Introduction of Software Engineering

Chapter 6:

Requirements Modeling

VŨ THỊ TRÀ

CONTENTS

Understanding Requirements

- Requirements engineering
- Establishing the groundwork
- Eliciting requirements
- Developing use cases
- Building the analysis model

Requirements Engineering

- Requirements engineering is a major software engineering action that begins during the communication activity and continues into the modeling activity.
- It must be adapted to the process, the project, the products and the people doing the work.
- Requirement engineering builds a bridge to design and construction.
  - ✓ Where business need is defined
  - ✓ User scenario are described
  - ✓ Functions and features are delineated
  - ✓ Project constraints are identified

Requirements Engineering

- *“The hardest single part of building a software system is deciding what to build. No part of the work so cripple the resulting system if done wrong. No other part is more difficult to rectify later”.*

  <Fred Brooks>

- *Expect to do a bit design during requirements work and a bit of requirements work during design*

Understanding Requirements

- Requirement engineering encompasses 7 tasks:
  1. Inception
  2. Elimination
  3. Elaboration
  4. Negotiation
  5. Specification
  6. Validation
  7. Management

7 task:: Inception

- Establish a basic understanding of the problem:
  - ✓ Define a business case for the idea
  - ✓ Try to identify the breath and depth of the market
  - ✓ A rough feasibility analysis
  - ✓ Identify a working description of project scope
- Establish the nature of the solution that is desired
- Establish the effectiveness of preliminary communication and collaboration between the stakeholders and the software team

7 task:: Elimination

- What are the objective for the system
- What is to be accomplished
- How the product fits into the need of business
- How the product is to be used on a day-to-day basis

  → *not simple and very hard*

- Establish the business goals
  - ✓ Prioritization mechanism
  - ✓ Design rationale for a potential architecture

7 task:: Elaboration

- Identify software function, behavior, and information
- Describe how end-user will interact with the system
- Each user scenario is parsed to extract analysis classes – business domain entities
- Define the attributes of each analysis class
- Identify the relationship and collaboration between classes
- Produce a variety of supplementary diagrams

7 task:: Negotiation

- Reconcile the requirements conflicts
- Prioritize requirements
- Access the cost and risk

7 task:: Specification

- Specify different things to different peoples
- Write specification document
- Collection of usage scenarios, a prototype, or any combination of them
- Reside within well-understood technical environments

→ the formality and format of a specification varies with the size and the complexity of the software to be built.

7 task:: Validation

- Exam specification documents to ensure all software requirements have been stated unambiguously
- Detect and correct the inconsistences, omission, errors
- Conform to the standards established for the process, the project, and the product
- Require the technique review (software engineers, customers, users, other stakeholders)
- Apply qualitative and quantitative assessment
- Mitigate risks and guarantee the consistency

7 task:: Validation

- Mitigate the risks
- Verify requirement by using qualitative and quantitative assessment
- Qualitative technique (user survey of check list)
- Guarantee the consistency

7 task:: Management

- Identify, control, track requirements and changes to requirements

Establish the groundwork

1. Identify stakeholders
2. Recognize multiple viewpoints
3. Work toward collaboration
4. Ask first questions
5. Nonfunctional requirements
6. Traceability

Eliciting requirements

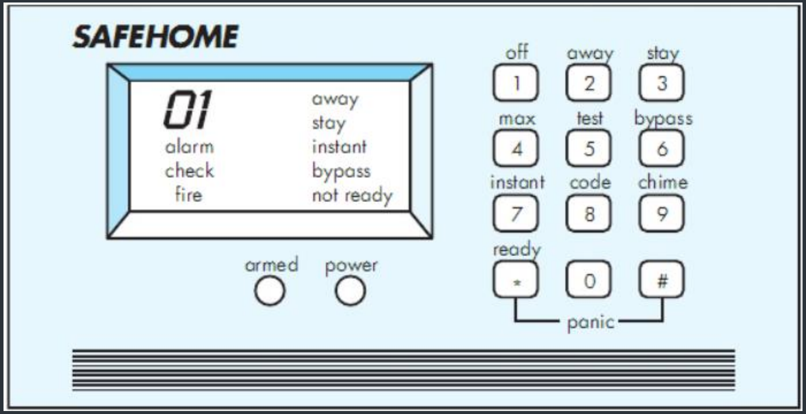
1. Collaborative requirements gathering
2. Quality function deployment (QFD)
3. Usage scenarios
4. Elicitation work products
5. Agile requirement elicitation
6. Service-oriented methods

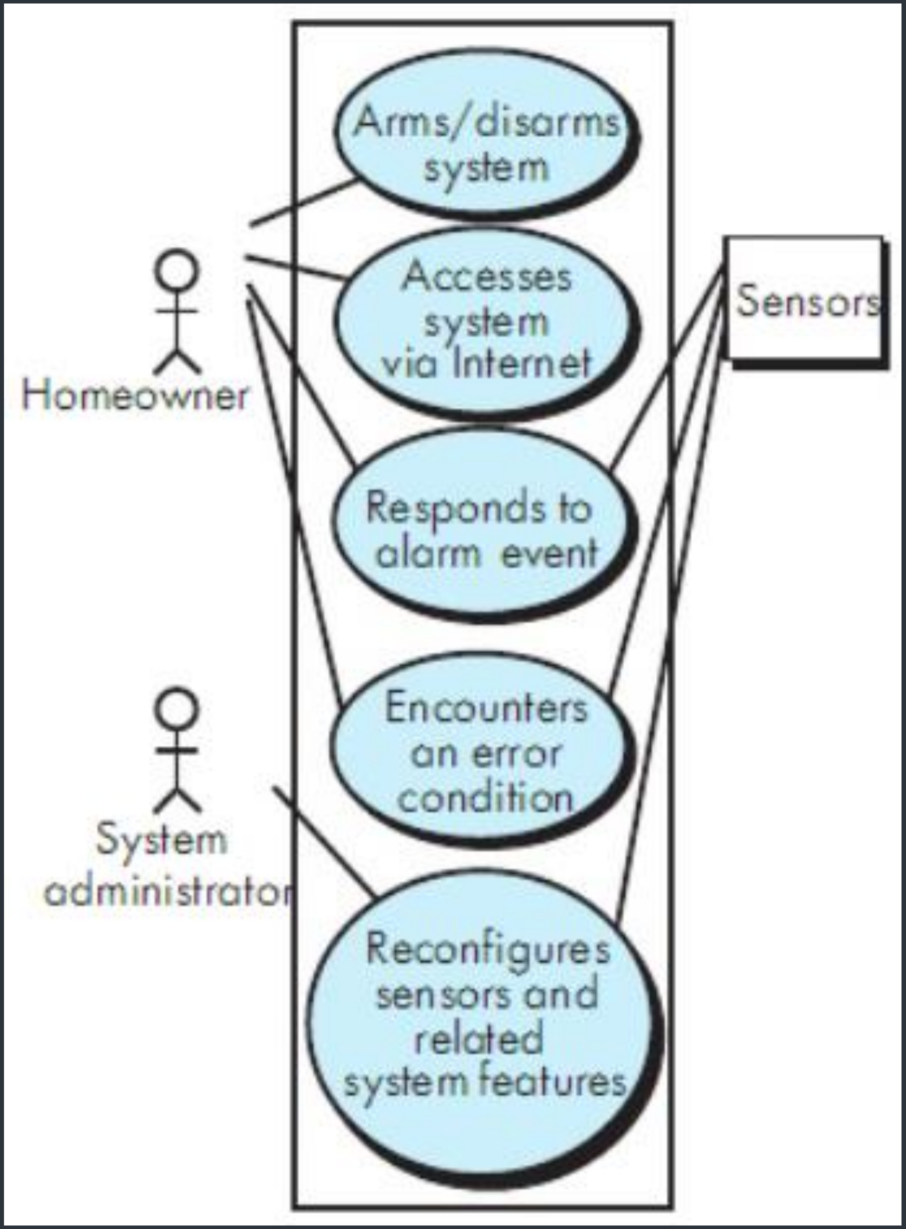
Developing Use Cases

Example of SafeHome control panel (p. 151-152)

