

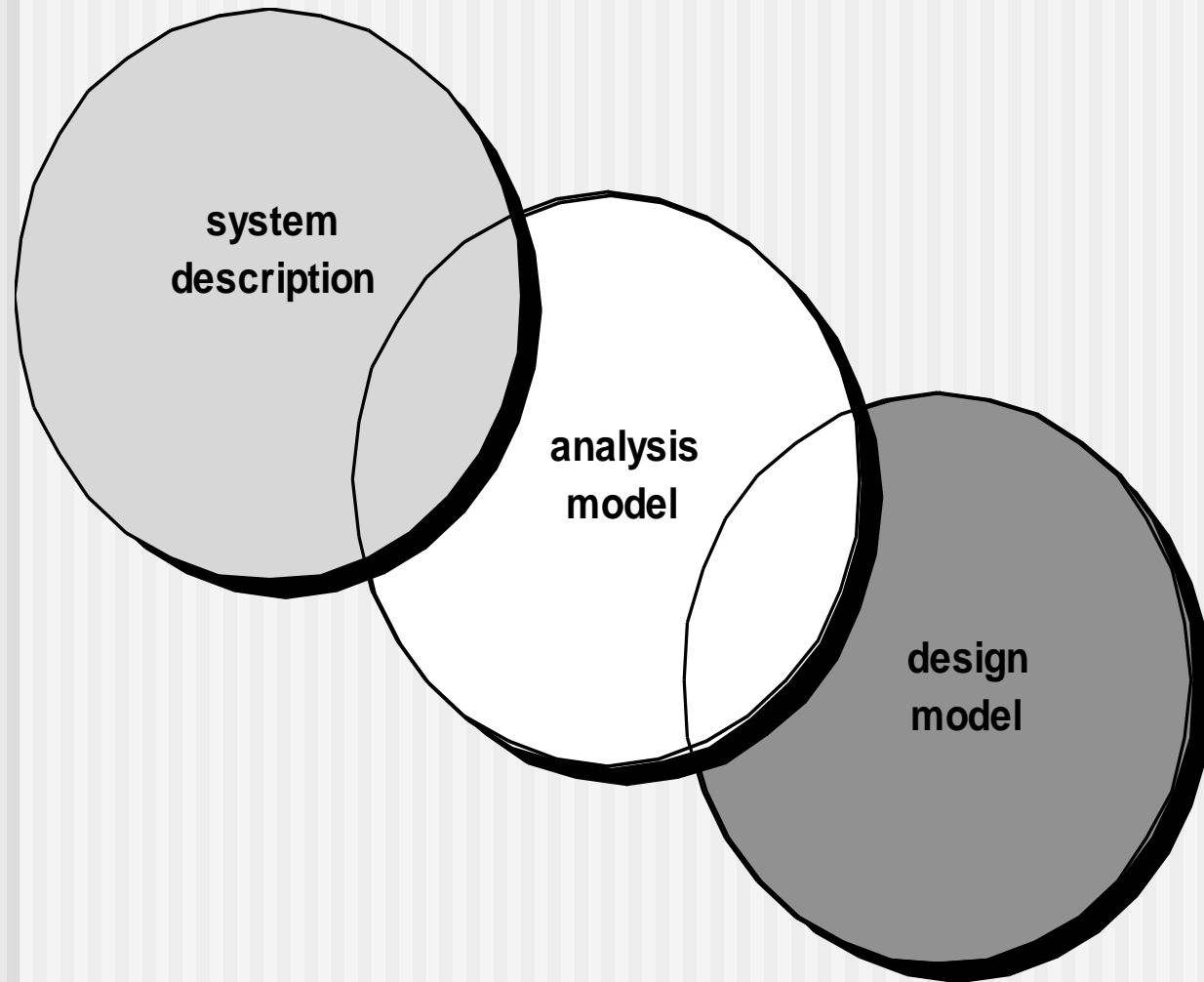
Chapter 5

- **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.

