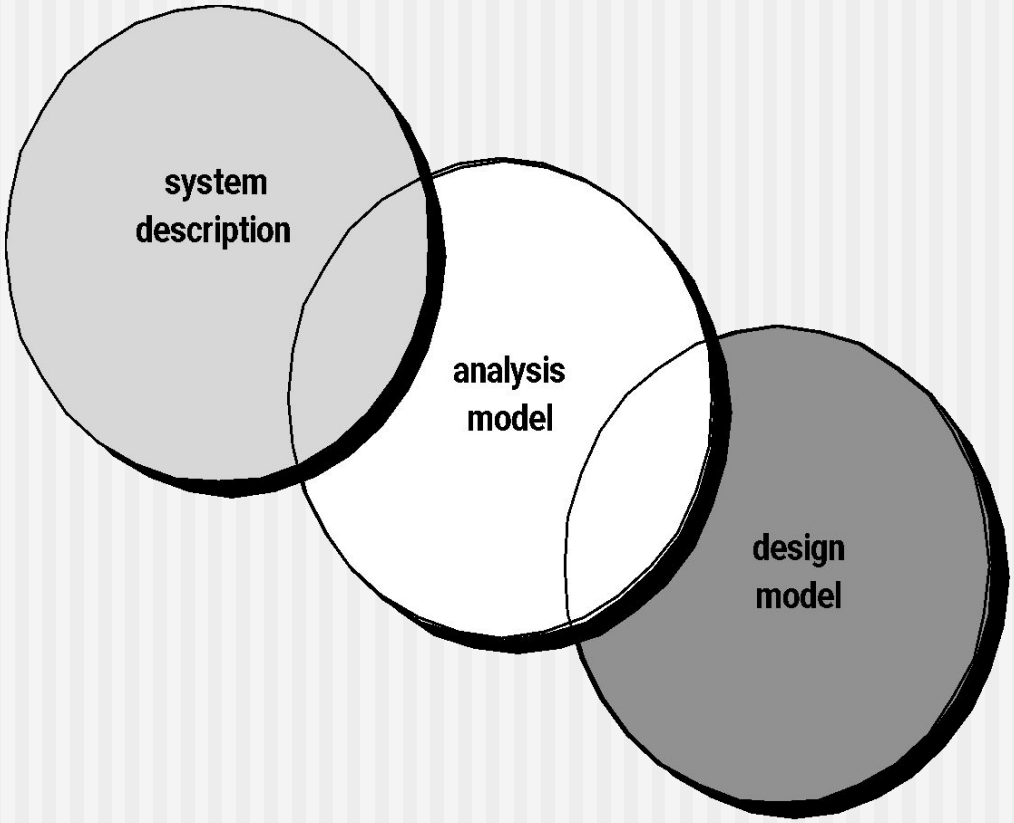
Chapter 6

- **Requirements Modeling: Scenarios, Information, and Analysis Classes**

Requirements Analysis

- Requirements analysis
  - specifies software's operational characteristics
  - indicates software's interface with other system elements
  - establishes constraints that software must meet
- Requirements analysis allows the software engineer (called an *analyst* or *modeler* in this role) to:
  - elaborate on basic requirements established during earlier requirement engineering tasks
  - build models that depict user scenarios, functional activities, problem classes and their relationships, system and class behavior, and the flow of data as it is transformed.

The requirements model as a bridge between the system description and the design model



Rules of Thumb

- The model should focus on requirements that are visible within the problem or business domain. The level of abstraction should be relatively high.
- Each element of the analysis model should add to an overall understanding of software requirements and provide insight into the information domain, function and behavior of the system.
- Delay consideration of infrastructure and other non-functional models until design.
- Minimize coupling throughout the system.
- Be certain that the analysis model provides value to all stakeholders.
- Keep the model as simple as it can be.

