Chapter 8
Analysis Modeling

Moonzoo Kim
CS Division of EECS Dept.
KAIST
moonzoo@cs.kaist.ac.kr
http://pswlab.kaist.ac.kr/courses/cs550-07

Roadmap of SEPA covered in CS550

- March : Ch 1 – Ch 5
  - 1. Intro to SE
  - 2. A Generic View of Process
  - 3. Process Models
  - 4. An Agile View of Process
  - 5. SE Practice
  - 6. System Engineering
- April: Ch 7- Ch 9
  - 7. Requirement Engineering
    - Req. eng tasks
    - Req. elicitation
    - Developing use-cases
    - Building the analysis model
  - 8. Building the Analysis Model
  - 9. Design Engineering

SafeHome Project

- Use-case diagram
- Use-cases
- Activity diagram

- May : Ch 10 – Ch 14
  - 10. Creating an Architectural Design
  - 11. Modeling Component-Level Design
  - 12. Performing UI Design
  - 13. Testing Strategies
  - 14. Testing Tactics

SafeHome Project

- Class diagram
- CRC cards
- Sequence diagram
- State diagram

Overview of Ch 8. Building the Analysis Model

- April 10: ch 8.1- ch 8.5
  - 8.1 Requirement Analysis
  - 8.2 Analysis Modeling Approaches
  - 8.3 Data Modeling Concepts
  - 8.4 Object-Oriented Analysis
  - 8.5 Scenario-based modeling

- April 12: ch 8.6- ch 8.8
  - 8.6 Flow-oriented modeling
  - 8.7 Class-based modeling
  - 8.8 Creating a behavioral model