A Bridge



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

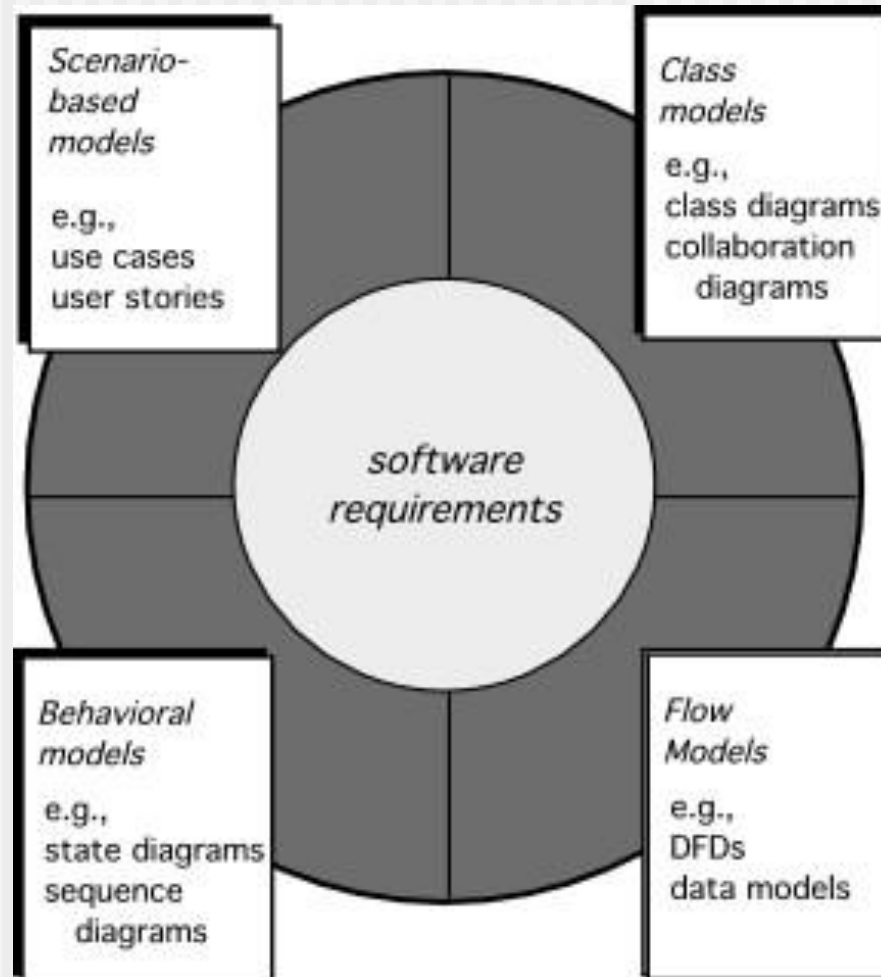
Software domain analysis is the identification, analysis, and specification of common requirements from a specific application domain, typically for reuse on multiple projects within that application domain . . . [Object-oriented domain analysis is] the identification, analysis, and specification of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

Donald Firesmith

Domain Analysis

- 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.

Elements of Requirements Analysis



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?