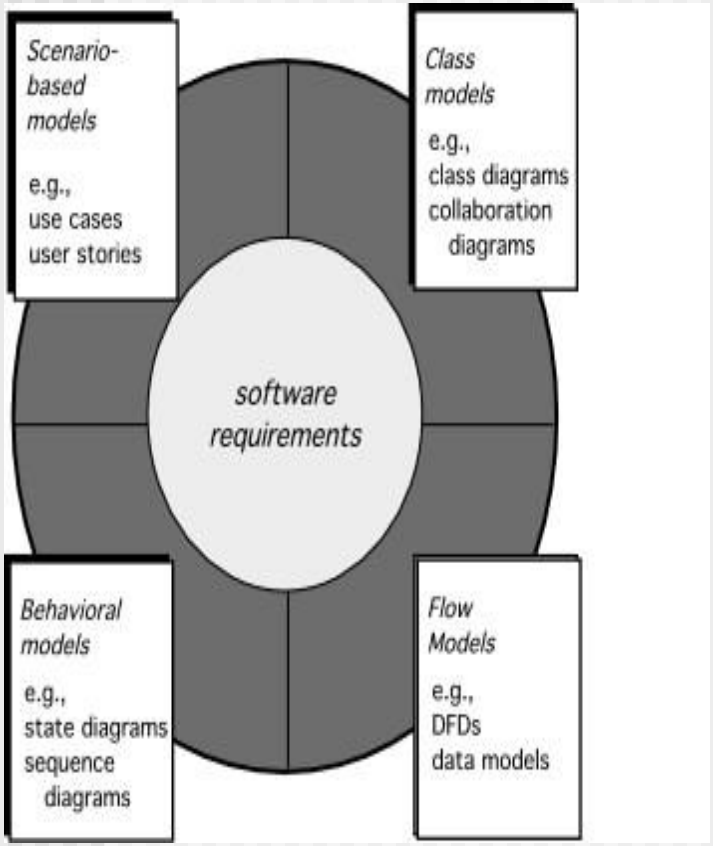
Domain Analysis

S/W Domain Analysis is the identification, analysis and specification of common requirements from a specific application domain .

- Define the domain to be investigated.
- Collect a representative sample of applications in the domain.
- Analyze each application in the sample.
- Develop an analysis model for the objects.

Domain Analysis is an ongoing software engineering activity that is not connected to any one S/W

Elements of Requirements Analysis



- Model propose combine features of structured analysis and object oriented analysis
- Analysis modeling leads to the derivation of each of these modeling elements
- Each element presents the problem from different point Of view.
- The specific content of each element may differ from project to project
- only those modeling elements That add value to the model Should be used.

Scenario-Based Modeling

Developing a software system in object oriented approach is very much dependent on understanding the problem.

Requirement modeling with Unified Modeling Language (UML) begins with the creation of scenarios in the form of use-case, activity diagram and swimlane diagrams

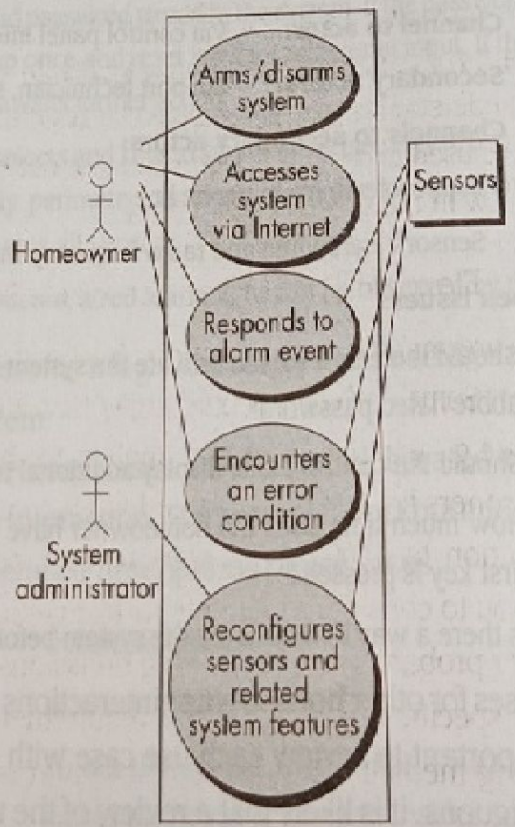
- "[Use-cases] are simply an aid to defining what exists outside the system (actors) and what should be performed by the system (use-cases)."

Use-Cases

- A scenario describes the specific actions required by the actor for a system
- *actors* represent roles play by people or devices, as the system functions
- *users* can play a number of different roles for a given scenario

