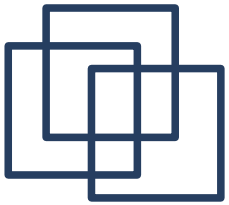
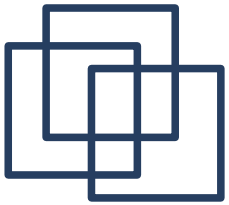


CMSC 128

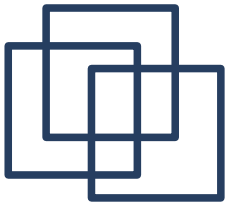
Introduction to Software Engineering
Second Semester AY 2007-2008

jachermocilla@uplb.edu.ph



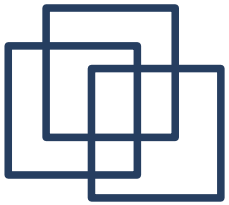
OO Analysis

- Define all classes (and relationships and behavior) that are relevant to the problem to be solved – develop the Object Model
- Major tasks
  - Communicate requirements
  - Identify classes
  - Specify class hierarchies
  - Represent object-object relationship
  - Model object behavior



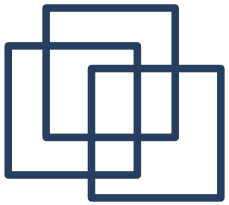
OOA vs SA

- Structured Analysis (SA)
  - Distinct input-process-output
  - Data is separate from process
  - Behavior tends to play a secondary role
  - Makes heavy use of functional decomposition
- OOA
  - Data and operations (processes) are taken(encapsulated) as one
  - Object behavior is important



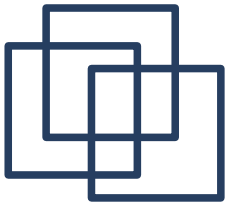
OOA Landscape

- Booch Method
  - Micro and macro development process
  - Micro consists of analysis tasks re-applied for each macro process
  - Outline
    - Identify classes and objects
    - Identify semantics of classes and objects
    - Identify relationships among classes and objects
    - Conduct series of refinements
    - Implement classes and objects



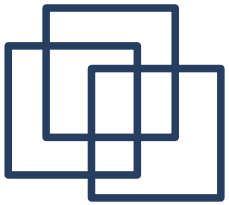
OOA Landscape

- Coad and Yourdon Method
  - Easiest to learn
  - Outline
    - Identify objects using 'what to look for' criteria
    - Define a generalization-specification structure
    - Define a whole-part structure
    - Identify subjects
    - Define attributes
    - Define services



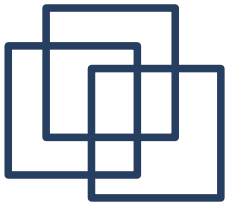
OOA Landscape

- Jacobson Method
  - Known as OOSE
  - Heavy use of use case – a description or scenario that depicts how the user interacts with the product or system
  - Outline
    - Identify system users and their responsibilities
    - Build a requirements model
    - Build analysis model



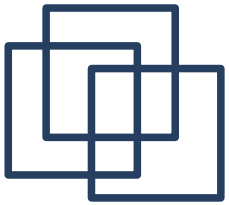
OOA Landscape

- Rambaugh Method
  - Object Modeling Technique (OMT)
  - Creates three models: object model(object,classes,hierarchies) , dynamic model (system behavior), functional model (DFD-like representation)
  - Outline
    - Develop statement of scope
    - Build models: object model, dynamic model, functional model



Generic Steps in OOA

1. Obtain customer requirements
   - Identify scenarios or *use cases*
   - Build a requirements model
2. Select classes and objects
3. Identify attributes and operations for each object
4. Define structures and hierarchies that organize classes
5. Build an object-relationship model
6. Build an object-behavior model
7. Review the OO analysis model against use cases



Domain Analysis

- Conducted when an organization wants to create a library of reusable classes (components) that will be broadly applicable to an entire category of applications
- Identification, analysis, and specification of common requirements from a specific application domain

Domain Analysis

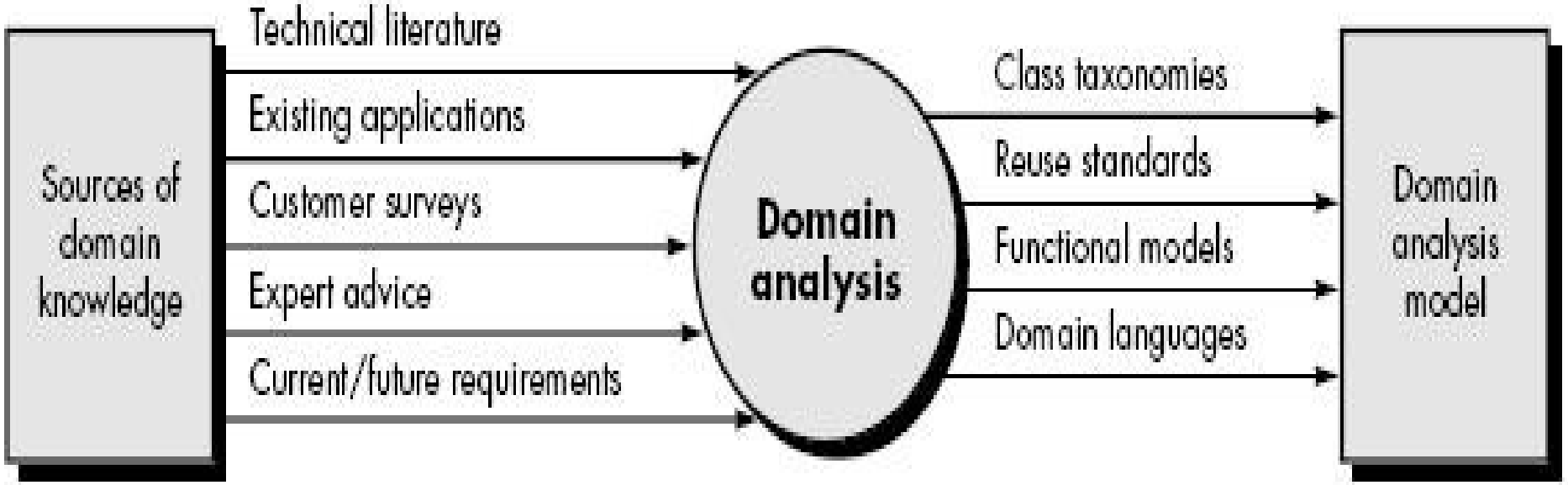
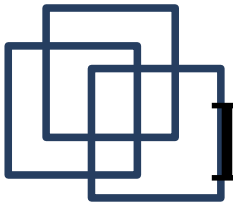
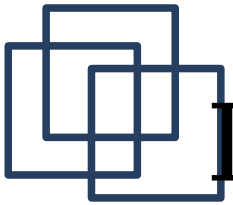