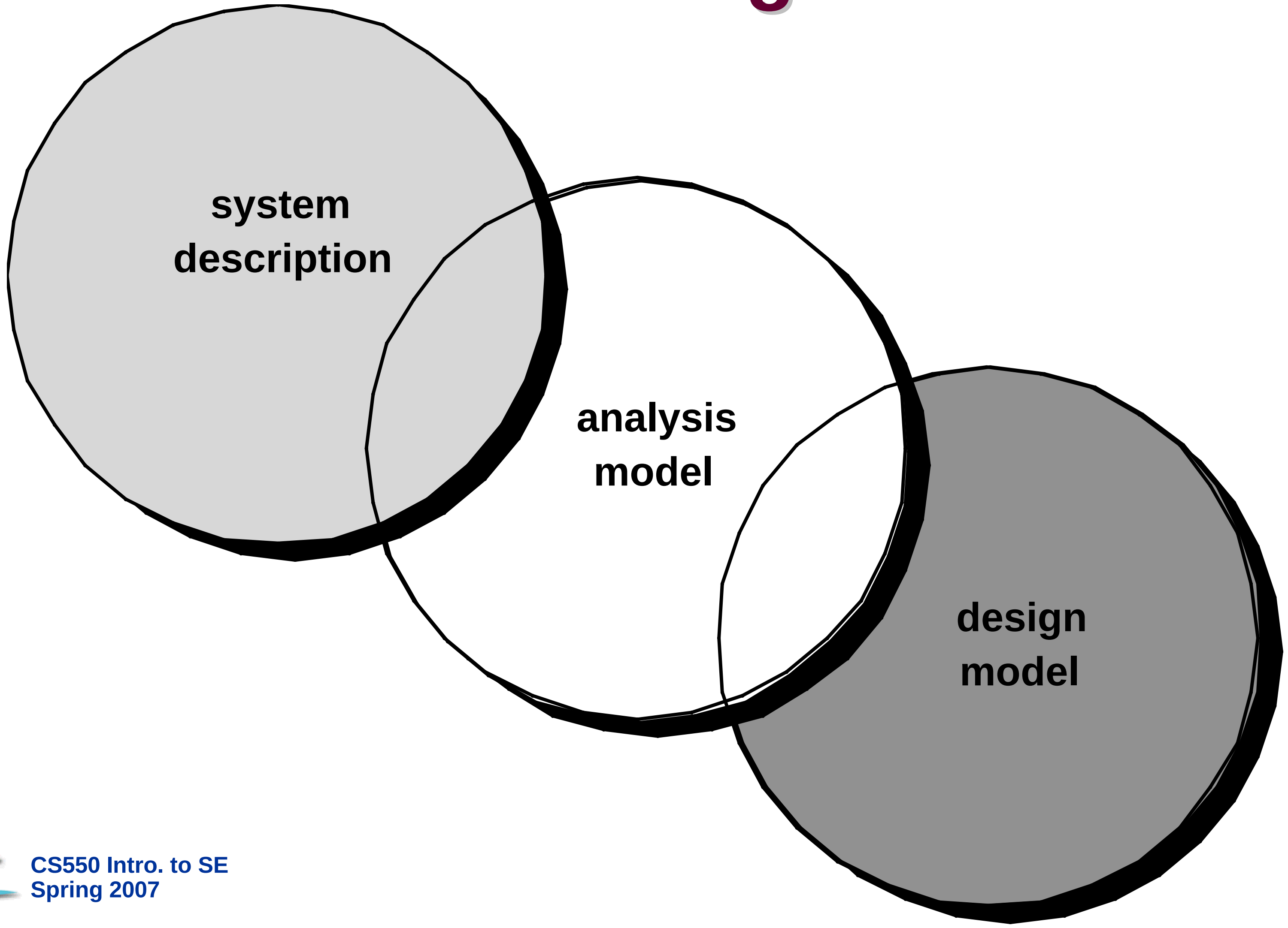
Requirements Analysis

- At a technical level, SE begins with a building an analysis model of a target system
- Requirements analysis
  - specifies software’s operational characteristics
  - indicates software's interface with other system elements
  - establishes constraints that software must meet
- Objectives
  1. To describe what the customer requires
  2. Establish a basis for the creation of a SW design
  3. To define a set of requirements that can be validated once the software is built

Requirements Analysis

- Requirements analysis allows the software engineer to:
  - elaborate on basic requirements established during earlier requirement engineering tasks
    - see Ch 7. “Requirements Engineering”
  - build models that depict
    - user scenarios
    - functional activities
    - problem classes and their relationships
    - system and class behavior
    - the flow of data as it is transformed.

A Bridge



Rules of Thumb

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

Domain Analysis

Software domain analysis is the identification, analysis, and specification of common requirements from a specific application domain, typically for reuse on multiple projects within that application domain . . . [Object-oriented domain analysis is] the identification, analysis, and specification of common, reusable capabilities within a specific application domain, in terms of common objects, classes, subassemblies, and frameworks . . .

Donald Firesmith

- 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.

Data Modeling

- Analysis modeling often begins with data modeling
  - Examines data objects independently of processing
  - Focuses attention on the data domain
  - Indicates how data objects relate to one another
- Relationship among data objects can be expressed in UML very well