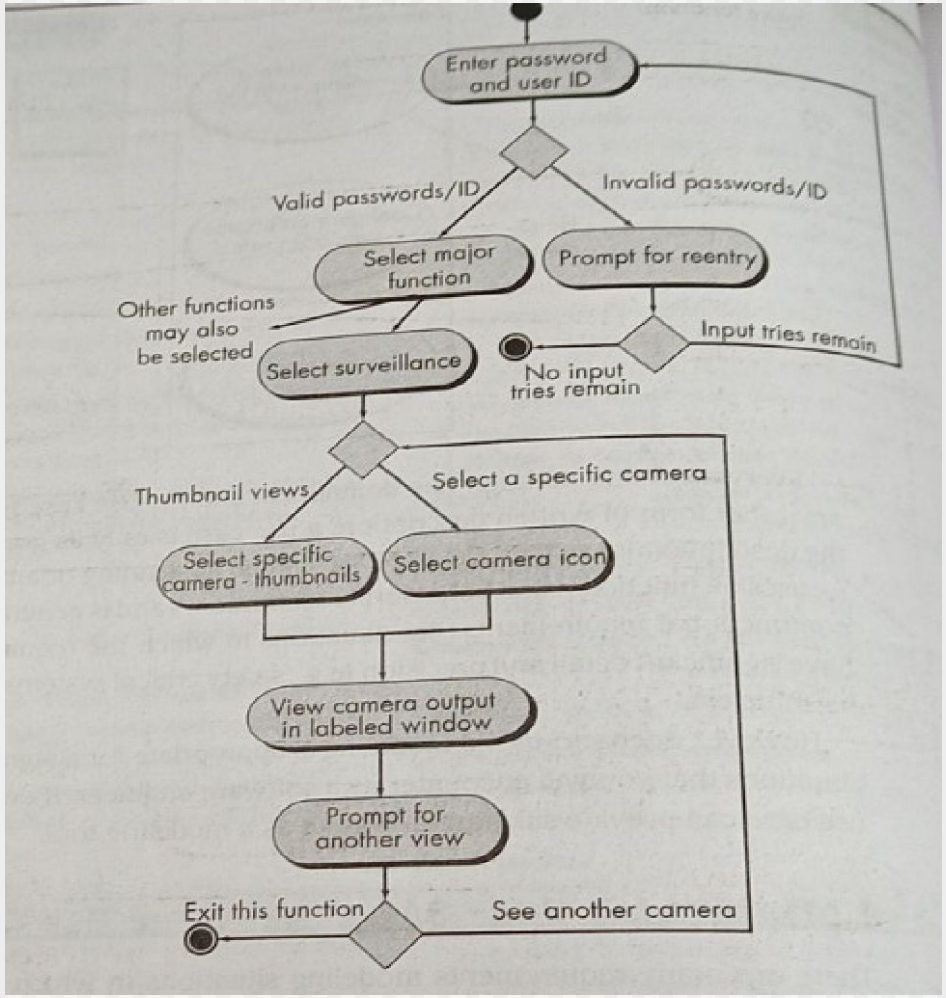
FIGURE 5.2

UML use case diagram for *SafeHome* home security function



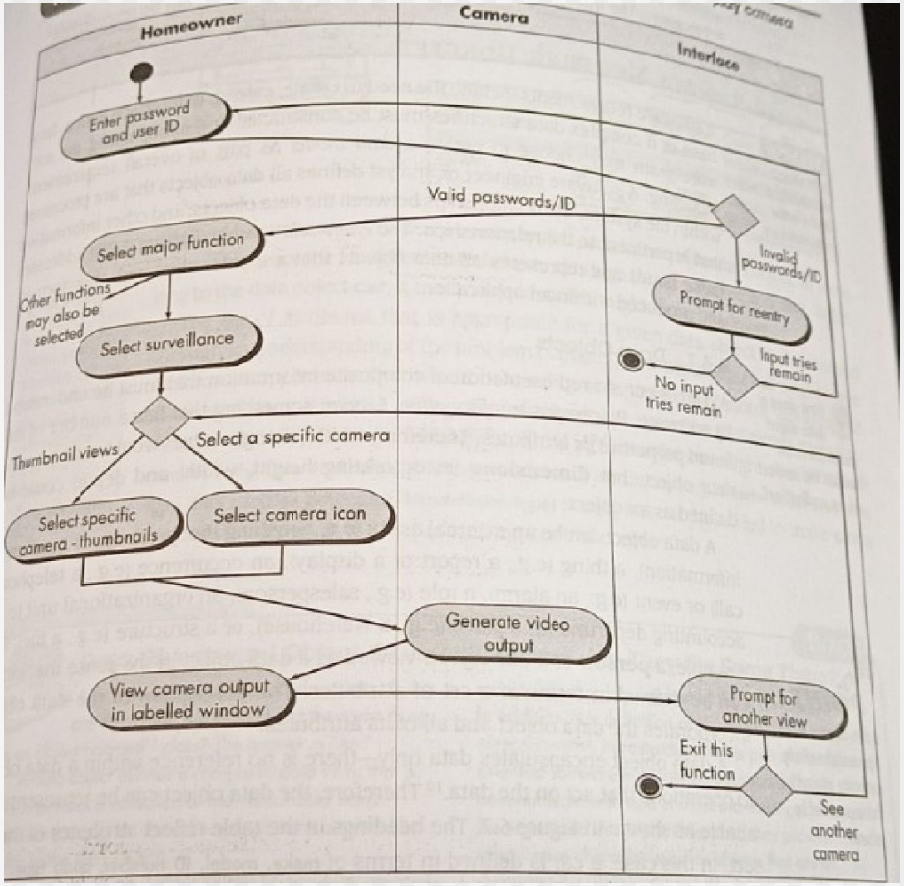
Activity Diagram

- ***Supplements*** *the use case by providing a graphical representation of the flow of interaction within a specific scenario*


