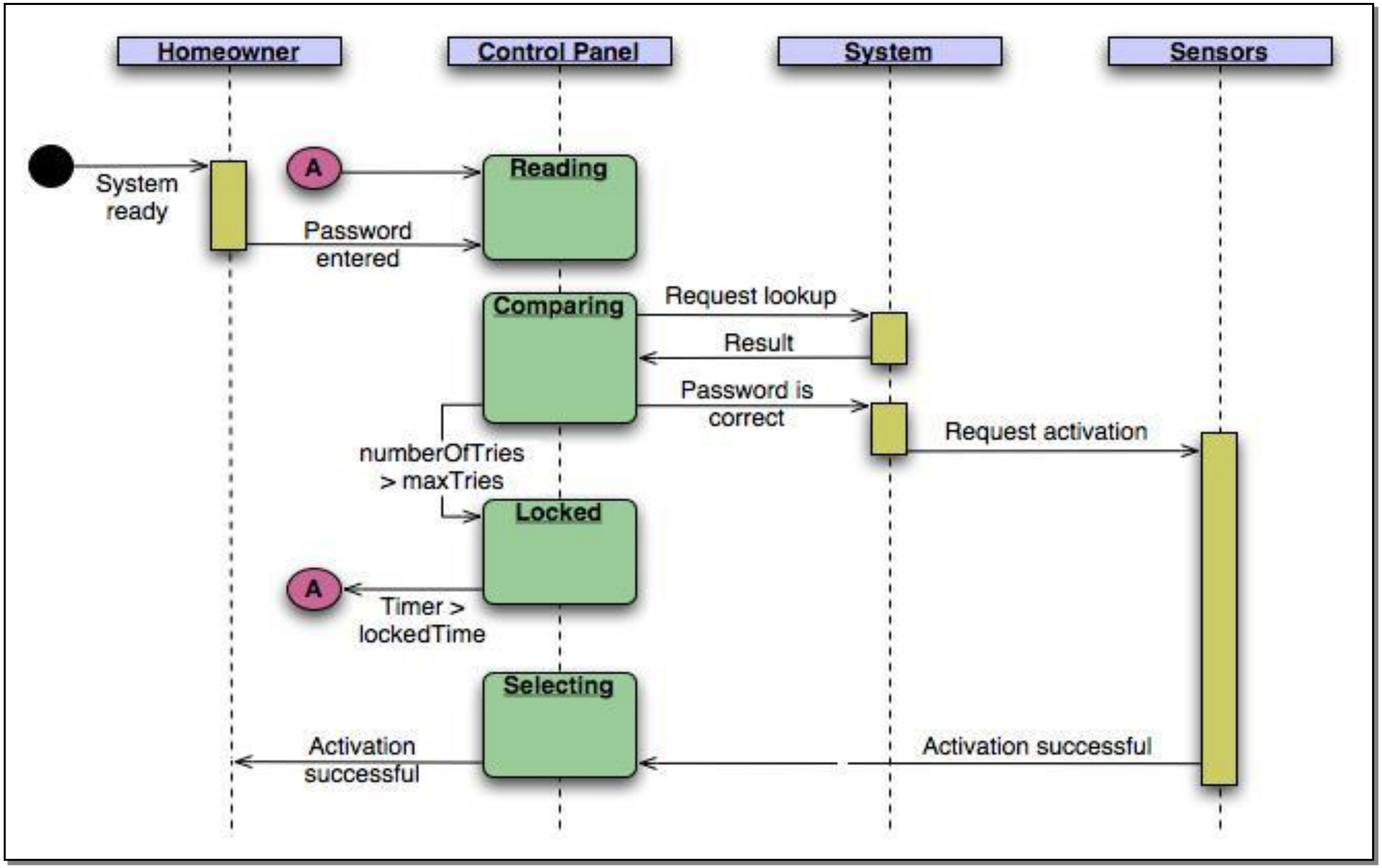
- A use-case is examined for *points of information exchange*.
- The homeowner uses the keypad to key in a four-digit password. The password is compared with the valid password stored in the system. If the password in incorrect, the control panel will beep once and reset itself for additional input.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

SOFTWARE REQUIREMENTS SPECIFICATION

- **A Software Requirements Specification [SRS]** is a document that is created when a detailed description of all aspects of the software to be built must be specified before the project is to commence.

- When a software is to be developed by a third party, when a lack of specification would create severe business issues, or when a system is extremely complex or business critical, an SRS is prepared.

- Karl Wiegers has developed a worthwhile template at **www.processimpact.com/process_assets/srs_template.doc**

SOFTWARE REQUIREMENTS SPECIFICATION

SRS TEMPLATE

Table of Contents :

SOFTWARE REQUIREMENTS SPECIFICATION

SRS TEMPLATE … contd.

2.5 Design and Implementation Constraints

2.6 User Documentation

2.7 Assumptions and Dependencies

3. **System Features**

   3.1 System Feature 1

   3.2 System Feature 2 (and so on)

4. **External Interface Requirements**

   4.1 User Interfaces

   4.2 Hardware Interfaces

   4.3 Software Interfaces

   4.4 Communications Interfaces

SOFTWARE REQUIREMENTS SPECIFICATION

SRS TEMPLATE … contd.

5. **Nonfunctional Requirements**
    - 5.1 Performance Requirements
    - 5.2 Safety Requirements
    - 5.3 Security Requirements
    - 5.4 Software Quality Attributes
6. **Other Requirements**

Appendix A : Glossary

Appendix B : Analysis Model

Appendix C : Issues List