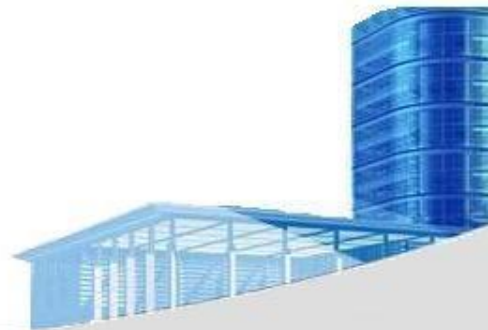




Ch.9 Requirements Modeling: Scenario-Based Methods





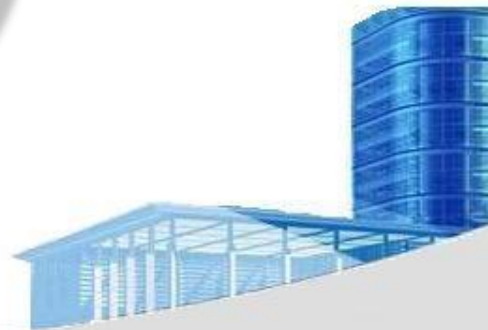
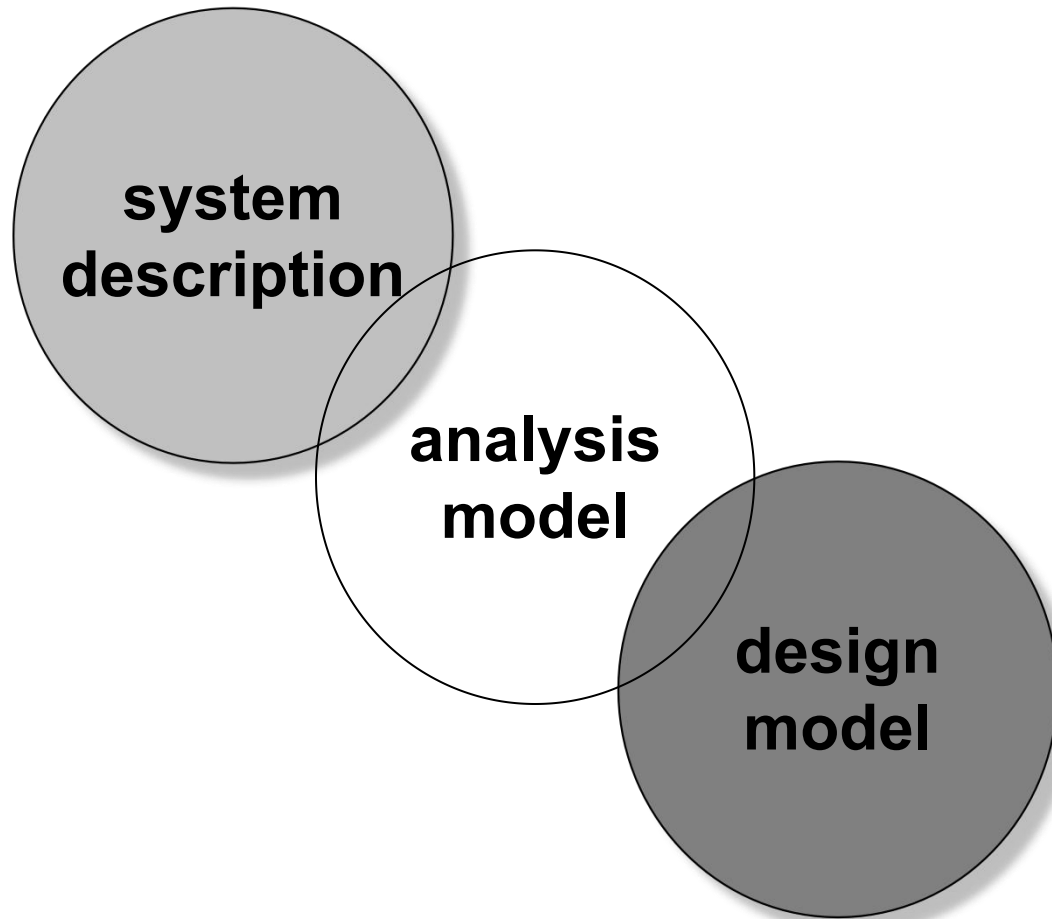
Requirements Analysis

- objectives
 - Describe **what** the customer requires
 - Establish a **basis** for the creation of a software design
 - Define a set of requirements that can be **validated**
- 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, the flow of data as it is transformed, **constraints** that software must meet.