Enter password and user ID
Valid passwords/ID
Invalid passwords/ID
Select major function
Prompt for reentry
Other functions may also be selected
Input tries remain
Select surveillance
No input tries remain
Thumbnail views
Select a specific camera
Select specific camera - thumbnails
Select camera icon
View camera output in labeled window
Prompt for another view
Exit this function
See another camera

Swimlane Diagrams

- *Allows the analyst to represent the flow of activities described by the use-case*
- *Also indicate which actor has responsibility for that action is described by an activity rectangle*


Homeowner
Camera
Interface
Enter password and user ID
Valid passwords/ID
Invalid passwords/ID
Select major function
Other functions may also be selected
Prompt for reentry
Select surveillance
Input tries remain
No input tries remain
Thumbnail views
Select a specific camera
Select specific camera - thumbnails
Select camera icon
Generate video output
View camera output in labelled window
Prompt for another view
Exit this function
See another camera

Flow-Oriented Modeling

- Represent the functional elements of the system and how they transform data as it moves through the system (DFD) , it provide a view of the system that is unique—it should be used to supplement other analysis model elements
- indicates how data objects relate to one another (ERD)

Class-Based 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.

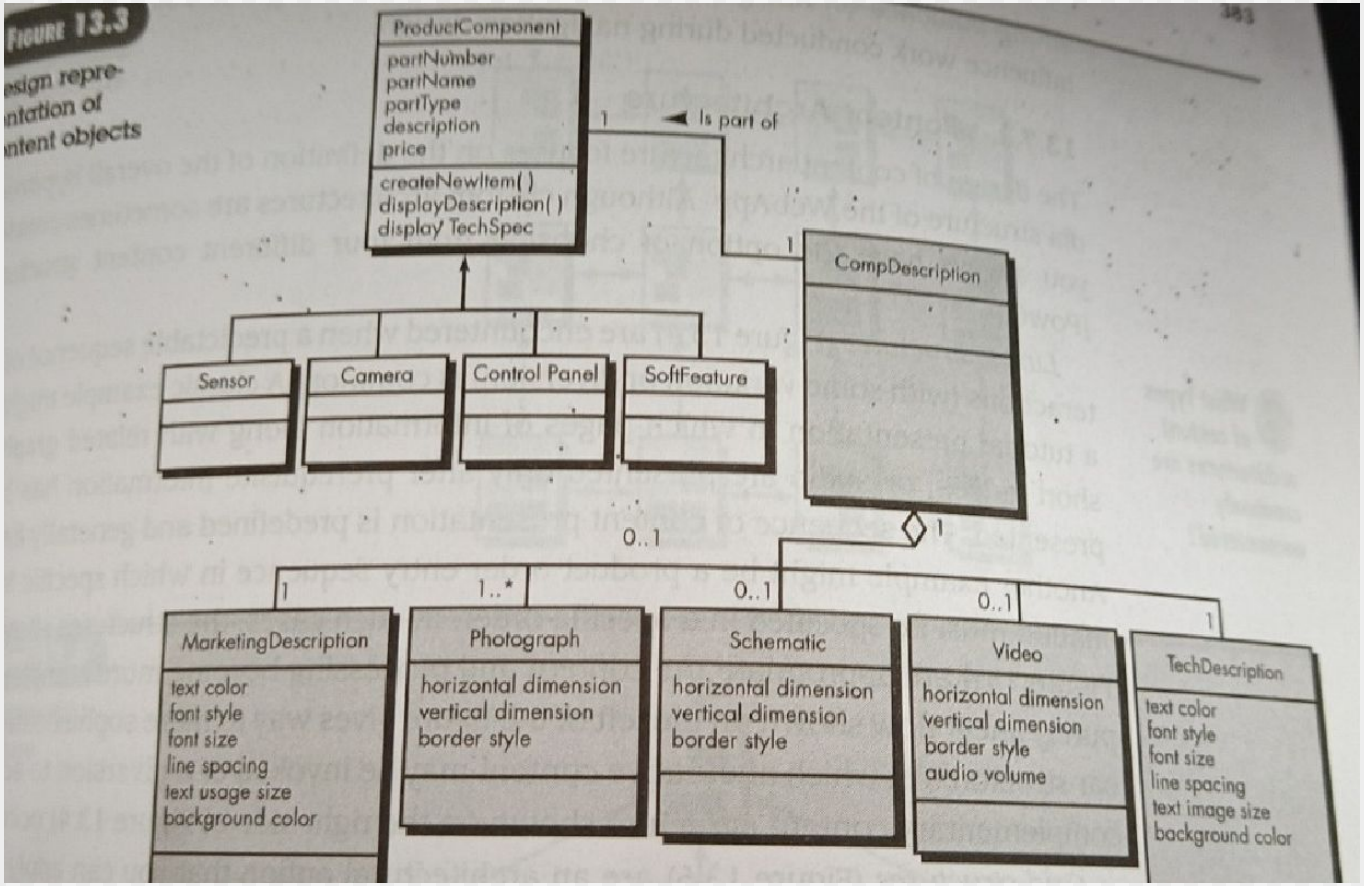


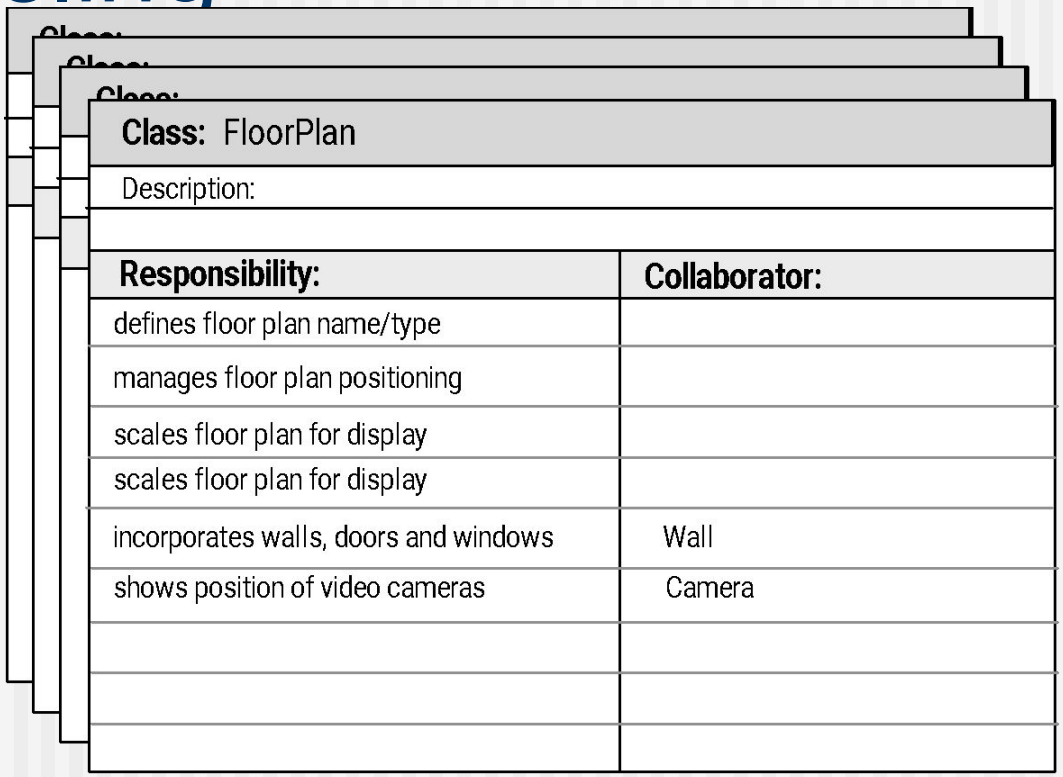
FIGURE 13.3
esign repre-
ntation of
ntent objects

CRC Models

- *Class-responsibility-collaborator (CRC)*

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.

CRC Modeling



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

Responsibilities

- System intelligence should be distributed across classes to best address the needs of the problem
- Each responsibility should be stated as generally as possible
- Responsibilities should be shared among related classes, when appropriate.