1. User case
2. Primary actor
3. Goal in context
4. Preconditions
5. Trigger
6. Scenario
7. Exceptions
8. Priority
9. When available
10. Frequency of use
11. Channel to actors
12. Secondary actors
13. Channel to secondary actors
14. Open Issues

UML use case diagram
for SafeHome security function

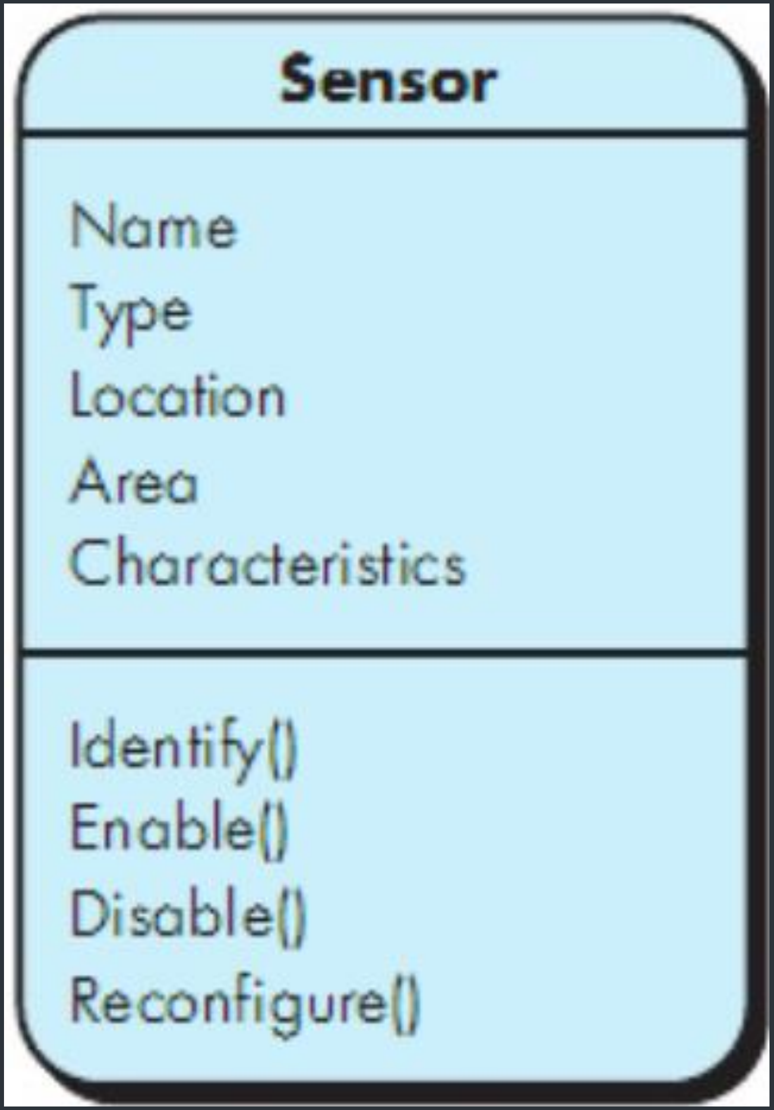



Homeowner
System administrator
Arms/disarms system
Accesses system via Internet
Responds to alarm event
Encounters an error condition
Reconfigures sensors and related system features
Sensors

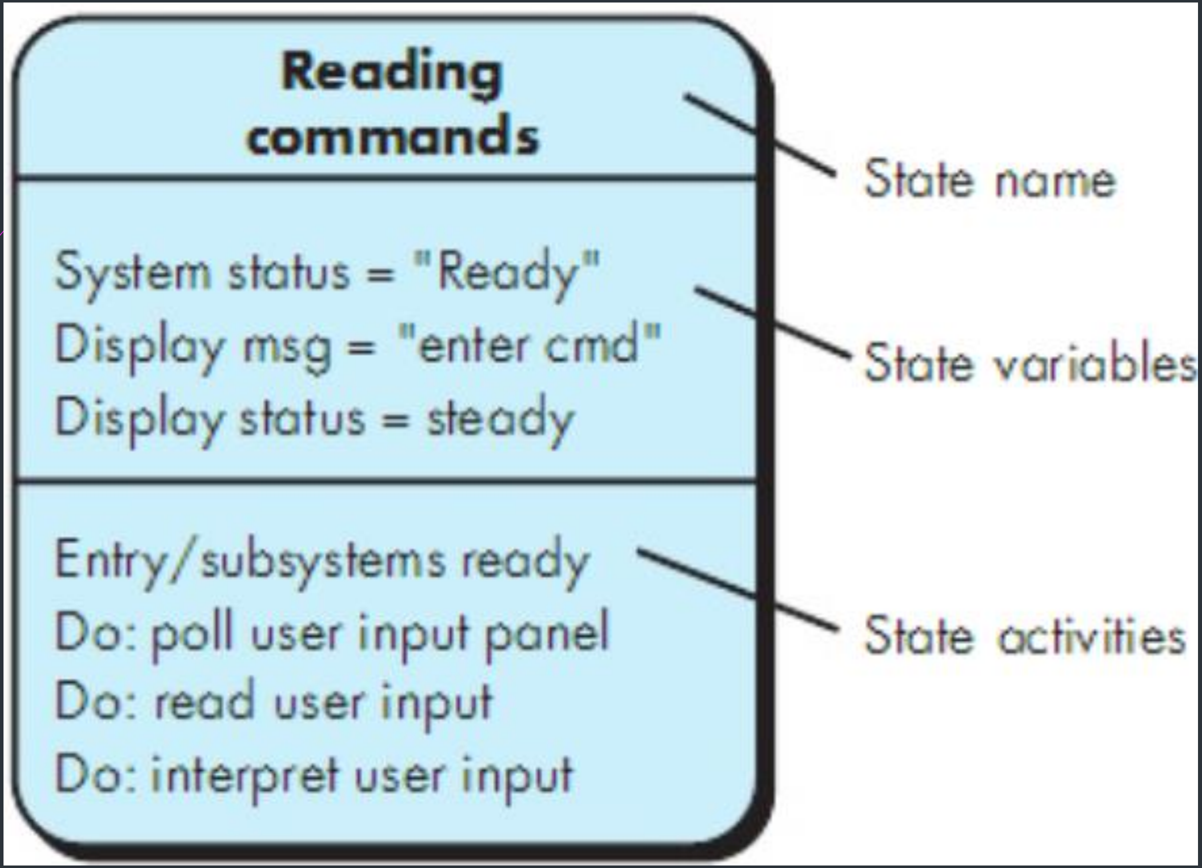
Building The Analysis Model

- Elements of the analysis model
  - ✓ Scenario-based elements
  - ✓ Class-based elements: a set of objects, attributes, common behaviors (Ex: a Sensor class)
  - ✓ Behavioral elements