FIGURE 21.1 Input and output for domain analysis



Domain Analysis Activities

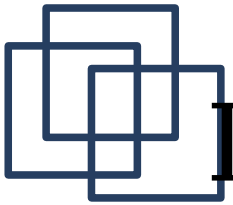
1. Define the domain to be investigated
   - Isolate business area, system type, or product category of interest
   - OO and Non-OO items must be extracted
2. Categorize the items extracted from the domain
   - Classification scheme is proposed and naming conventions for each item are defined
3. Collect representative samples of applications in the domain



Domain Analysis Activities

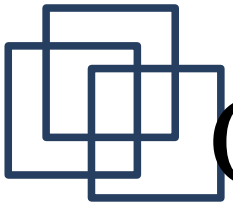
4.Analyze each application in the sample

- Identify candidate reusable objects
- Indicate reasons why object is candidate for reuse
- Define adaptations to the object that they may also be reusable
- Estimate percentage of applications in the domain that might make reuse of the object
- Identify object by name and use SCM to control them



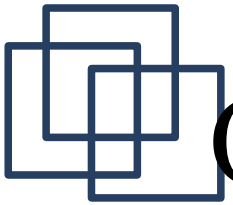
Domain Analysis Activities

5. Develop an analysis model for the objects
   - Will serve as basis for design and construction



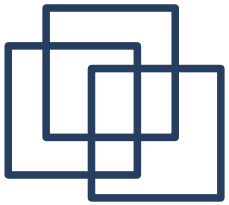
Components of OOA Model

- Static components
  - Structural in nature
  - Indicate characteristics that hold throughout the operational life of an application
- Dynamic components
  - Focus on control
  - Sensitive to timing and event processing
  - Define how one object interacts with other objects over time
  - State transitions



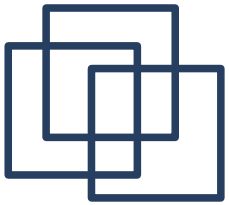
Components of OOA Model

- Identified components
  - Static view of semantic classes
  - Static view of attributes
  - Static view of relationships
  - Static view of behaviors
  - Dynamic view of communication
  - Dynamic view of control and time



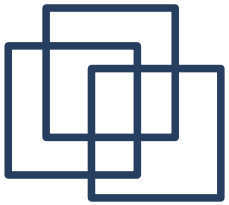
OOA Process

- OOA process begins with an understanding of the manner in which the system will be used
- Several techniques are available



Use Cases

- Scenarios in which the system will be used
- First, identify *actors* (people, device, etc)
  - Anything that communicates with the system or product and that is external to the system itself
  - Primary or secondary
- Describe the manner in which the actor interacts with the system
- Narrative decription

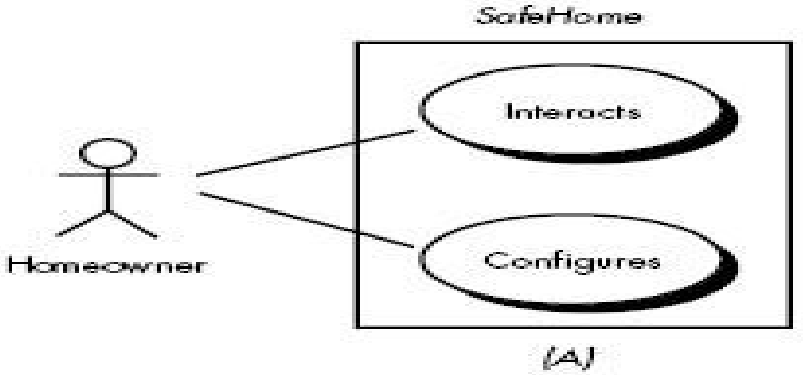


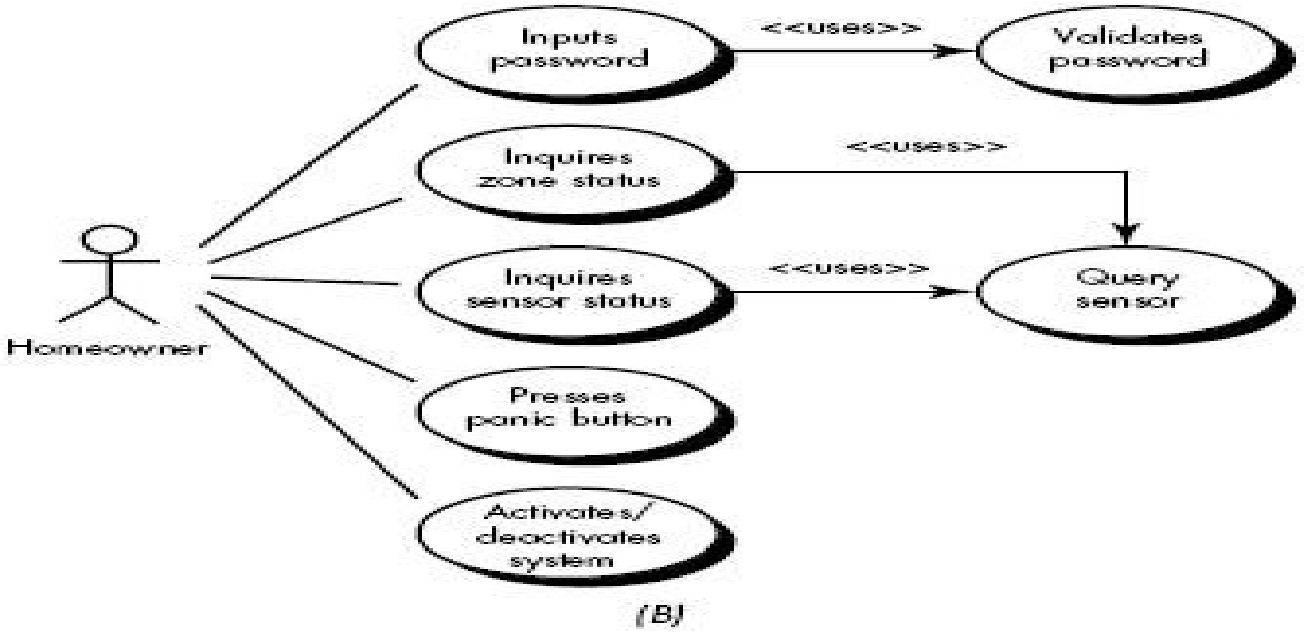
Use Cases

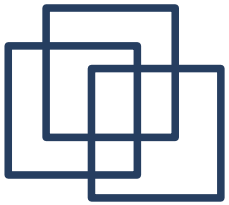
- Must answer the following
  - What are the main tasks performed by actor?
  - What system information will the actor acquire, produce, or change?
  - Will the actor have to inform the system about changes in the external environment?
  - What information does the actor desire from the system?
  - Does the actor wish to be informed about unexpected changes?