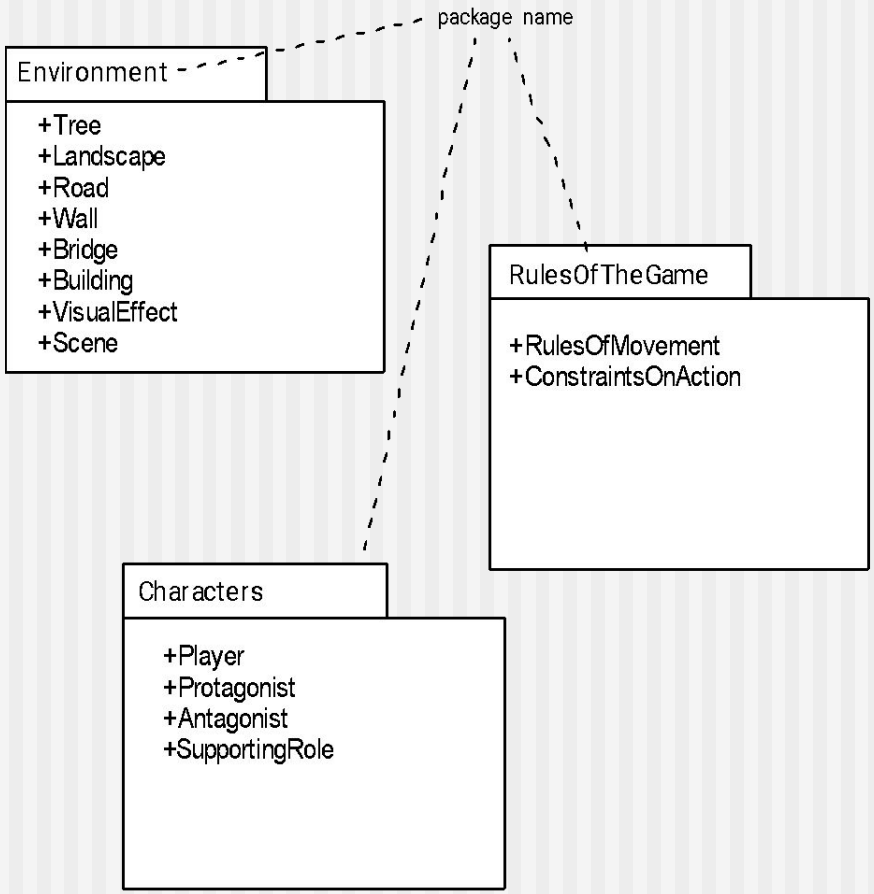
Collaborations

- a class can collaborate with other classes.
- Collaborations identify relationships between classes
- three different generic relationships between classes
  - the *is-part-of* relationship
  - the *has-knowledge-of* relationship
  - the *depends-upon* relationship

Analysis Packages

- Various elements of the analysis model (e.g., use-cases, analysis classes) are categorized in a manner that packages them as a grouping
- The plus sign preceding the analysis class name in each package indicates that the classes have public visibility and are therefore accessible from other packages.
- A minus sign indicates that an element is hidden from all other packages
- # symbol indicates that an element is accessible only to packages contained within a given package.

Analysis Packages



Behavioral Modeling

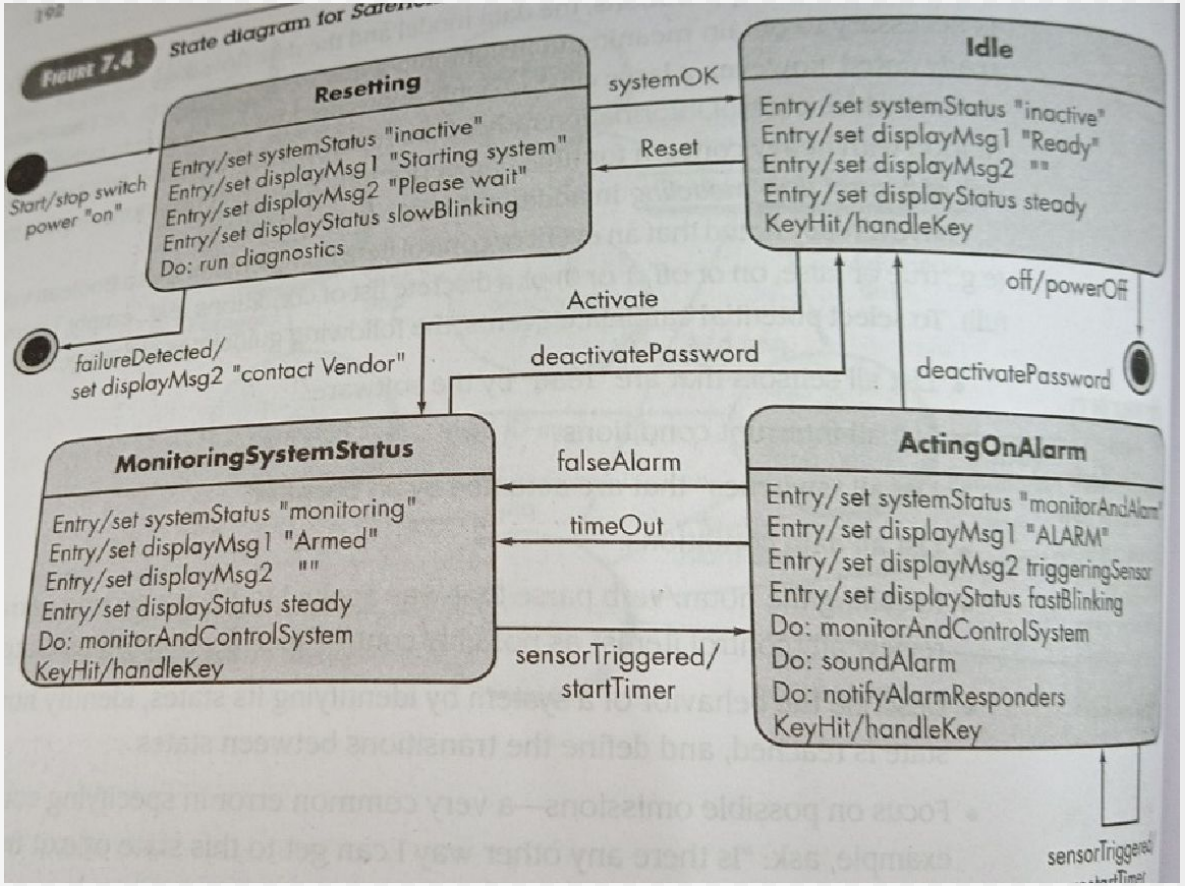
- The behavioral model indicates how software will respond to external events. To create the model, the analyst must perform the following steps:
  - Evaluate all use-cases to fully understand the sequence of interaction within the system.
  - Identify events that drive the interaction sequence and understand how these events relate to specific objects.
  - Create a sequence for each use-case.
  - Build a state diagram for the system.
  - Review the behavioral model to verify accuracy and consistency.