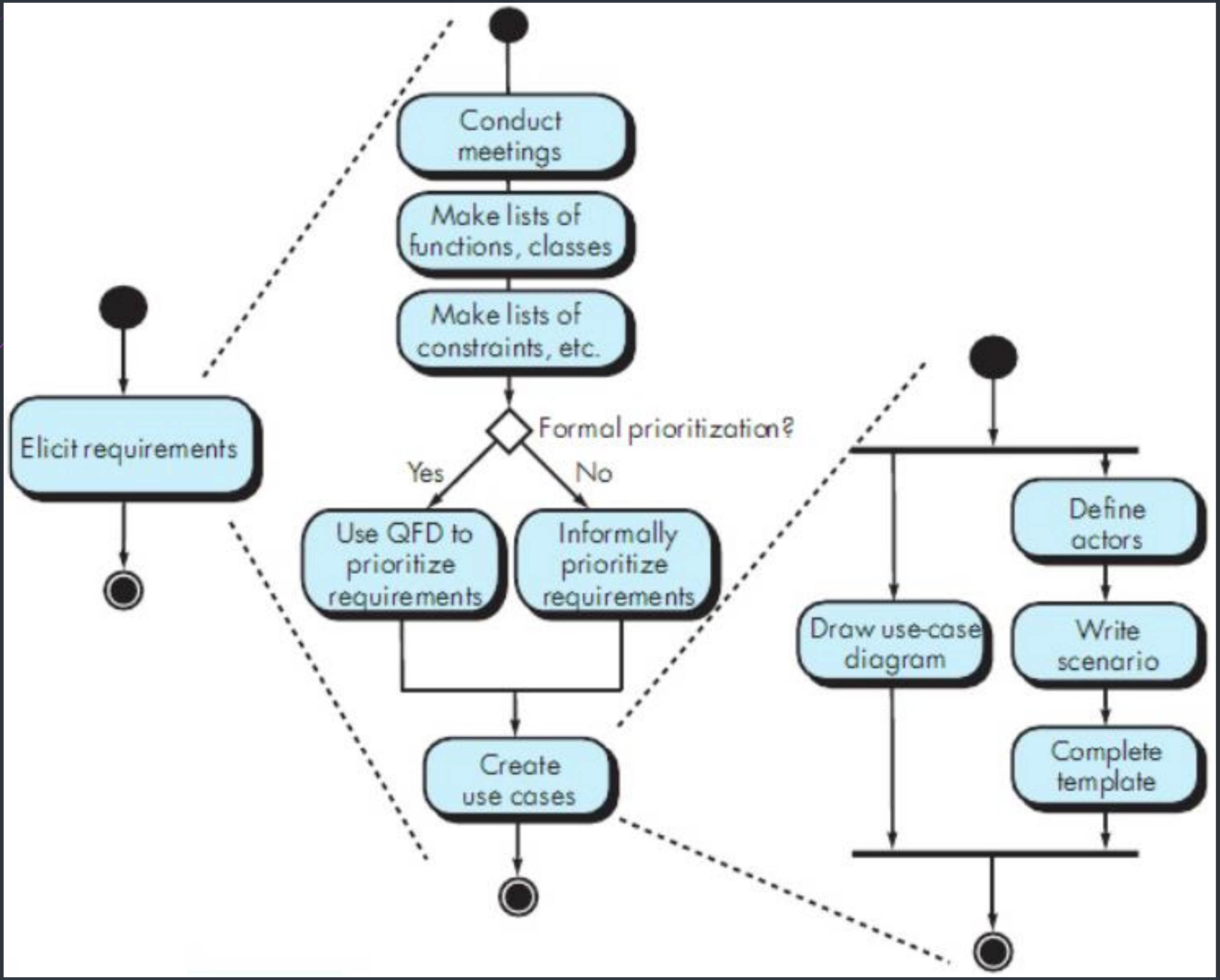
Sensor Class Diagram



UML state diagram

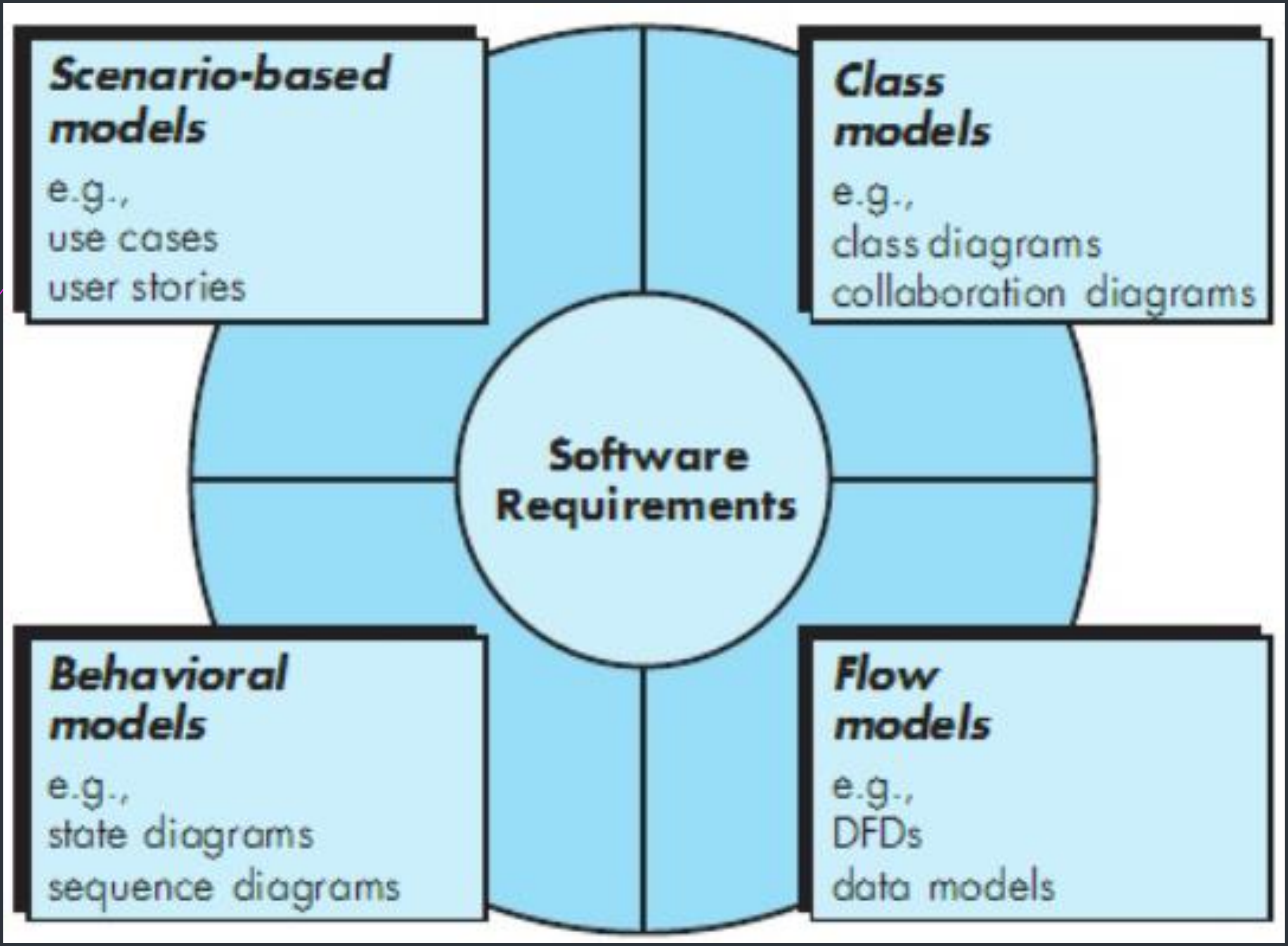


UML activity diagram



CONTENTS

Elements of the analysis model



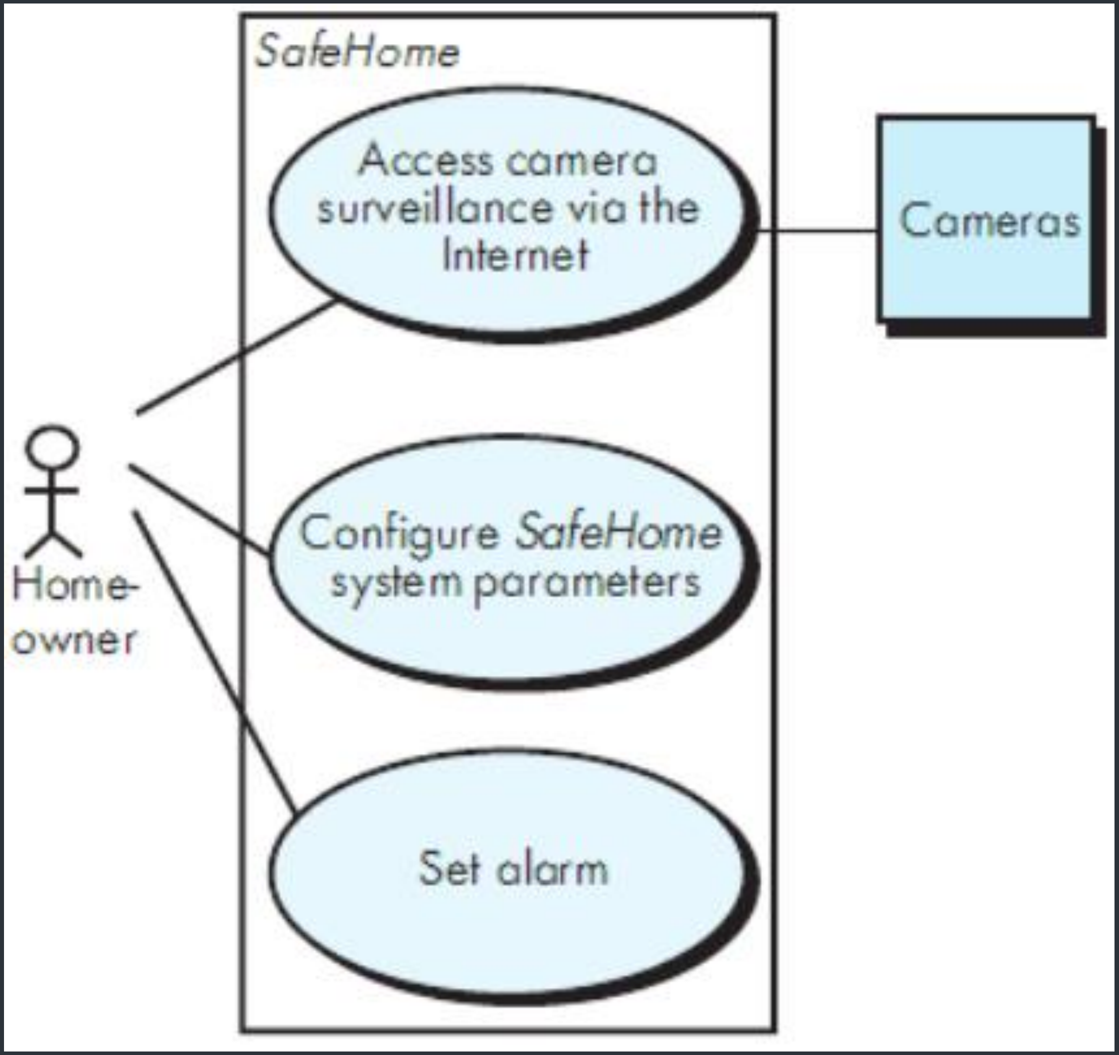
Requirements Modeling

- Scenario-based methods
- Class-based methods
- Behavior-based methods

Scenario-based methods

- Create a preliminary use case
- Refine a preliminary use case
- Writing a formal use case

