

Chapter 9

■ Requirements Modeling: Scenarios, Information, and Analysis Classes

Slide Set to accompany

Software Engineering: A Practitioner's Approach, 7/e
by Roger S. Pressman

Slides copyright © 1996, 2001, 2005, 2009 by Roger S. Pressman

For non-profit educational use only

May be reproduced ONLY for student use at the university level when used in conjunction with *Software Engineering: A Practitioner's Approach, 7/e*. Any other reproduction or use is prohibited without the express written permission of the author.

All copyright information MUST appear if these slides are posted on a website for student use.

Just enough engineering

- If you create artifacts that are not used, you have over-engineered.
 - If you do not create artifacts that are much needed, you didn't do enough engineering.
 - Just enough engineering produces only the artifacts that are much needed but no more.
- ☛ When you do something or create something you should be able to justify “why”?

Table of Contents

9.1 Requirements Analysis

9.2 Scenario-Based Modeling

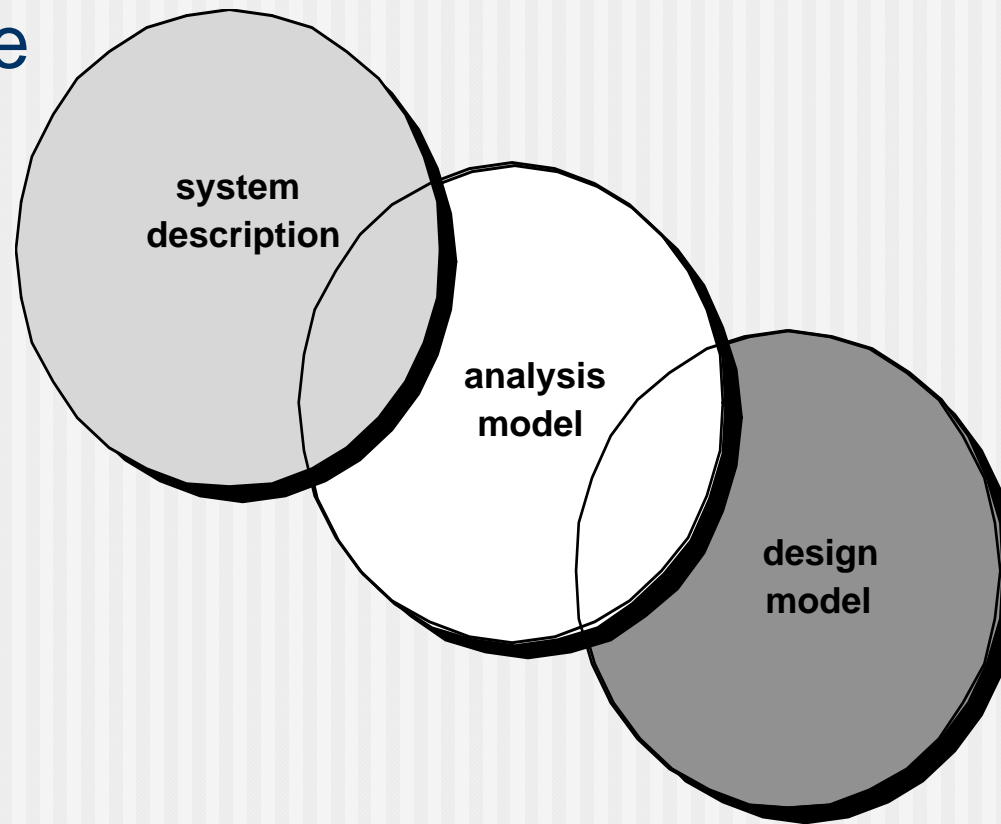
9.3 UML Models That Supplement the Use Case

9.1 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.

9.1.1 Overall Objectives and Philosophy

A Bridge



9.1.2 Analysis 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.

9.1.3 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 . . .