Use Cases

FIGURE 21.2 (A) High-level use-case diagram, (B) elaborated use-case diagram







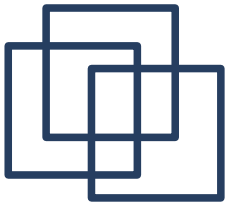
CRC

- Class-Responsibility-Collaborator
- Collection of index cards which is used to develop an organized representation of classes, responsibilities, and collaborators

CRC

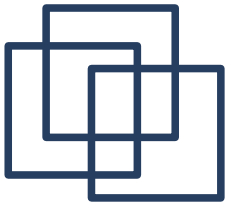
FIGURE 21.3
A CRC model index card

Class name:	
Class type: (e.g., device, property, role, event)	
Class characteristic: (e.g., tangible, atomic, concurrent)	
responsibilities:	collaborations:



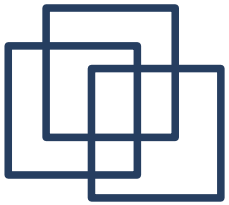
CRC

- Identifying Classes
  - Nouns in the grammatical parse
  - Types
    - Device classes – external entities (sensors)
    - Property classes – represents an important property of the problem environment (credit rating)
    - Interaction classes – model interactions that occur among objects (purchase or a license)



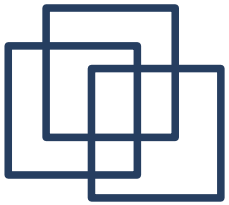
CRC

- Identifying Classes
  - Categories based on characteristics
    - Tangibility – does the class represent a tangible thing
    - Inclusiveness – is the class atomic or aggregate
    - Sequentiality – is the class concurrent or sequential
    - Persistence – is the class transient, temporary, permanent
    - Integrity – is the class corruptible or guarded



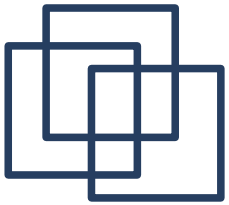
CRC

- Define Responsibilities
  - Attributes and operations
  - Five guidelines
    1. System intelligence should be evenly distributed
       - No class should have a long list of responsibilities
    2. Each responsibility should be stated as generally as possible
       - Must reside high in the hierarchy
    3. Properly encapsulated
    4. Localize information about one thing in one class
    5. Share responsibilities among related classes



CRC

- Describe Collaborators
  - Collaboration represents *requests from a client to a server in fulfillment of a client responsibility*
  - Embodiment of a contract between the client and the server
  - Identify relationship between classes
  - If a class cannot perform a responsibility itself, it needs to interact with other classes



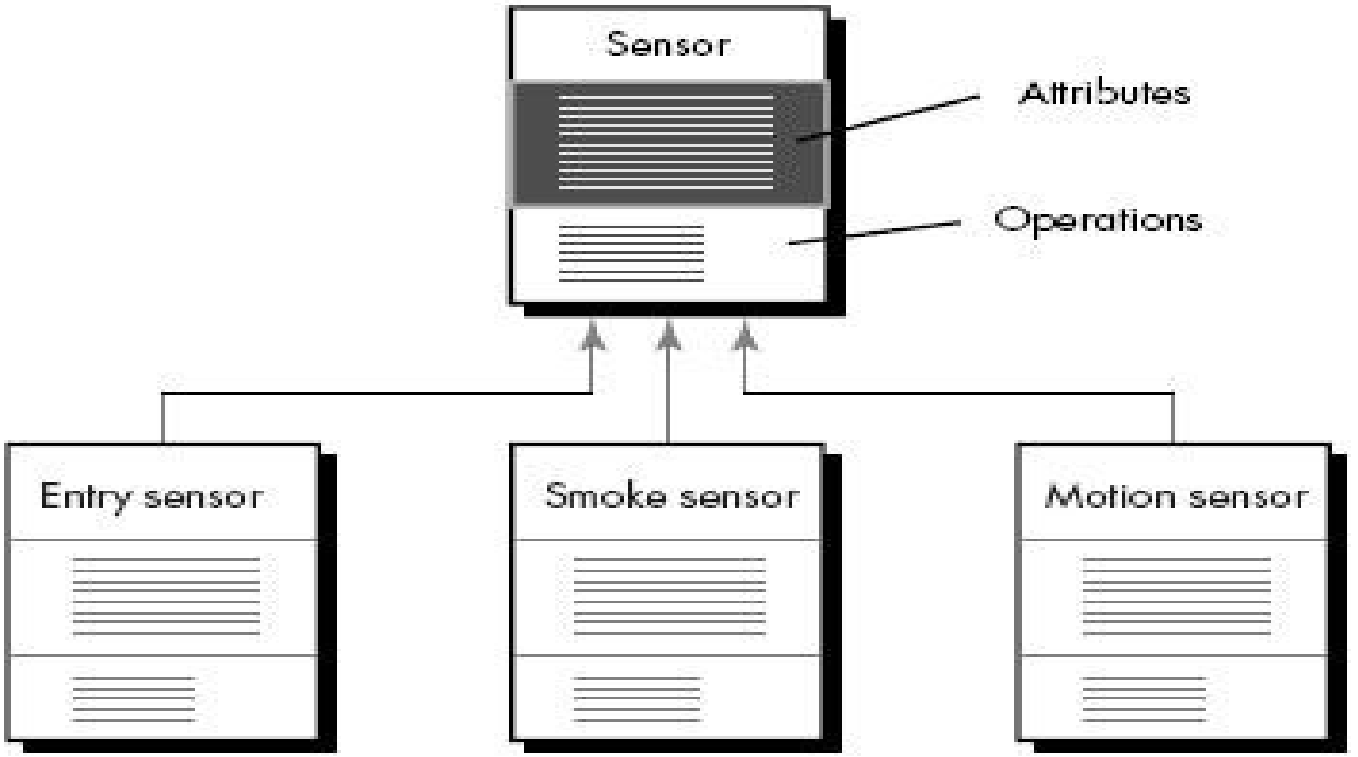
CRC

- Collaborators
  - Generic relationships between classes
    - *Is-part-of*
      - Aggregation
    - *Has-knowledge*
      - One class must acquire information from another
    - *Depends-upon*
      - Existence of an object of a class is dependent on another class
  - Collaborator class name recorded in CRC
    - How the responsibility will be realized
  - Walk through the CRC