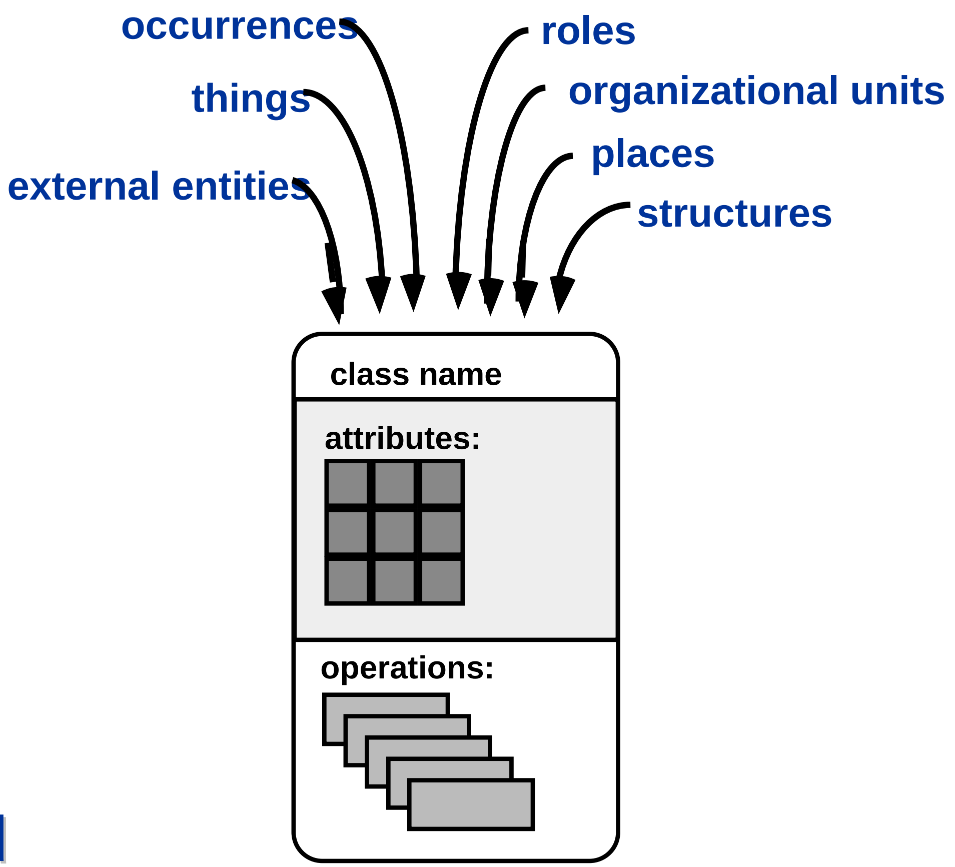
- Typical data objects
  - External entities
    - printer, user, sensor
  - Things
    - reports, displays, signals
  - Occurrences or events
    - interrupt, alarm
  - Roles
    - manager, engineer, salesperson
  - Organizational units
    - division, team
  - Places
    - manufacturing floor
  - Structures
    - employee record

Object-Oriented Concepts

- Must be understood to apply class-based elements of the analysis model
- Key concepts:
  - Classes and objects
  - Attributes and operations
  - Encapsulation and instantiation
  - Inheritance

Classes

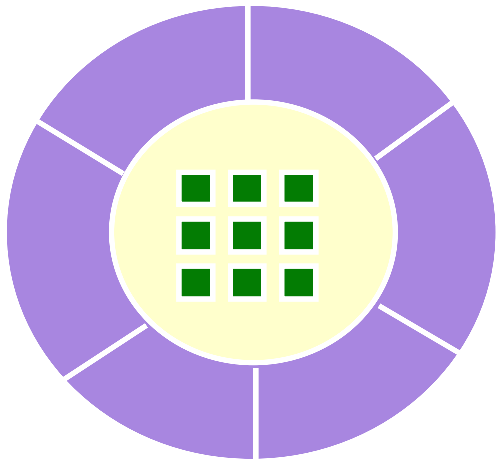
- Object-oriented thinking begins with the definition of a class, often defined as:
  - template
  - generalized description
  - "blueprint" ... describing a collection of similar items
- A superclass establishes a hierarchy of classes
- Once a class of items is defined, a specific instance of the class can be identified



Methods (a.k.a. Operations, Services)

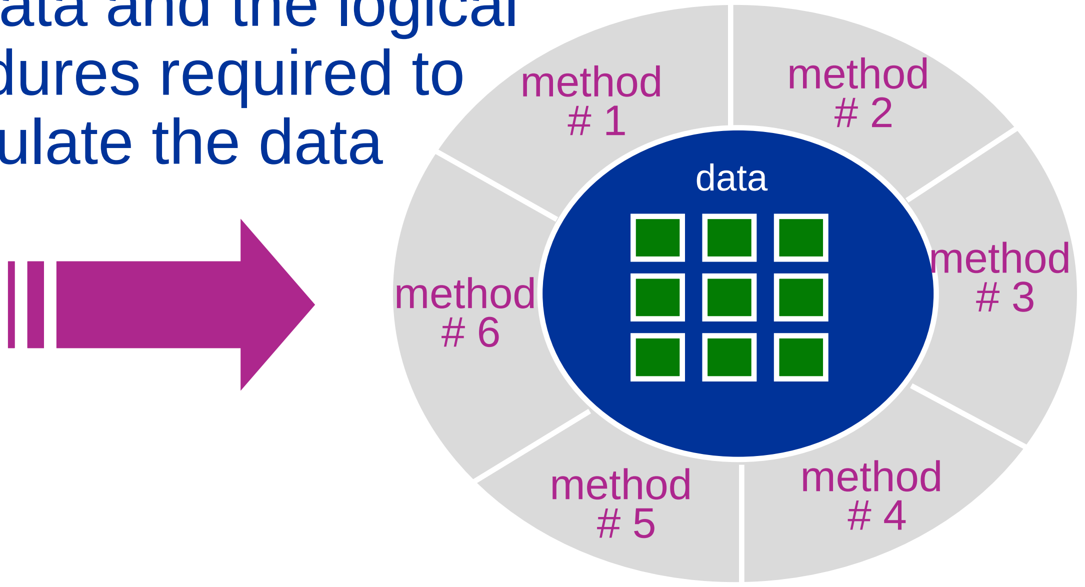
An executable procedure that is encapsulated in a class and is designed to operate on one or more data attributes that are defined as part of the class.