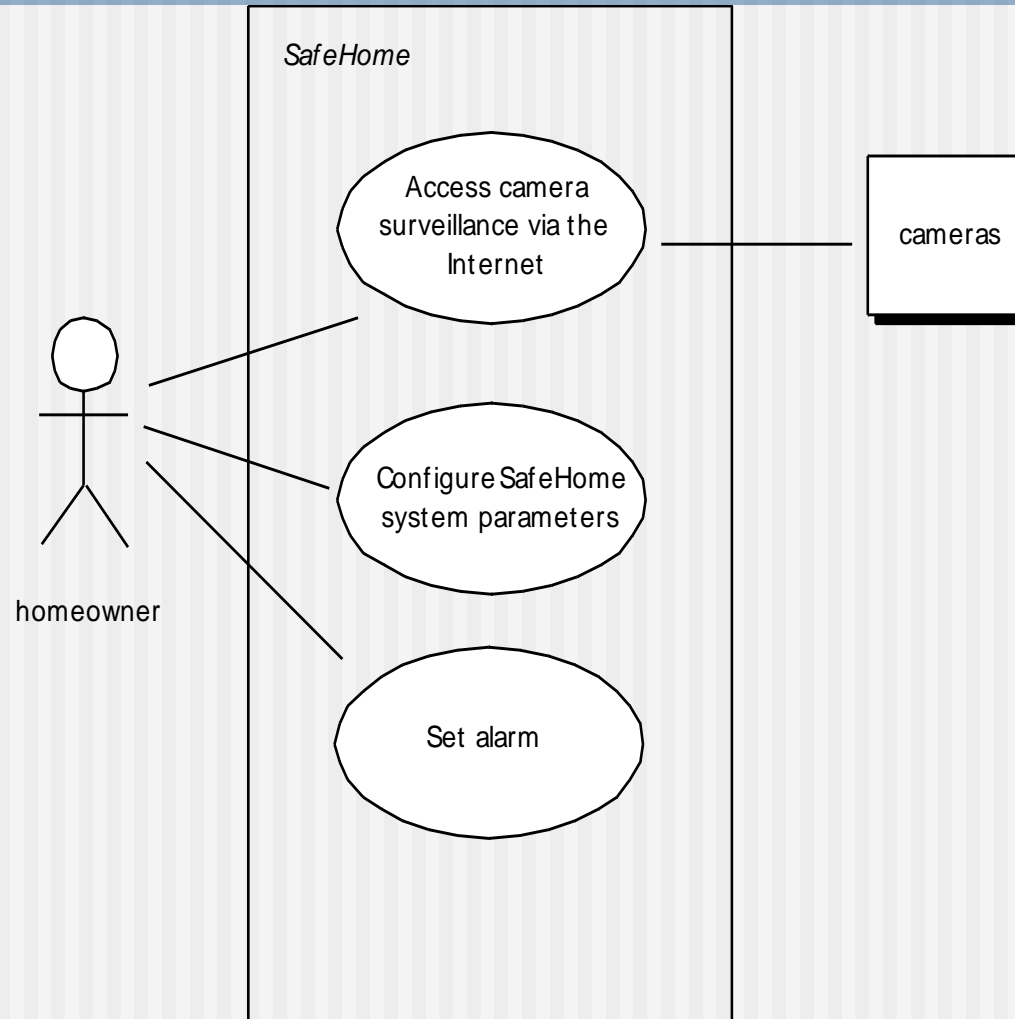
(3) How detailed should we make our description?