Structures and Hierarchies

- Generalization-Specialization structure

FIGURE 21.4
Class diagram for generalization/specialization



Structures and Hierarchies

- Whole-Part Structure (composition)

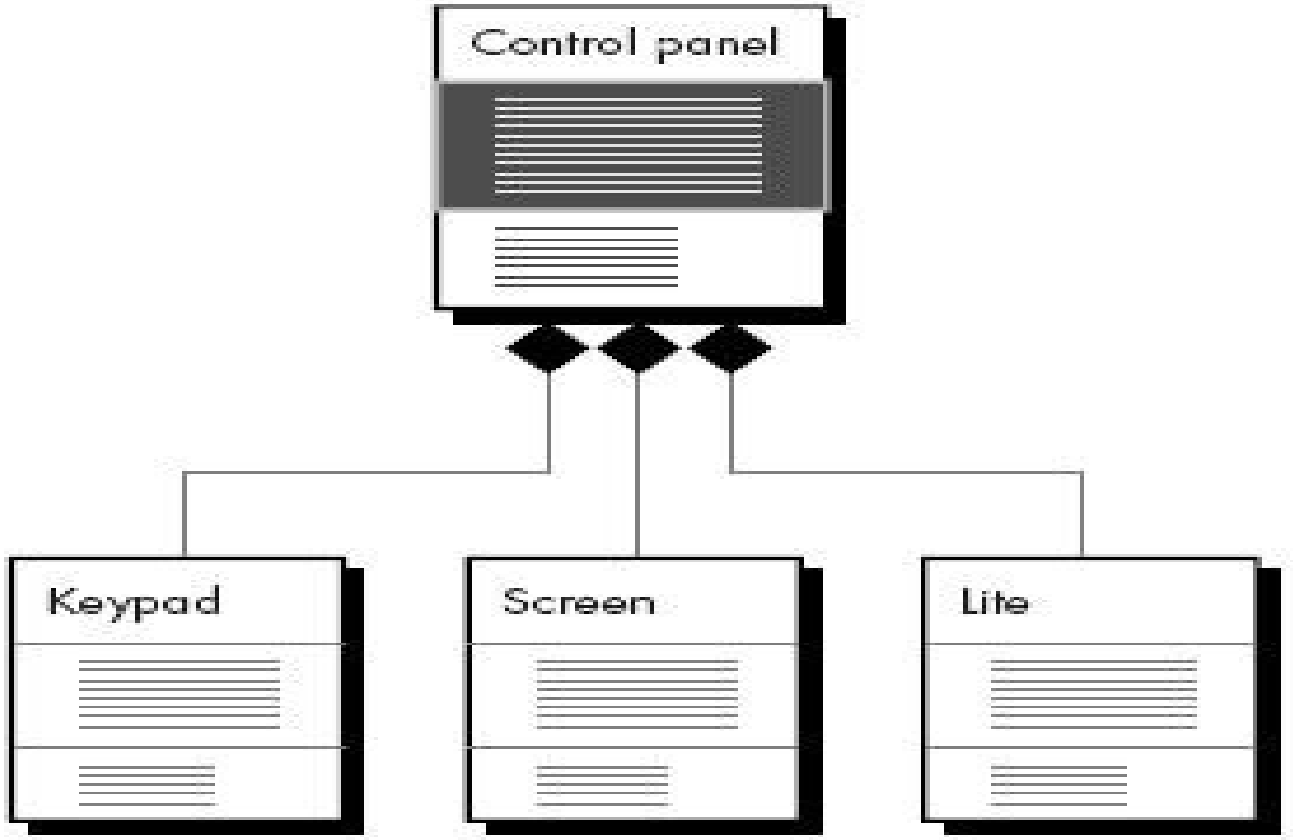
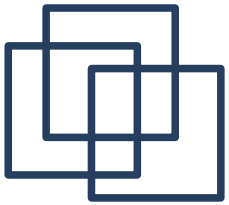