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.

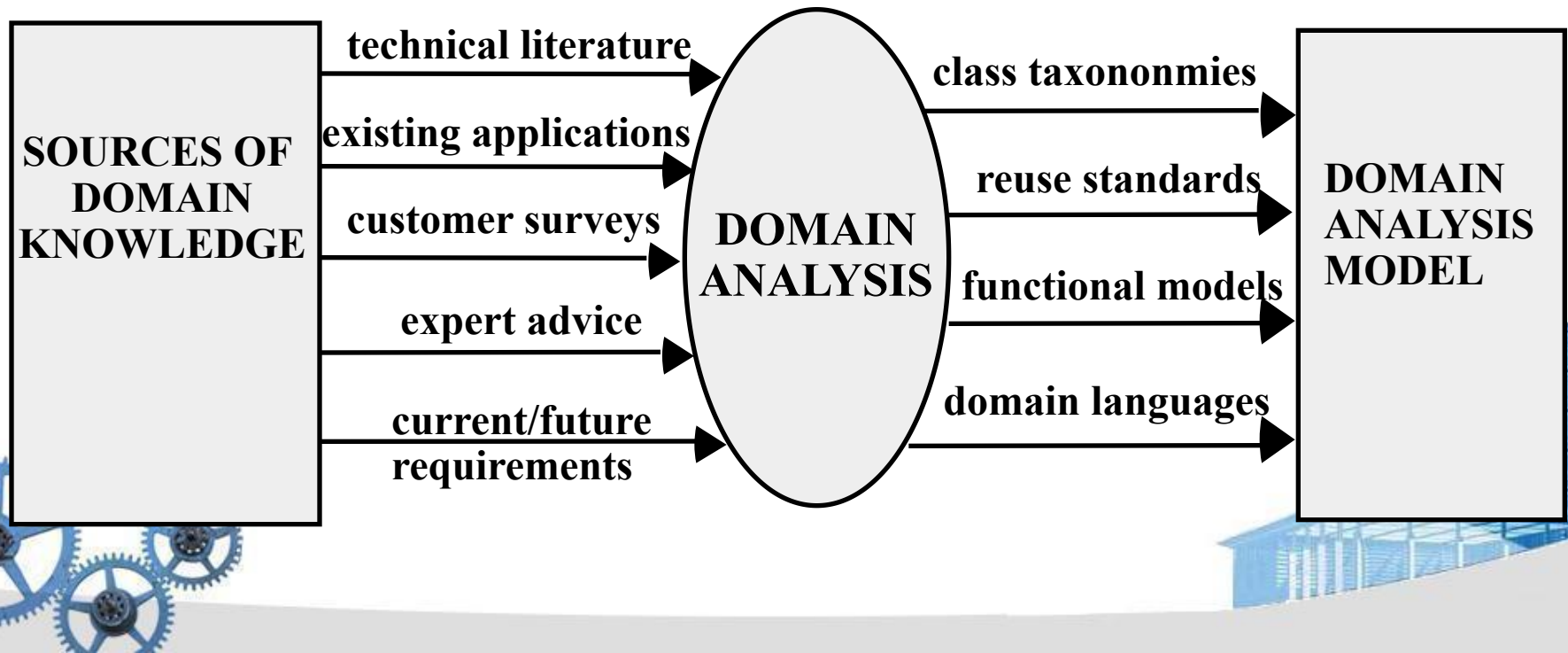


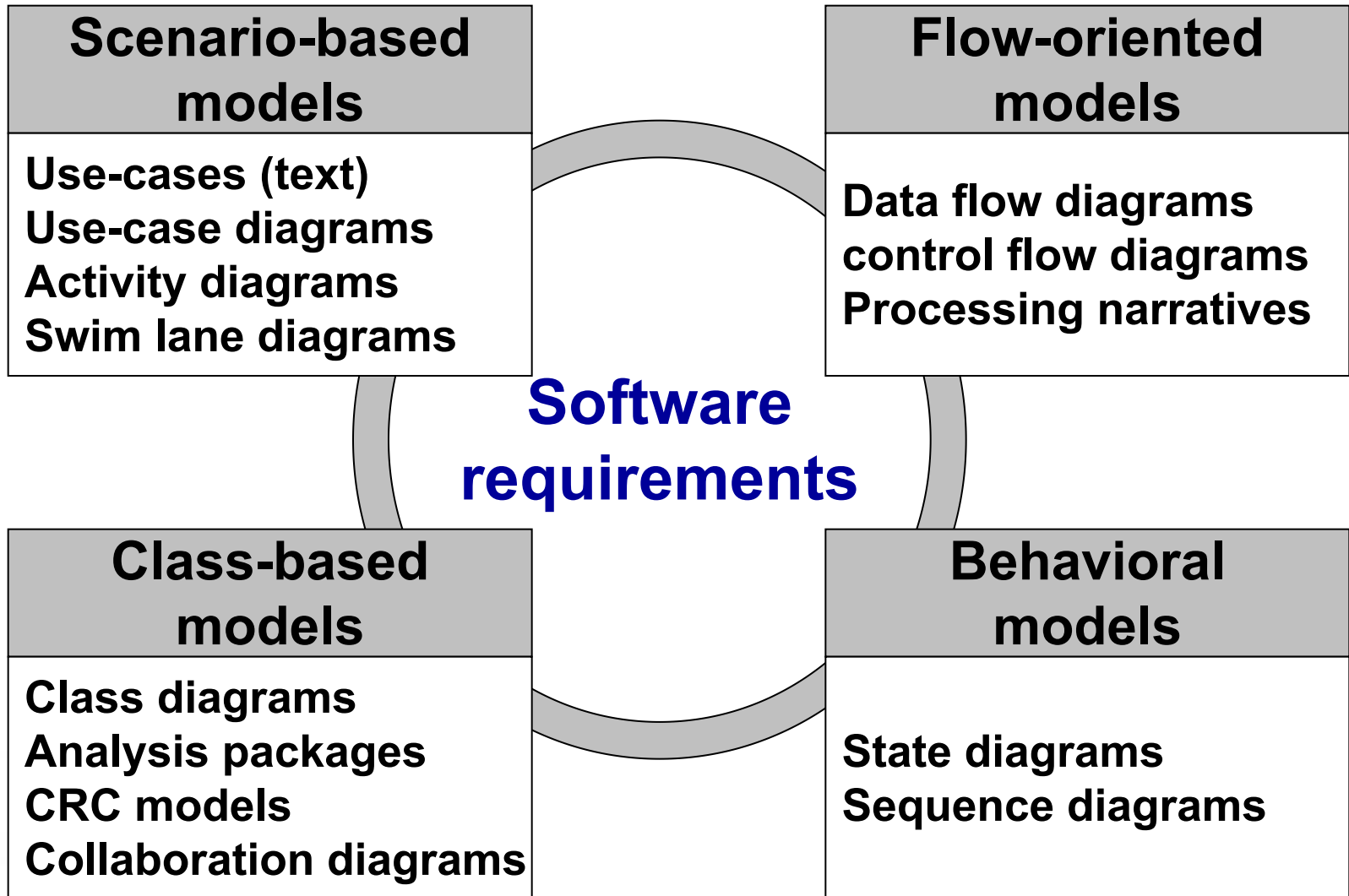


Domain Analysis



Goal: 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 . . .



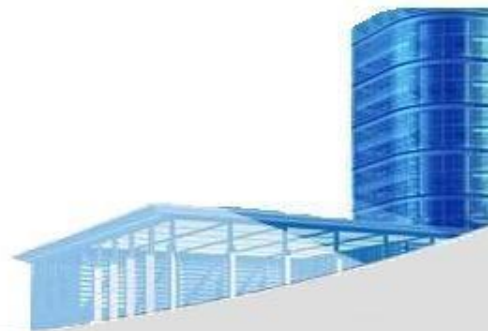




Scenario-Based Modeling

Use-cases are simply an aid to defining what exists outside the system (**actors**) and what should be performed by the system

- (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?
- 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?