(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

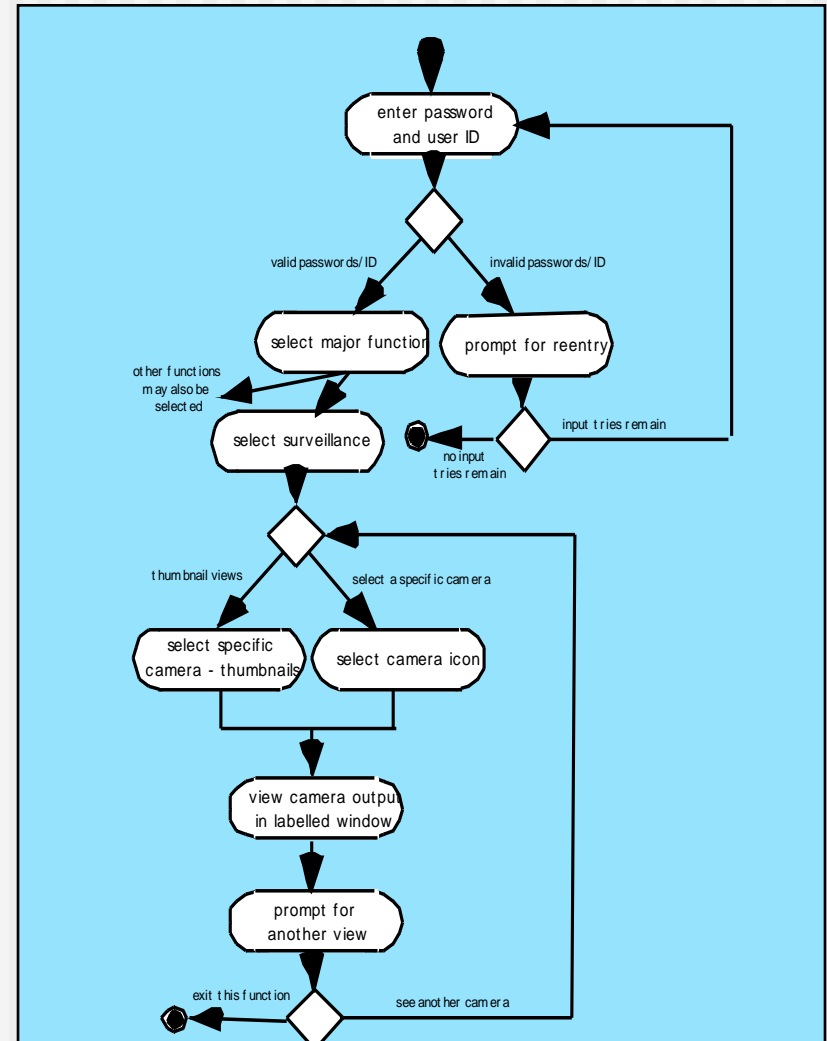
Use-Case Diagram



Activity Diagram

Supplements the use case by providing a graphical

representation of the flow of interaction within a specific scenario



Data Modeling

- examines data objects independently of processing
- focuses attention on the data domain
- creates a model at the customer's level of abstraction
- indicates how data objects relate to one another

System类的类图

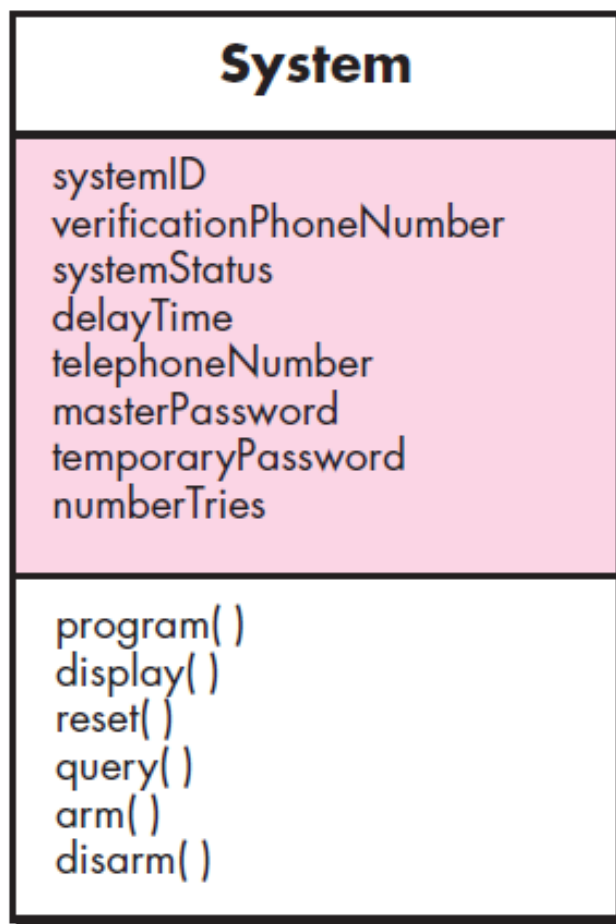


图 System类的类图

CRC Models

- *Class-responsibility-collaborator (CRC) modeling* [Wir90] provides a simple means for identifying and organizing the classes that are relevant to system or product requirements. Ambler [Amb95] describes CRC modeling in the following way:
 - 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

Questions

- 1 What are the models of requirements analysis?
- 2 What are the elements of a class-based model
- 3 What is CRC?