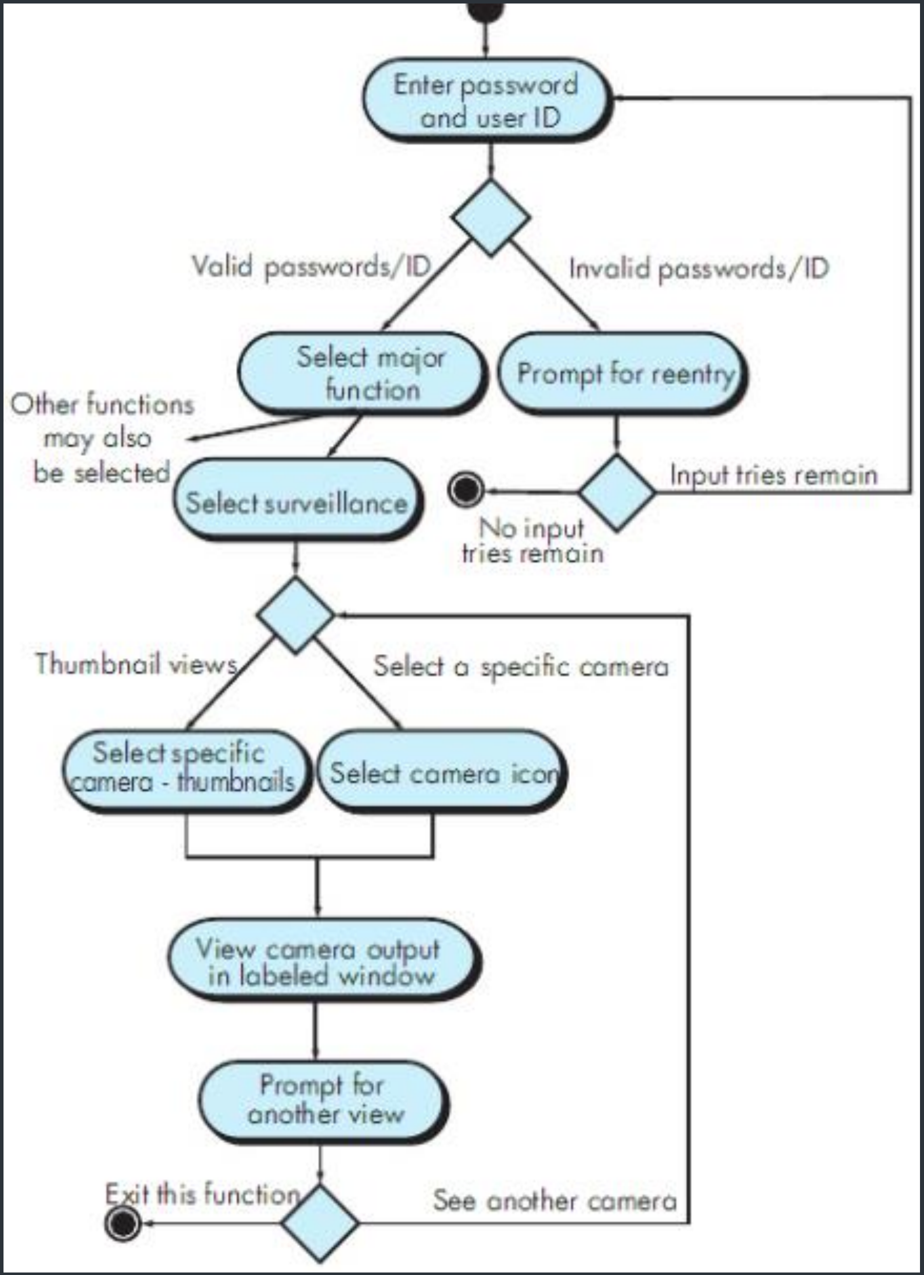
Preliminary use case diagram


SafeHome
Access camera surveillance via the Internet
Cameras
Configure SafeHome system parameters
Home-owner
Set alarm

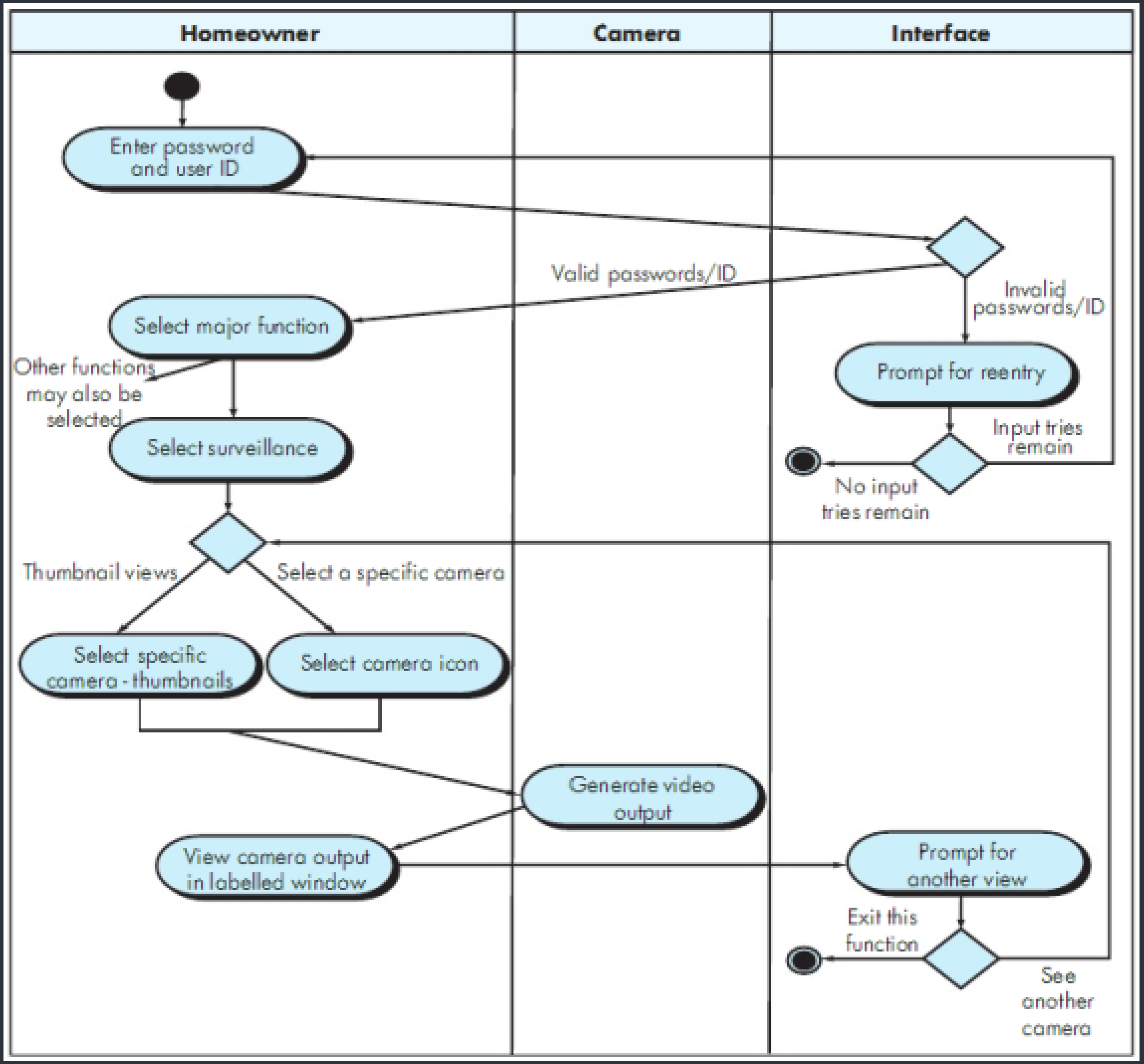
Functions performed by Homeower actor

- Select camera to view
- Request thumbnails from all cameras
- Display camera views in PC window
- Control pan and zoom for a specific camera
- Selectively record camera output
- Replay camera output
- Access camera surveillance via the Internet

Activity diagram for Access camera surveillance via the Internet – display camera views (ACS-DCV) function