A method is invoked via message passing.



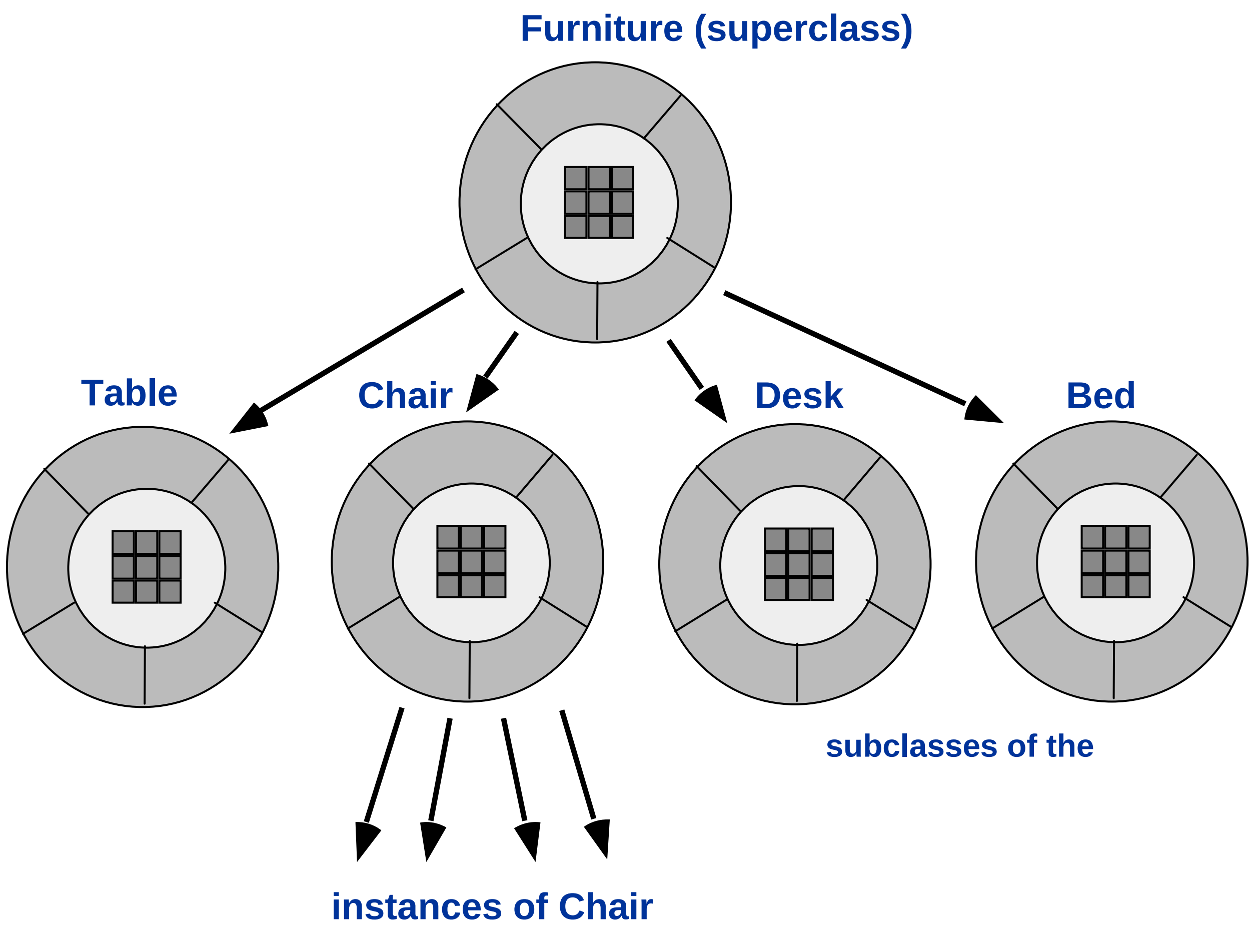
Encapsulation/Hiding

The object encapsulates both data and the logical procedures required to manipulate the data