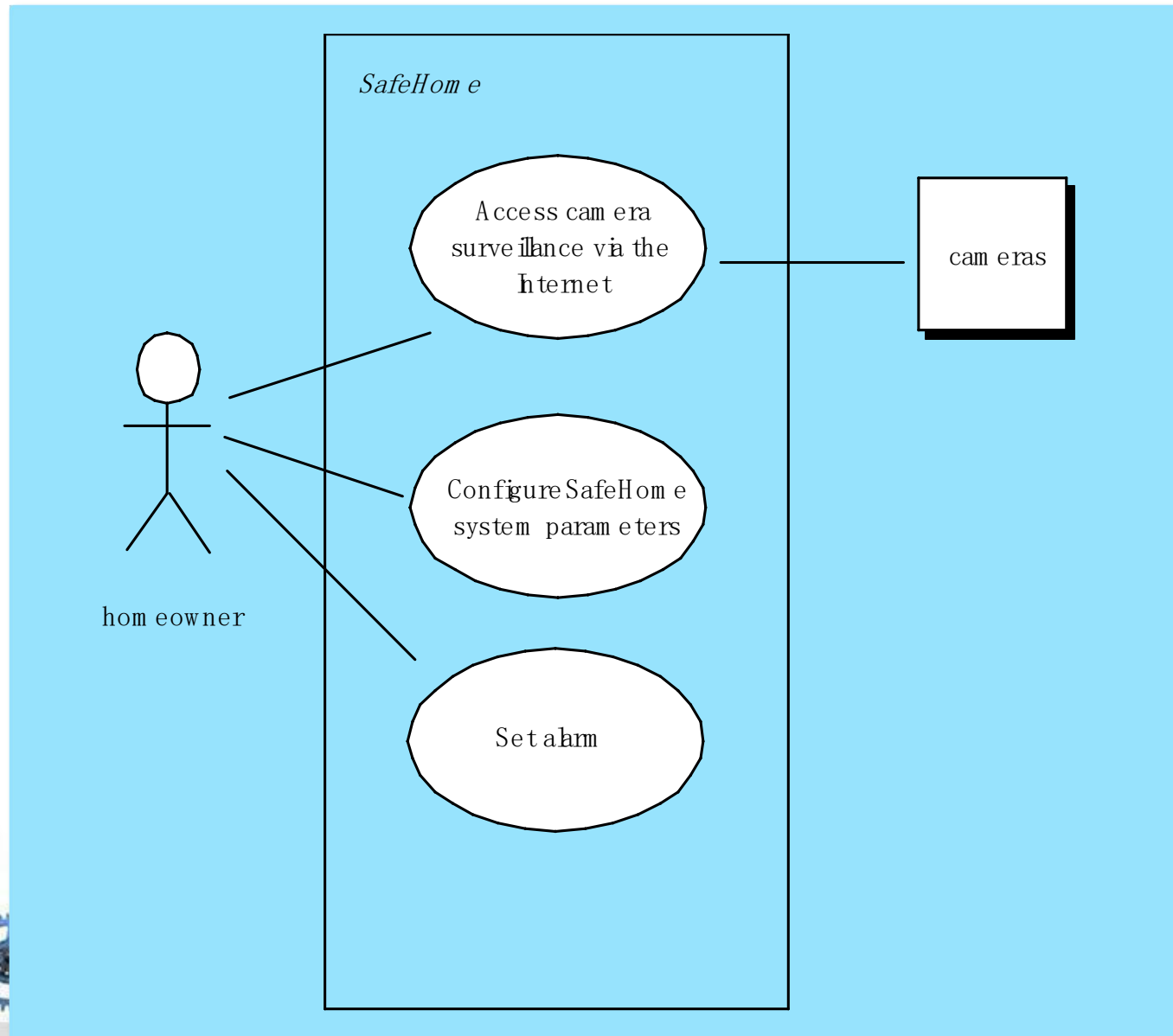
Reviewing a Use-Case

- Use-cases are written first in narrative form and mapped to a template if formality is needed
- Each primary scenario **should be reviewed and refined** to see if alternative interactions are possible
 - Can the actor take some other action at this point?
 - Is it possible that the actor will encounter an error condition at some point? If so, what?
 - Is it possible that the actor will encounter some other behavior at some point? If so, what?





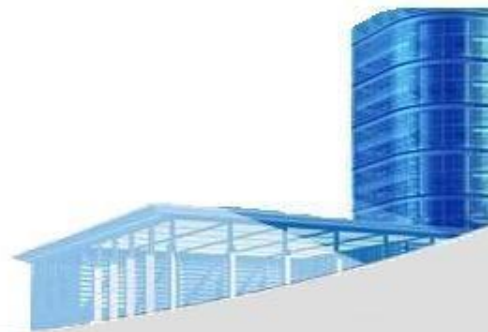
Use-Case Diagram





Activity and Swim Lane Diagrams

- **Activity diagram** supplements the use-case by providing a diagrammatic representation of procedural flow
- **Swim lane diagram** allows the modeler to represent the flow of activities described by the use-case and at the same time indicate which actor (if there are **multiple actors** involved in a specific use-case) or analysis class has **responsibility** for the action described by an activity rectangle





Activity diagram for access camera surveillance — display camera views function