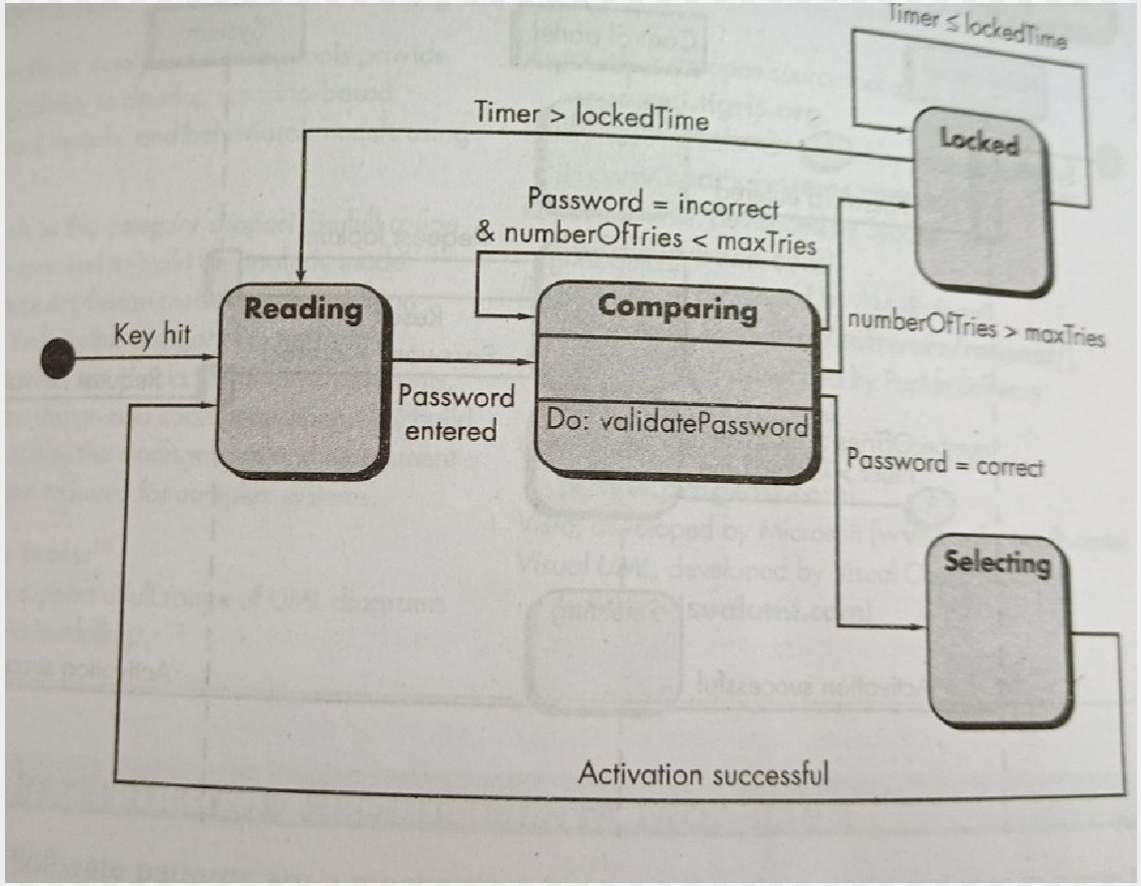
State Representations

- In the context of behavioral modeling, two different characterizations of states must be considered:
  - the state of each class as the system performs its function and
  - the state of the system as observed from the outside as the system performs its function

The States of a System

- state—a set of observable circum-stances that characterizes the behavior of a system at a given time
- state transition—the movement from one state to another
- event—an occurrence that causes the system to exhibit some predictable form of behavior
- action—process that occurs as a consequence of making a transition




Timer ≤ lockedTime
Timer > lockedTime
Locked
Password = incorrect
& numberOfTries < maxTries
Reading
Comparing
Key hit
numberOfTries > maxTries
Password
entered
Do: validatePassword
Password = correct
Selecting
Activation successful

Sequence diagram

- represents the behavioral aspects of a system. Sequence diagram shows the interactions between the objects by means of passing messages from one object to another with respect to time in a system.