Achieves “information hiding”

Class Hierarchy



How to Define All Classes

1. Basic user requirements must be communicated between the customer and the SW engineer
2. Classes must be identified
   - Attributes and methods are to be defined
3. A class hierarchy is defined
4. Object-to-object relationships should be represented
5. Object behavior must be modeled
6. Tasks 1 through 5 are repeated until the model is complete

Scenario-Based Modeling

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

(1) What should we write about?

(2) How much should we write about it?

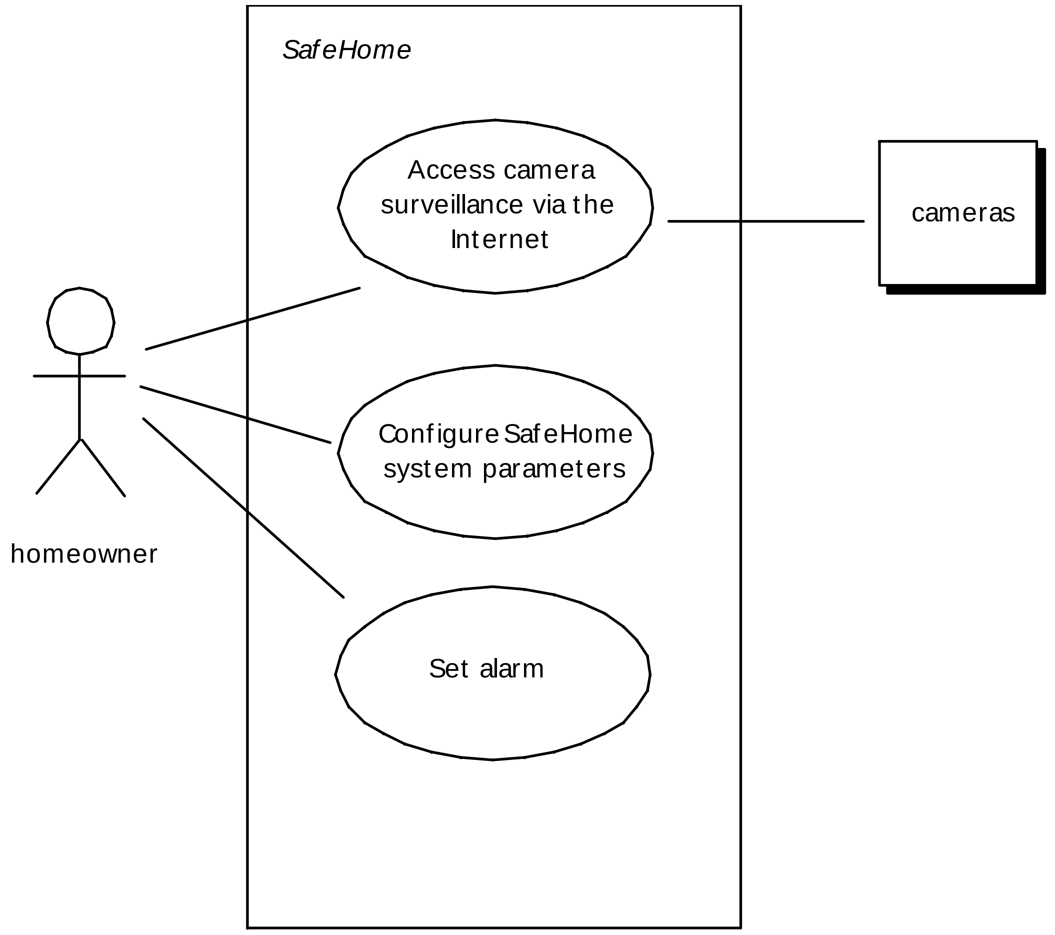
(3) How detailed should we make our description?