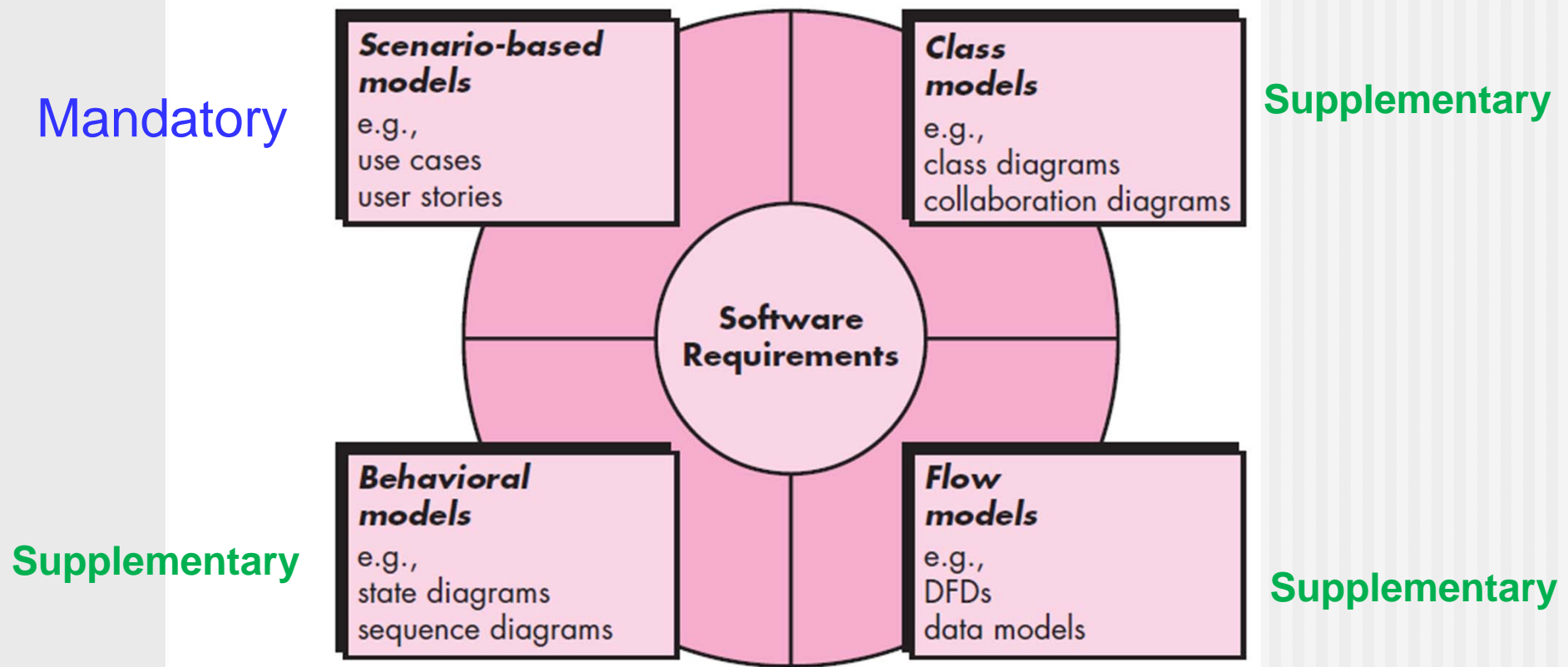
[Object-oriented domain analysis] is the identification, analysis, and specification of common, reusable capabilities within a specific application domain, in terms of common objects, classes, subassemblies, and frameworks . . .

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.

9.1.4 Elements of Requirements Analysis



9.2 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?

9.2.1 Creating a Preliminary Use Case

What to Write About?

- **Inception and elicitation**—provide you with the information you'll need to begin writing use cases.
- **Requirements gathering meetings, QFD, and other requirements engineering mechanisms** are used to
 - identify stakeholders
 - define the scope of the problem
 - specify overall operational goals
 - establish priorities
 - outline all known functional requirements, and
 - describe the things (objects) that will be manipulated by the system.
- To begin developing a set of use cases, **list the functions or activities performed by a specific actor.**

How Much to Write About?

- As further conversations with the stakeholders progress, the requirements gathering team develops use cases for each of the functions noted.
 - In general, use cases are written first in an **informal narrative fashion**.
 - If more formality is required, the same use case is rewritten using a **structured format**.