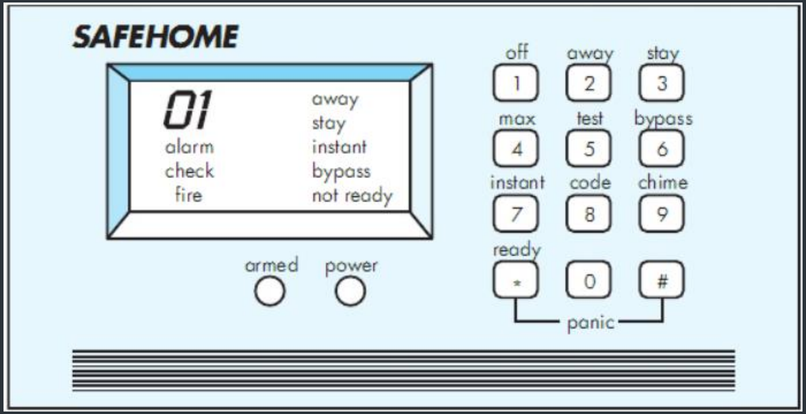
Swimlane diagram

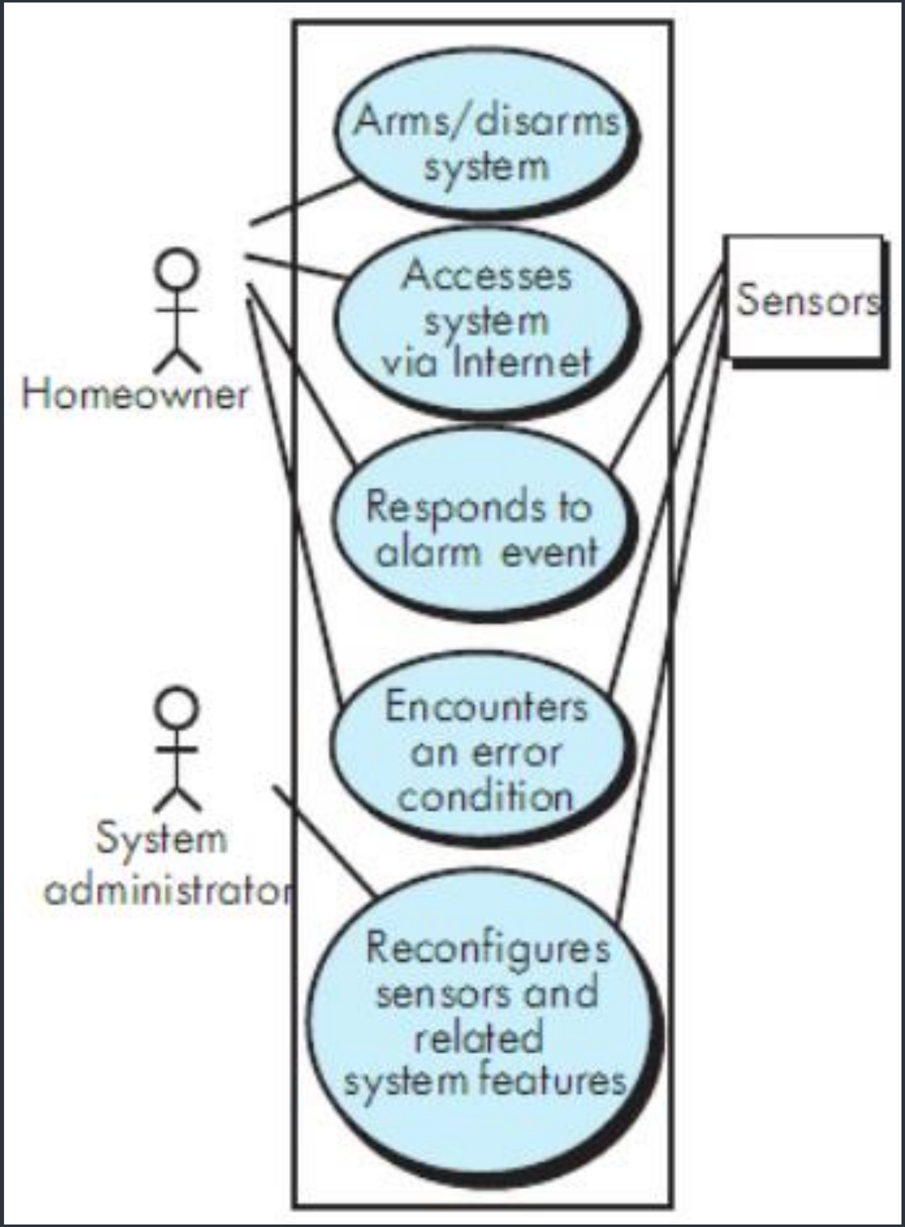


Requirements Modeling

- Scenario-based methods
- Class-based methods
- Behavior-based methods

UML use case diagram
for SafeHome 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

Class-based methods

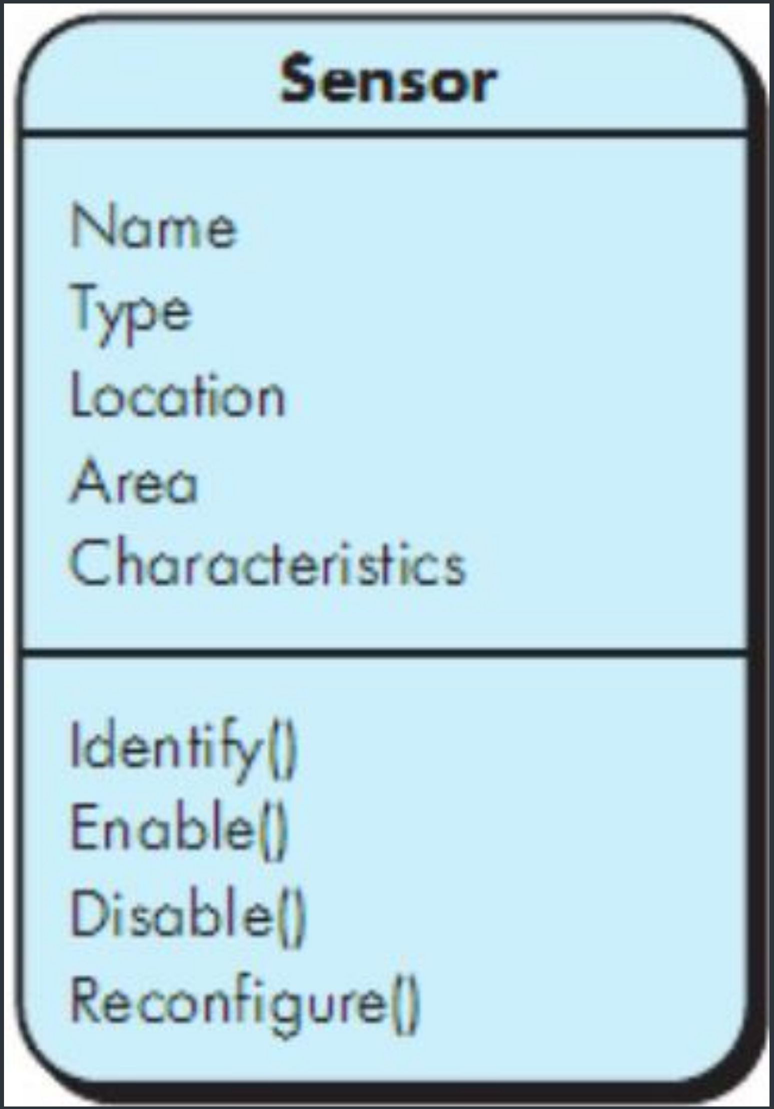
- How analysis class?

The SafeHome security function enables the Homeower to *configure* security system when it *is installed*, monitors all sensors *connected* to the security system, and *interacts* with the Homeower through the Internet, a PC or a control panel.