(4) How should we organize the description?

Use-Cases

- a scenario that describes a “thread of usage” for a system
- *actors* represent roles people or devices play as the system functions
- *users* can play a number of different roles for a given scenario
- Developing a use case
  - What are the main tasks or functions that are performed by the actor?
  - What system information will the actor acquire, produce or change?
  - What information does the actor desire from the system?

Use-Case Diagram

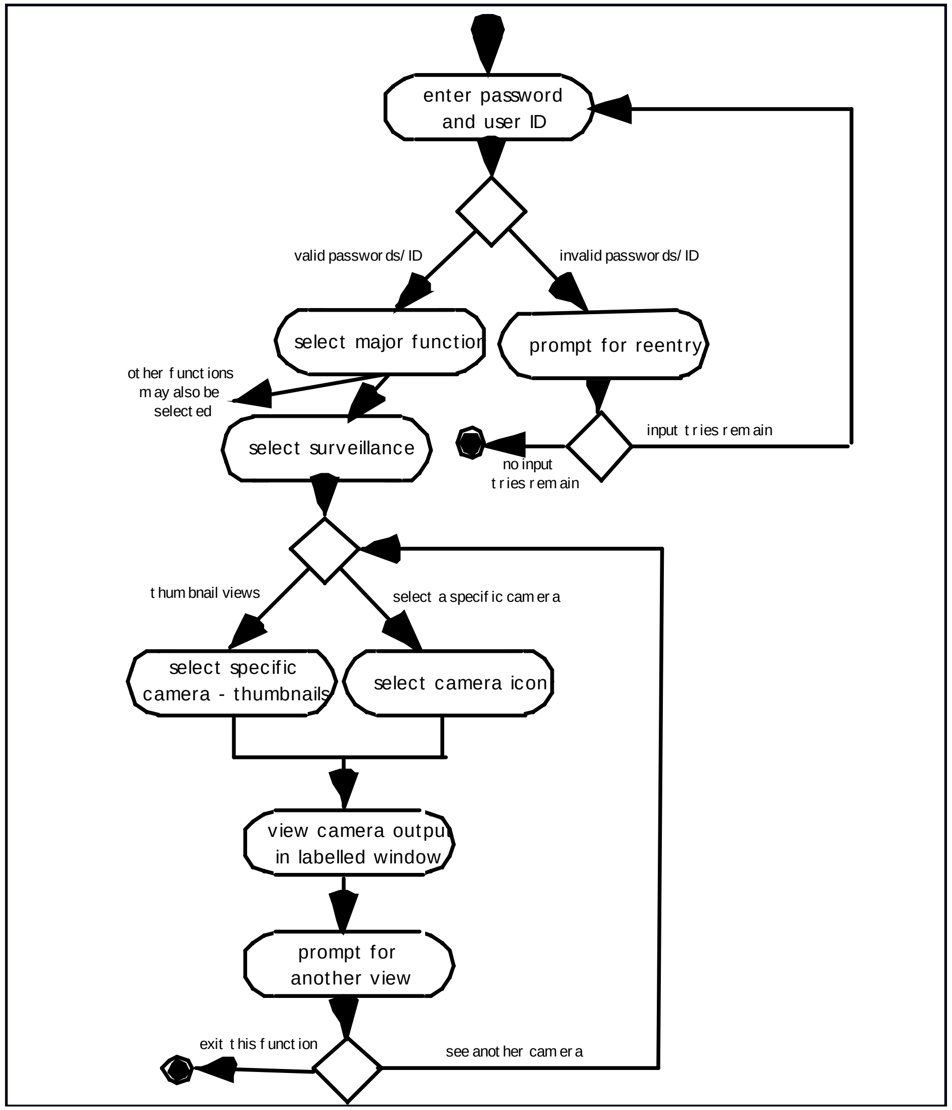


SafeHome
Access camera surveillance via the Internet
cameras
Configure SafeHome system parameters
homeowner
Set alarm

Activity Diagram

- Supplements the use-case by providing a diagrammatic representation of procedural flow