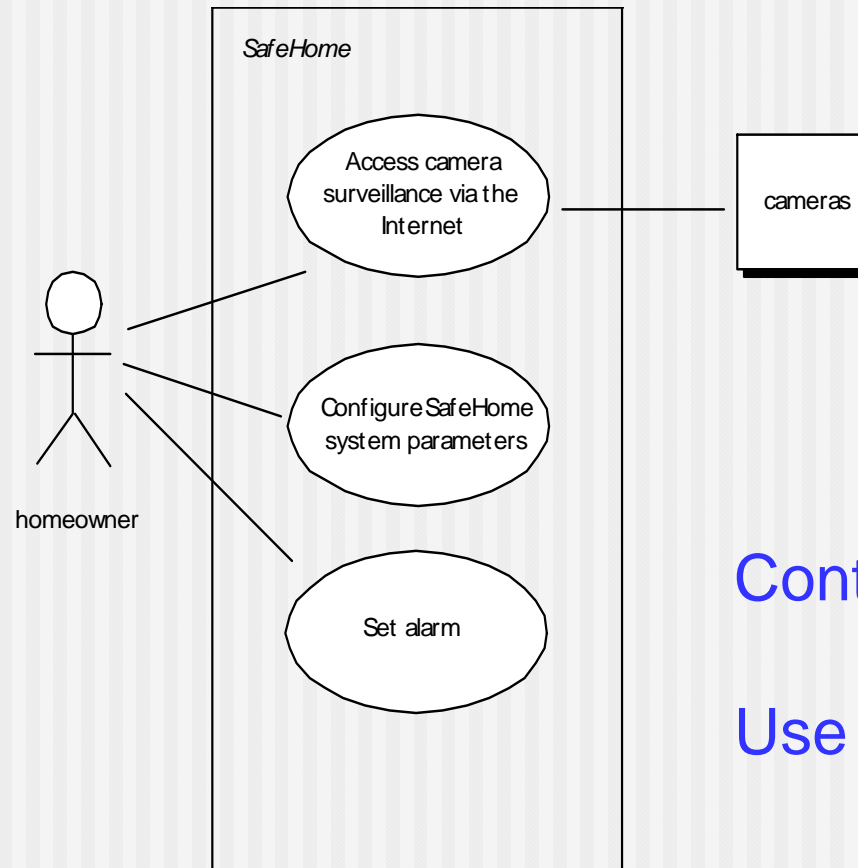
Use-Cases

- 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?

Use-Case Diagram



Context Diagram
VS.
Use Case Diagram

9.2.2 Refining a Preliminary Use Case

- A description of alternative interactions (**Exceptions**) is essential for a complete understanding of the function that is being described by a use case.
- Therefore, each step in the primary scenario is evaluated by asking the following questions:
 - *Can the actor take some other action at this point?*
 - *Is it possible that the actor will encounter some error condition at this point? If so, what might it be?*
 - *Is it possible that the actor will encounter some other behavior at this point (e.g., behavior that is invoked by some event outside the actor's control)? If so, what might it be?*

➡ See example in Section 9.2.3

9.2.3 Writing a Formal Use Case

- The informal use cases presented in Section 9.2.1 are sometimes sufficient for requirements modeling.
- When a use case involves a critical activity or describes a complex set of steps with a significant number of exceptions, a more formal approach may be desirable.



Use Case Template for Surveillance

Use case: Access camera surveillance via the Internet—display camera views (ACS-DCV)

Iteration: 2, last modification: January 14 by V. Raman.

Primary actor: Homeowner.

Goal in context: To view output of camera placed throughout the house from any remote location via the Internet.

Preconditions: System must be fully configured; appropriate user ID and passwords must be obtained.

Trigger: The homeowner decides to take a look inside the house while away.

Scenario:

1. The homeowner logs onto the *SafeHome Products* website.
2. The homeowner enters his or her user ID.
3. The homeowner enters two passwords (each at least eight characters in length).
4. The system displays all major function buttons.
5. The homeowner selects the "surveillance" from the major function buttons.
6. The homeowner selects "pick a camera."
7. The system displays the floor plan of the house.
8. The homeowner selects a camera icon from the floor plan.
9. The homeowner selects the "view" button.
10. The system displays a viewing window that is identified by the camera ID.
11. The system displays video output within the viewing window at one frame per second.

Exceptions:

1. ID or passwords are incorrect or not recognized—see use case **Validate ID and passwords**.
2. Surveillance function not configured for this system—system displays appropriate error message; see use case **Configure surveillance function**.
3. Homeowner selects "View thumbnail snapshots for all camera"—see use case **View thumbnail snapshots for all cameras**.
4. A floor plan is not available or has not been configured—display appropriate error message and see use case **Configure floor plan**.
5. An alarm condition is encountered—see use case **Alarm condition encountered**.

Priority: Moderate priority, to be implemented after basic functions.

When available: Third increment.

Frequency of use: Moderate frequency.

Channel to actor: Via PC-based browser and Internet connection.

Secondary actors: System administrator, cameras.

Channels to secondary actors:

1. System administrator: PC-based system.
2. Cameras: wireless connectivity.

Open issues:

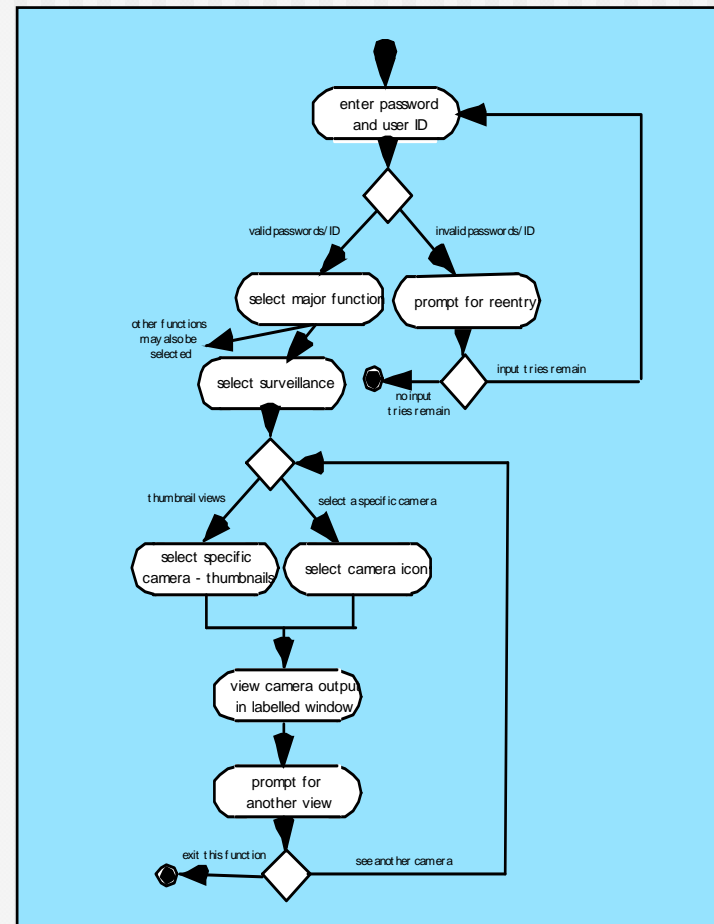
1. What mechanisms protect unauthorized use of this capability by employees of *SafeHome Products*?
2. Is security sufficient? Hacking into this feature would represent a major invasion of privacy.
3. Will system response via the Internet be acceptable given the bandwidth required for camera views?
4. Will we develop a capability to provide video at a higher frames-per-second rate when high-bandwidth connections are available?

9.3 UML Models that Supplements the Use Case

- There are many requirements modeling situations in which a text-based model does not impart information in a clear and concise manner.
- ☛ Can use UML graphical models.

9.3.1 Developing Activity Diagram

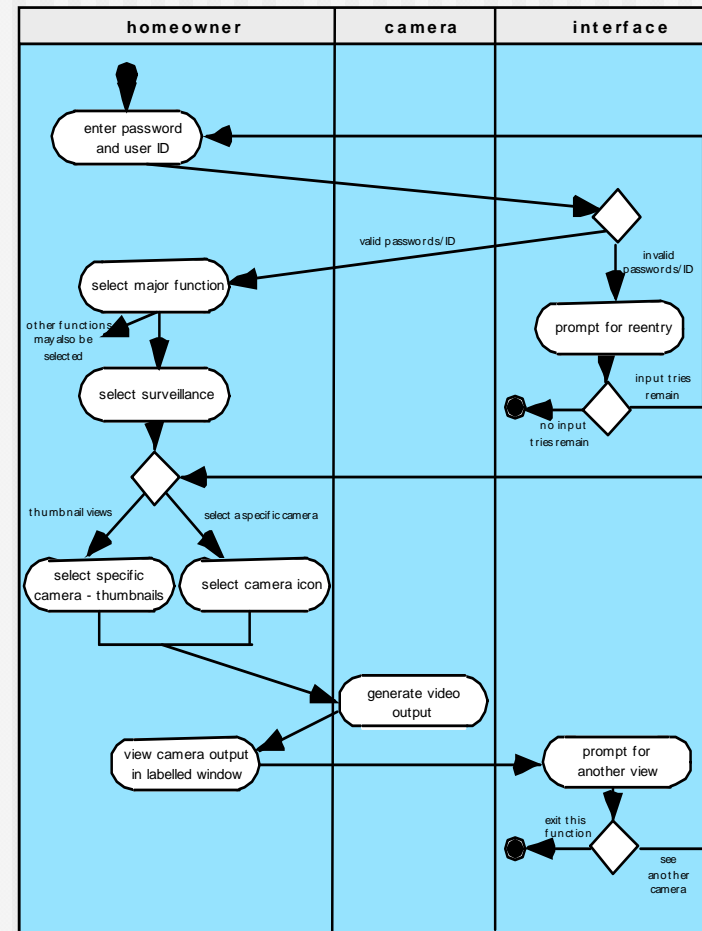
Supplements the use case by providing a graphical representation of the **flow of interaction** within a specific scenario



9.3.2 Swimlane Diagrams

Can represent

- flow of activities described by the use-case
- actors or analysis classes that have responsibility for the action described by an activity rectangle



These slides are designed to accompany *Software Engineering: A Practitioner's Approach*, 7/e (McGraw-Hill, 2009). Slides copyright 2009 by Roger Pressman.