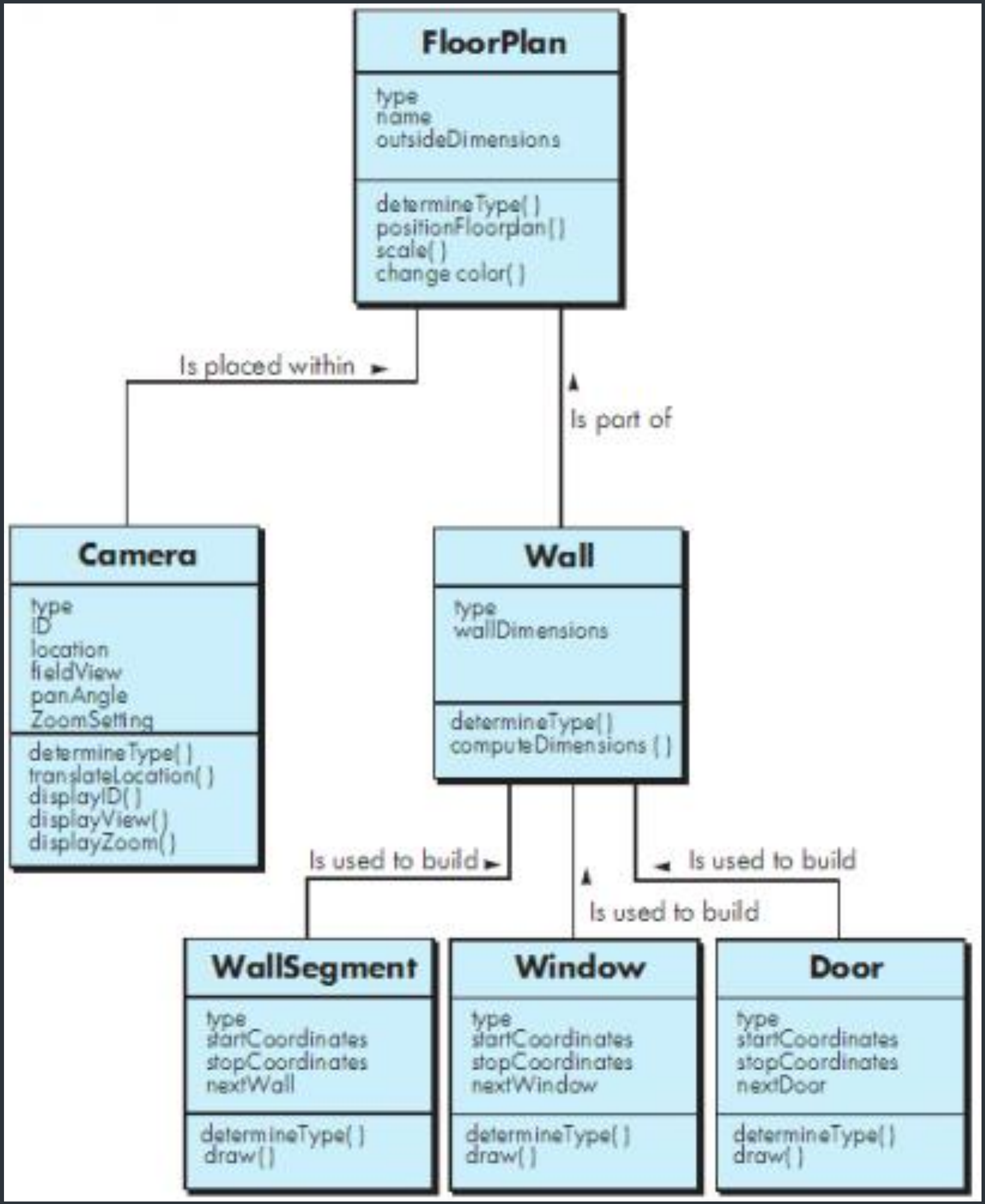
Class-based methods

- Potential classes:
  - ✓ Homeowner
  - ✓ Sensor
  - ✓ Control Panel

Sensor Class Diagram



Class diagram for FloorPlan



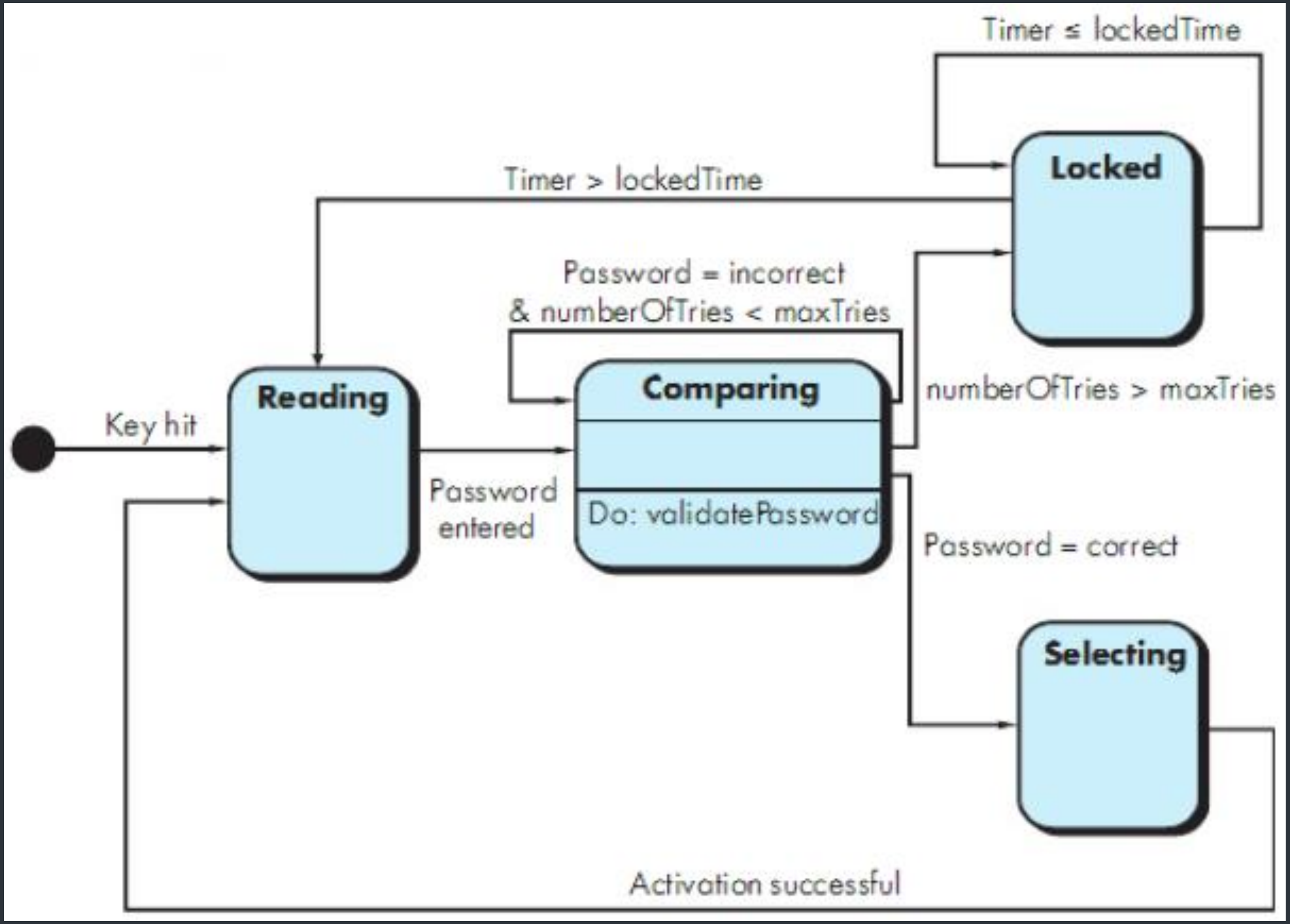
Requirements Modeling

- Scenario-based methods
- Class-based methods
- Behavior-based methods

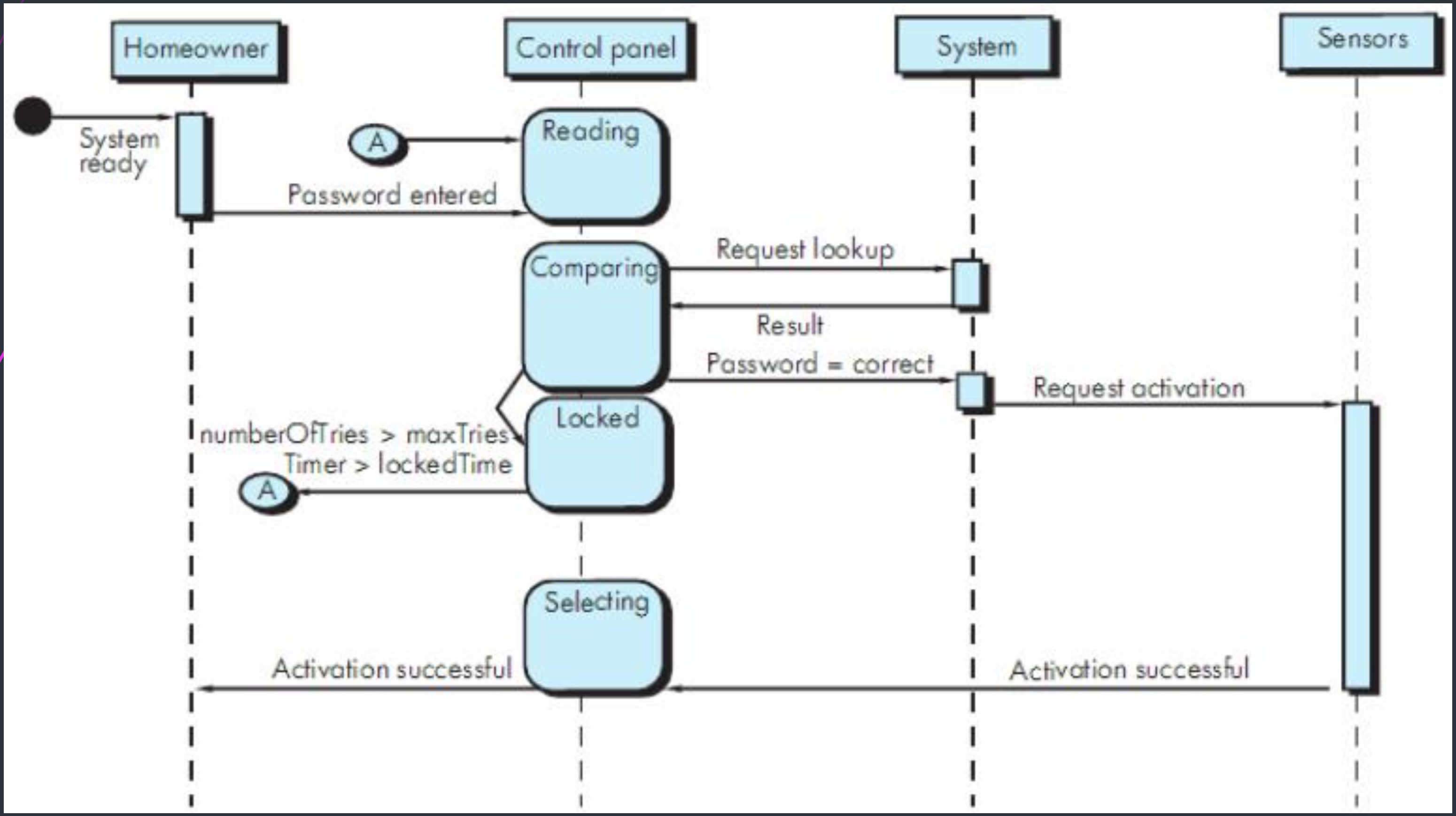
The use case for a portion of the SafeHome security functton

- The Homeowner used the keypad to key in a four-digit password. The password is compared with the valid password stored in the system. If the password