FIGURE 21.5
Class diagram for composite aggregates

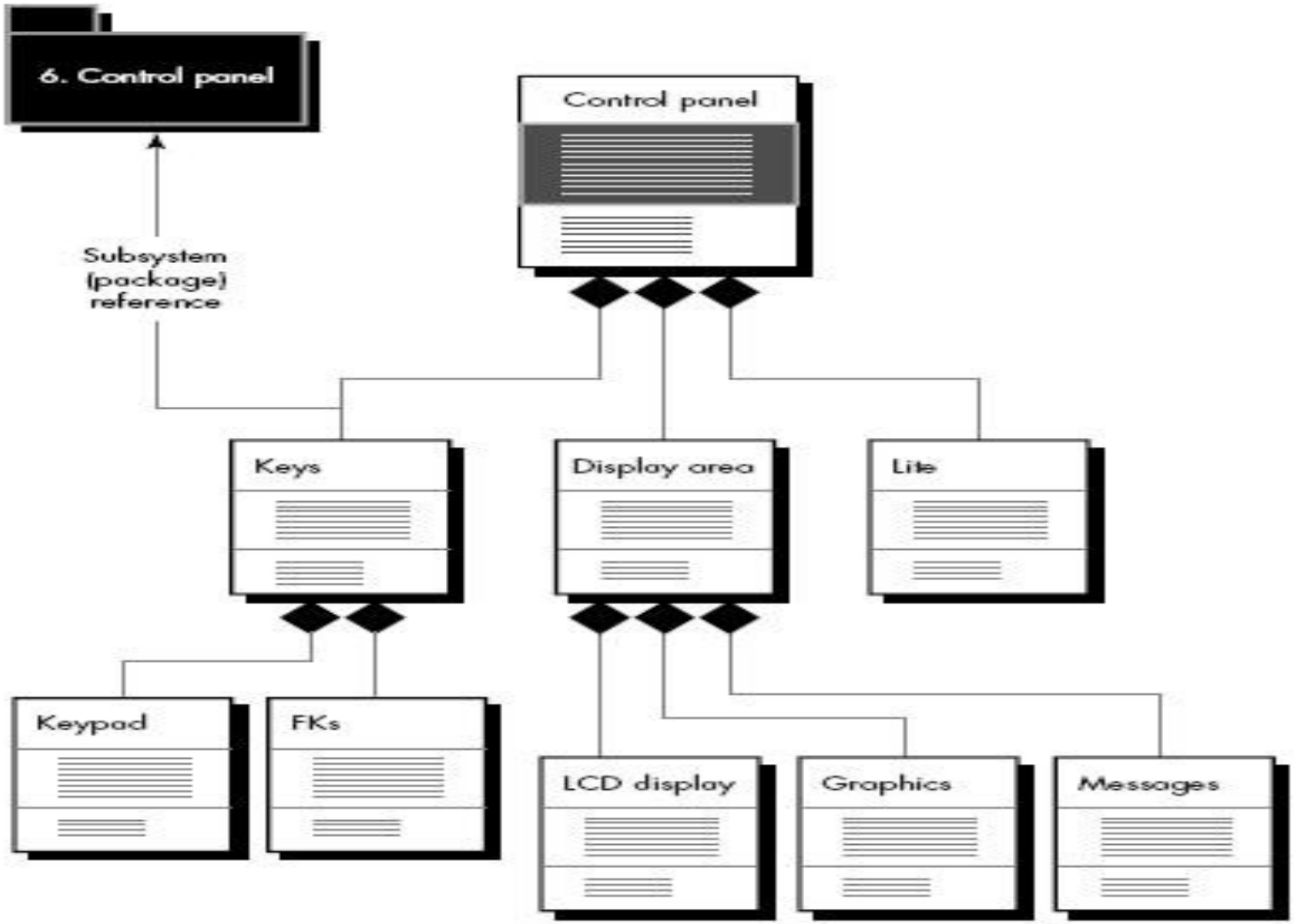


Subjects and Subsystems

- There may be many classes identified
- Concise representation is needed
- Subjects/Subsystems
  - Subset of all classes that collaborate among themselves to accomplish a set of cohesive responsibilities
  - Act as reference or pointer
  - Black-box
  - A separate card can be created

Subjects and Subsystems

FIGURE 21.6
Package (subsystem) reference



Subjects and Subsystems

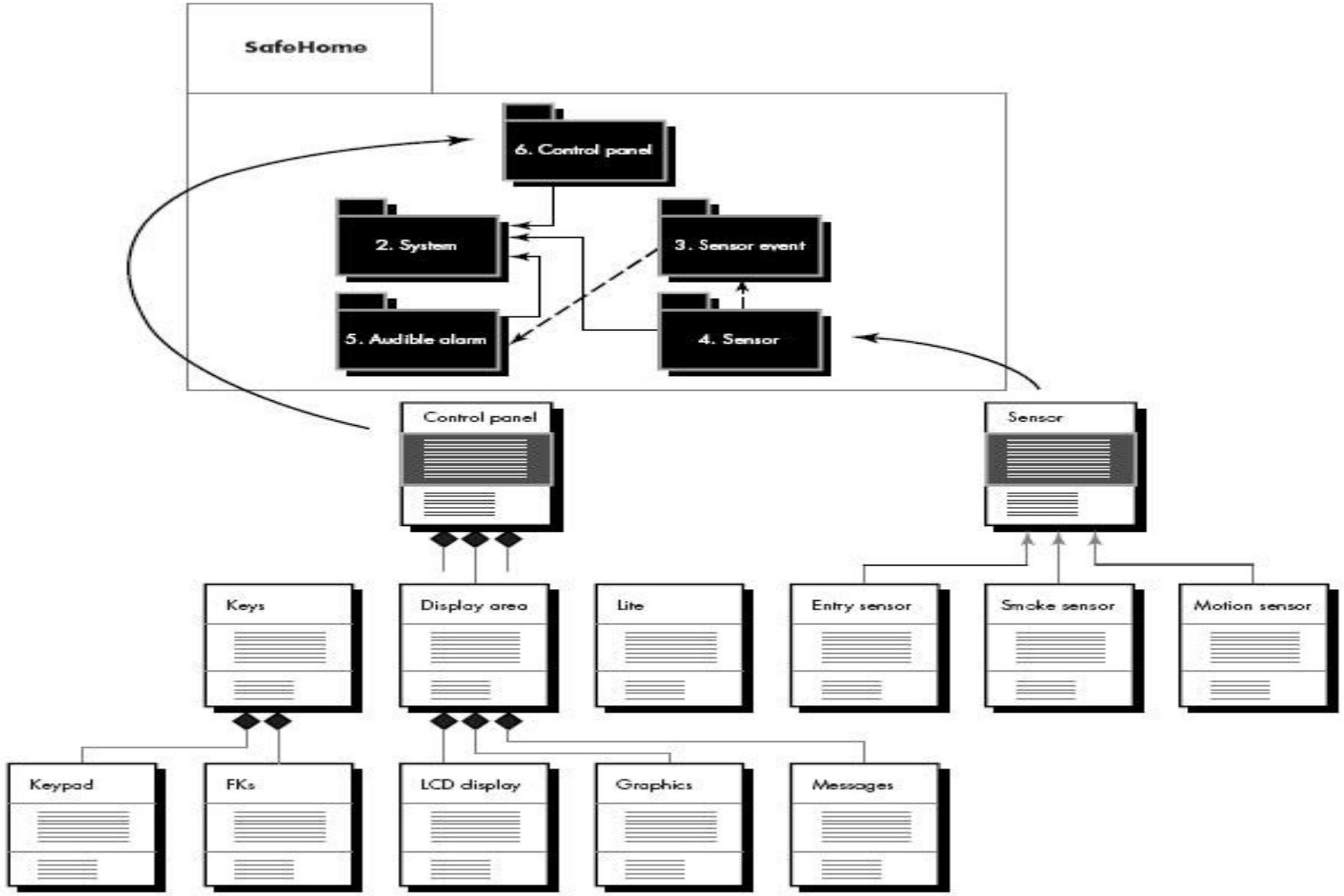
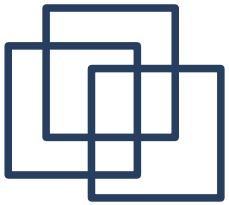
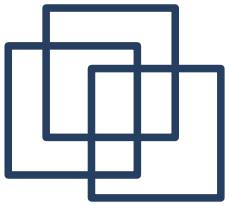