Elements in sequence diagram

- Sequence diagram contains the objects of a system and their life-line bar and the messages passing between them.
- Object : Objects appear at the top portion of sequence diagram. Object is shown in a rectangle box.
- Life-line bar: A down-ward vertical line from object-box is shown as the life-line of the object. A rectangle bar on life-line indicates that it is active at that point of time.
- Messages: Messages are shown as an arrow from the life-line of sender object to the life-line of receiver object and labeled with the message name.

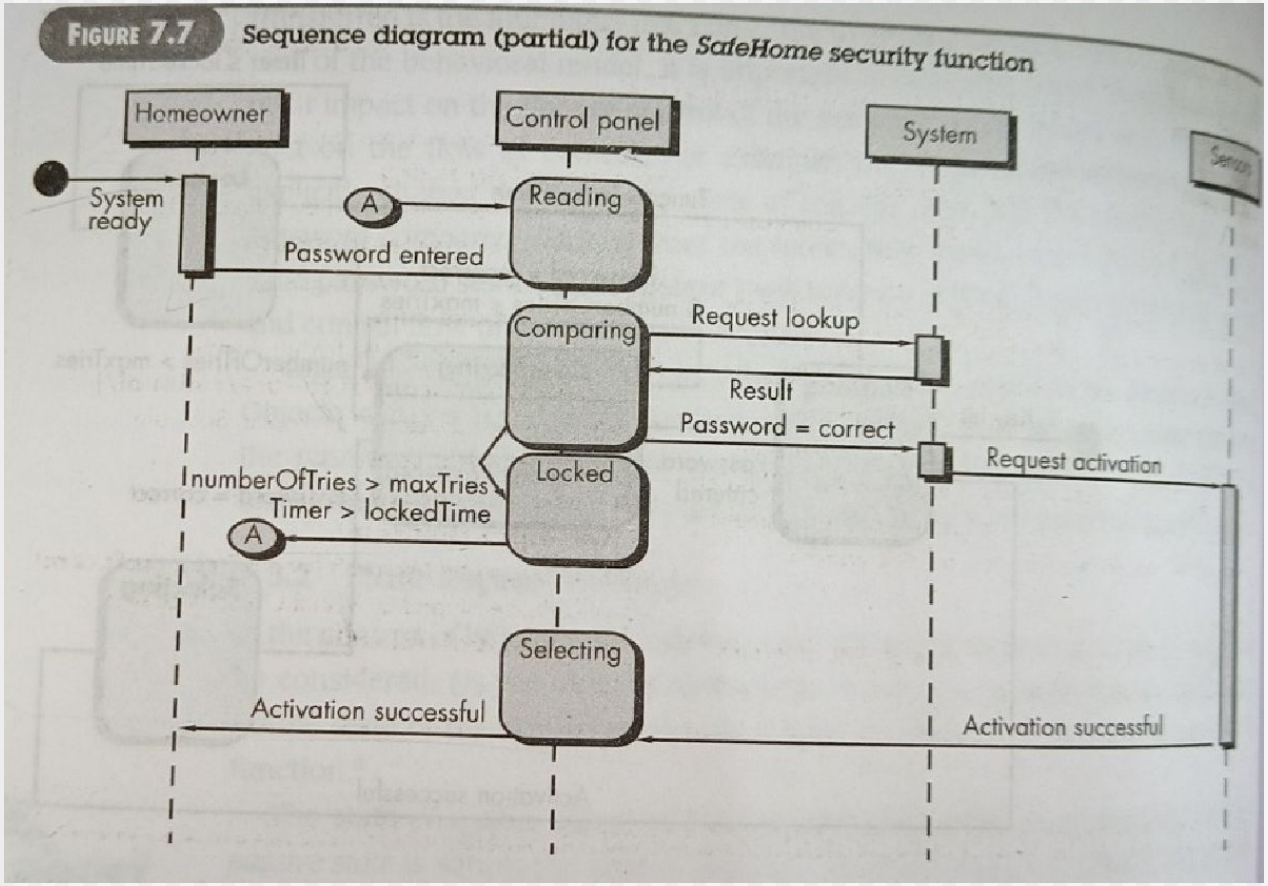


FIGURE 7.7 Sequence diagram (partial) for the *SafeHome* security function

Exercise

- Define Requirement Analysis . Explain in brief Rule of Thumb for requirement Analysis
- Explain with figure elements of requirement analysis