is incorrected, the control panel will beep once and reset itself for additional input. If the password is corrected, the control panel waits further action.

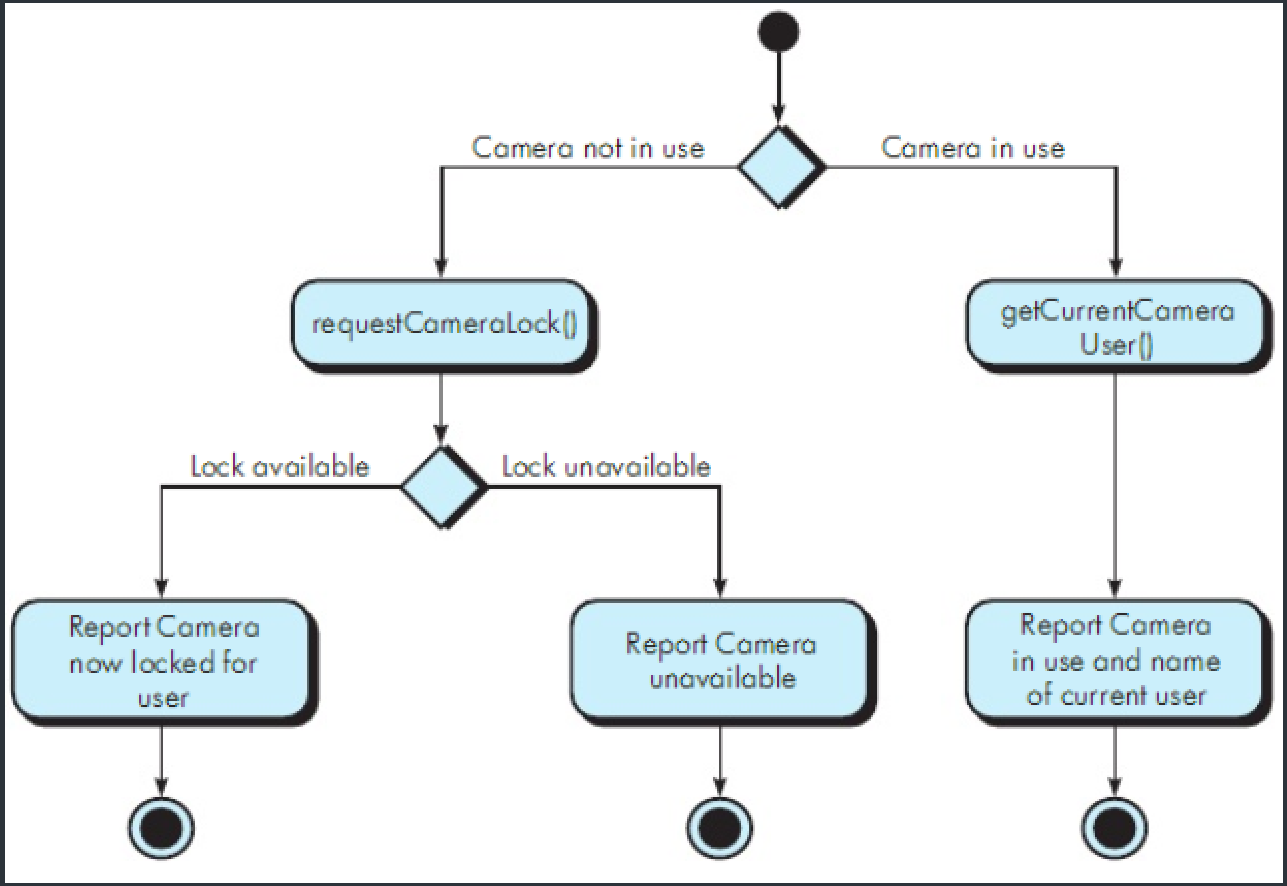
State diagram for the ControlPanel class



Sequence diagram (partial) for the SafeHome security function



Activity diagram for the takeControlOfCamera() operation


Camera not in use
Camera in use
requestCameraLock()
getCurrentCamera User()
Lock available
Lock unavailable
Report Camera now locked for user
Report Camera unavailable
Report Camera in use and name of current user