FIGURE 21.7 An analysis model with package references



Object-Relationship Model

- Relationship exists between two connected classes
  - Binary – between two classes
- Perform grammatical parse
  - Indicates physical location (next to, part of)
  - Communications (transmits to, acquires from)
  - Ownership (incorporated by, is composed of)
  - Satisfaction of a condition (manages, coordinates, control)

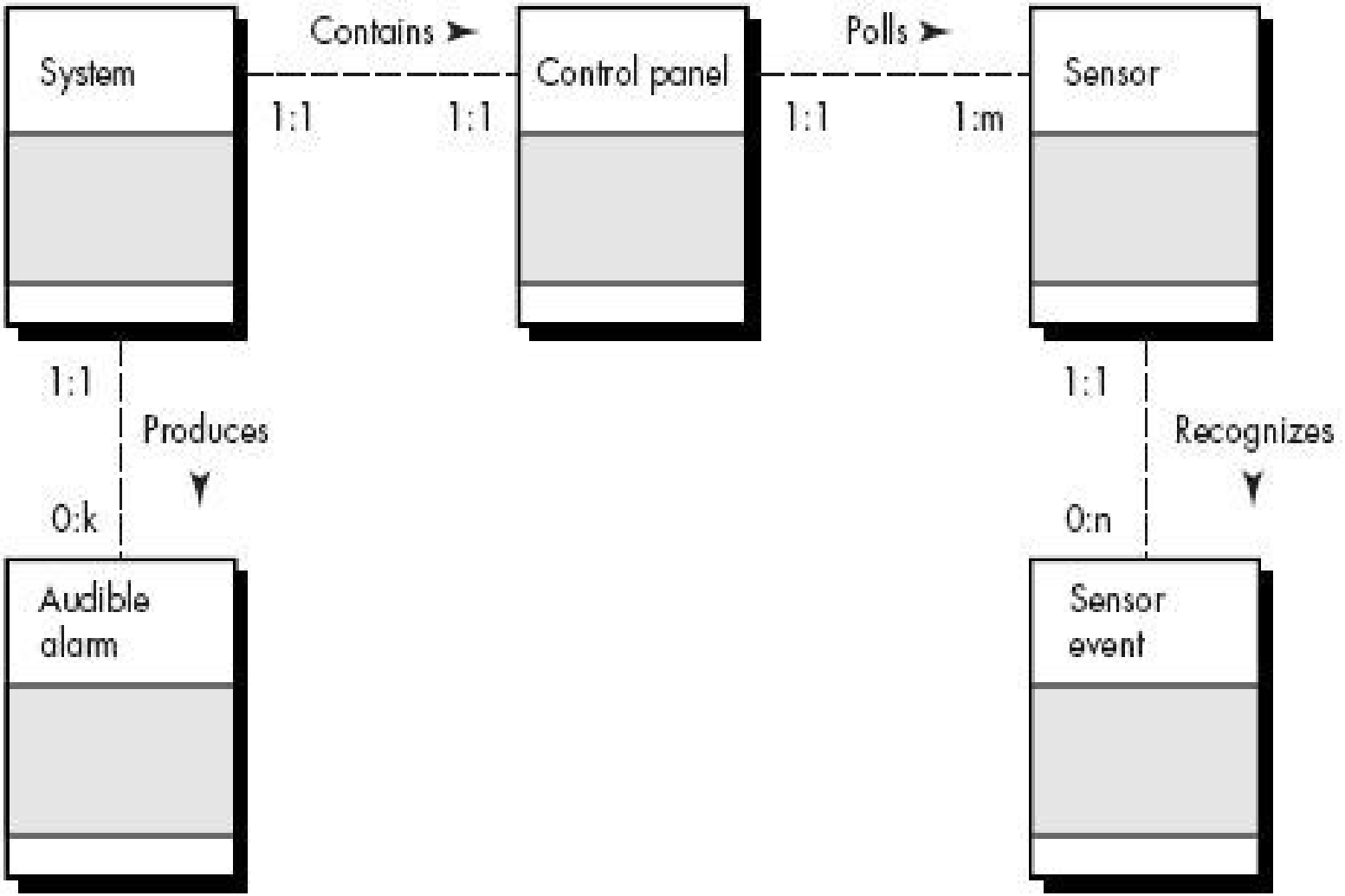


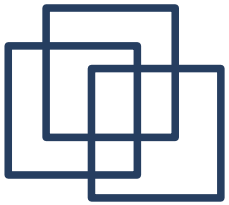
Object-Relationship Model

- Deriving object-relationship model (almost same as ER model)
  1. Using CRC cards, network of collaborators can be drawn. First objects are connected by unlabeled lines to indicate that relationship exists
  2. CRC cards are reviewed and unlabeled connected line is named
  3. Indicate cardinality of the relationship
- Indicated *message paths*

Object-Relationship Model

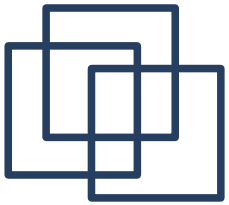
FIGURE 21.8
Relationships between objects





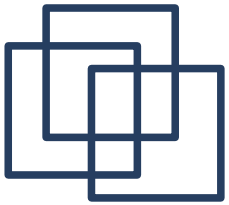
Object-Behavior Model

- Models behavior – *dynamic* aspect of object model
- Indicates how an OO system will respond to external events or stimuli
- Steps
  1. Understand sequence of interaction-review use case
  2. Identify events that drive the interaction sequence
  3. Create an event trace
  4. Build an STD
  5. Review to verify accuracy and consistency



State Representations

- Two difference characterizations of states
  - State of each object as the system performs its functions
  - State of the system as observed from the outside as the system performs its function
- May be *passive* or *active*
  - Passive – current status of all of an objects attributes
  - Active – status of object as it undergoes a continuing process