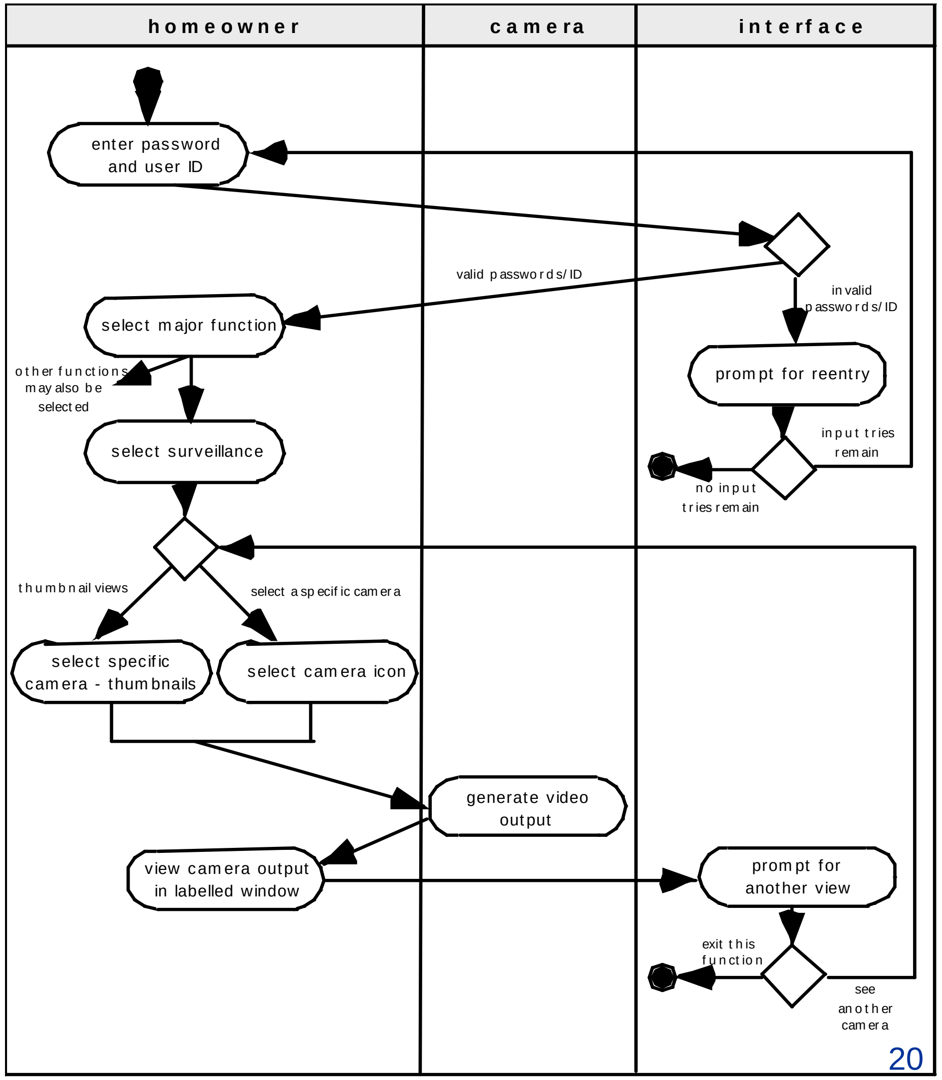
  (Fig 8.7 of 224 pg)



Swimlane Diagrams

- Allows the modeler to represent the flow of activities described by the use-case
- This diagram indicates which actor or analysis class has responsibility for the action described by an activity rectangle

  (Fig 8.8 of 225 pg)



HW #3. Due April 17

8.2 Is it possible to develop an effective analysis model without developing all four elements shown in Fig.8.3? Explain

8.3 Is it possible to begin coding immediately after an analysis model has been created? Explain your answer and then argue the counterpoint (e.g. negative effects)

8.5 An analysis rule of thumb is that the model "should focus on requirements that are visible within the problem or business domain." What types of requirements are not visible in these domains? Provide a few examples.

8.19 Write a template-based use-case for the SafeHome home management system ("SafeHome home management function") described informally in the sidebar following Sect 8.7.4. Also draw a swimlane activity diagram (use a template at Sect.7.5)