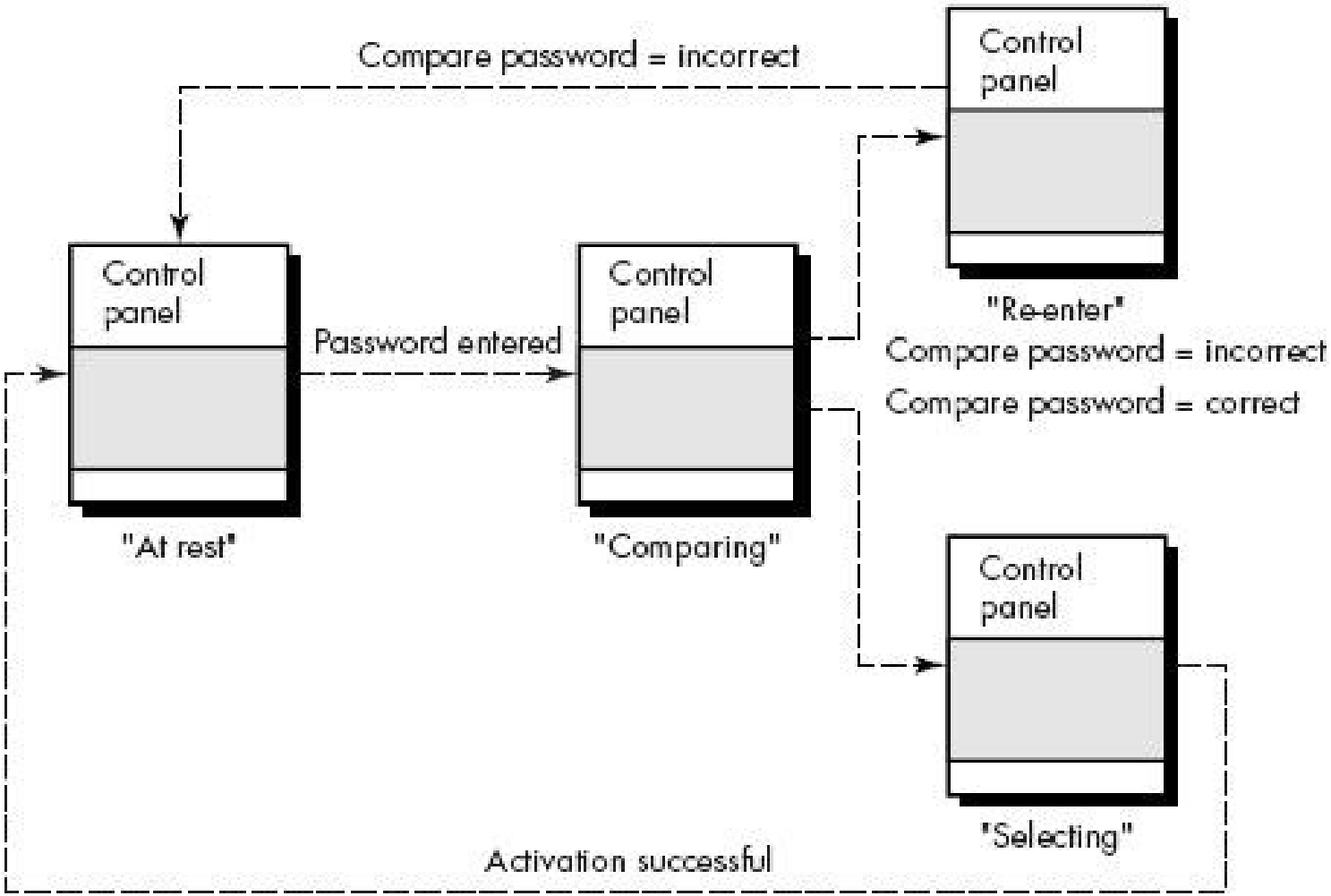


State Representations

- An event (trigger) must occur to force an object to make a transition from one active state to another
- A guard condition is a boolean condition that must be satisfied for the transition to occur
  - Depends on the passive state of object
- An action occurs concurrently with the transition
  - Involves one or more operations

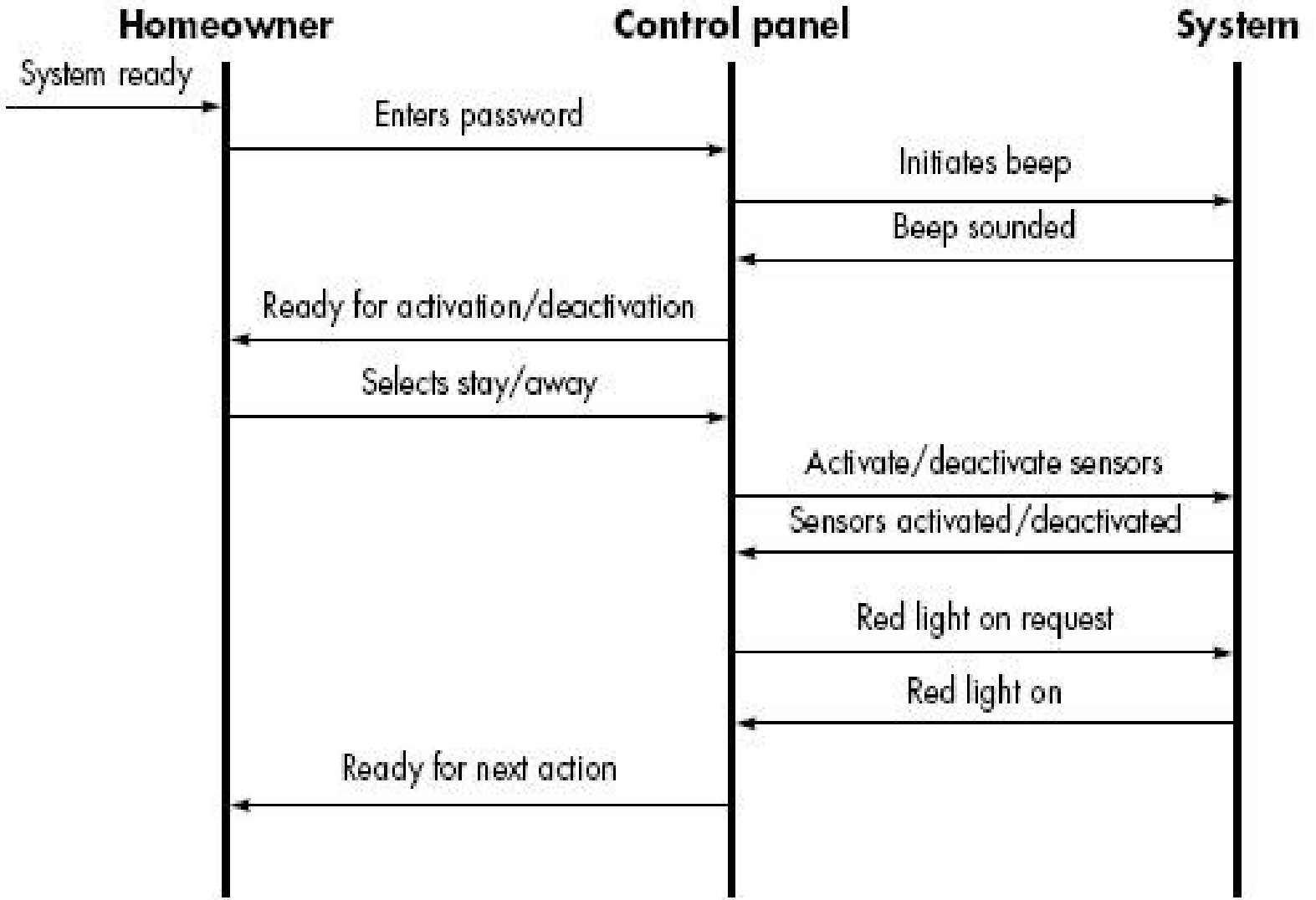
Object-Behavior Model

FIGURE 21.9
A representation of active state transitions



Object-Behavior Model

FIGURE 21.10
A partial event trace for SafeHome



Object-Behavior Model

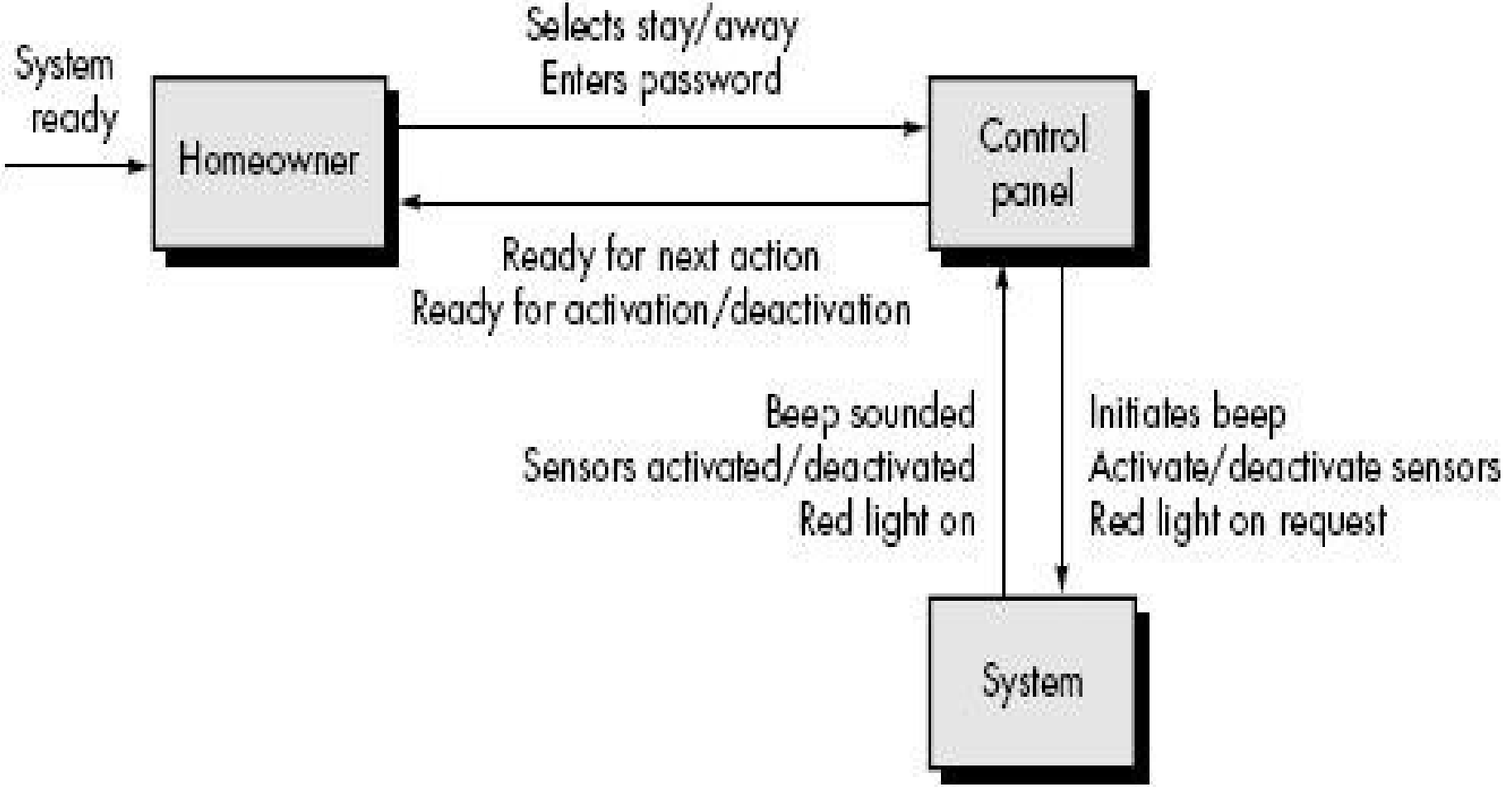
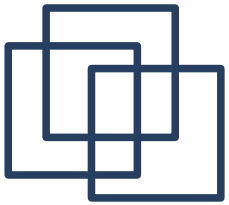
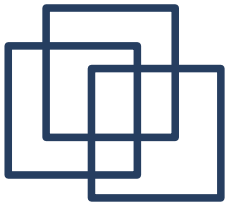


FIGURE 21.11 A partial event flow diagram for SafeHome



Summary

- OOA models a problem using objects, attributes, and operations as primary modeling components
- A wide variety of methods is available
- Analysis for OO systems occurs at many different levels of abstraction
- OOA involves the use of Use Cases, CRC, class hierarchies, object-relationship model, and object-behavioral model



Reference

- Roger S. Pressman.Software Engineering: A Practitioner's Approach, 4th Ed.McGraw-Hill,1997. Chapter 20