CHAPTER 9 REQUIREMENTS MODELING: SCENARIO-BASED METHODS

9.1 Requirements Analysis

- 9.1.1 Overall Objectives and Philosophy
- 9.1.2 Analysis Rules of Thumb
- 9.1.3 Domain Analysis
- 9.1.4 Requirements Modeling Approaches

- SE begins with a **series of modeling** tasks that lead to a **specification of requirements and a design representation** for the software to be built.

- The **requirements model** actually a set of models. It is the **first technical representation of a system**.

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

- RA **allows the software engineer** (*analyst* or *modeler*)
  - Elaborate on basic requirements established during the inception, elicitation, and negotiation tasks that are part of requirements engineering.
  - 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 Requirements Analysis (Cont'd)

- Requirements modeling results in 1 or more of the following **types of models:**

A. ***Scenario based models*** *of requirements from the point of view of various* system "actors."

B. ***Class-oriented models*** *represent object-oriented classes (attributes* and operations) and the manner in which classes collaborate to achieve system requirements.

C. ***Behavioral and patterns-based models*** *depict how the software behaves* as a consequence of external "events."

D. ***Data models*** *depict the information domain for the problem.*

E. ***Flow-oriented models*** *represent the functional elements of the system* and how they transform data as they move through the system.

9.1 Requirements Analysis (Cont'd)

- Models provide information that can be **translated to architectural, interface and component-level designs.**

- Finally, model (and the software requirements specification) provide the developer and the customer with the means to **assess quality once software is built.**

- **Site plan** (land measurement)
- **Floor plan** (measurement room and floor)
- **Working plan** (horizontal dimension, wall, columns)
- **Section drawing** (material used with height of each element)
- **Elevation drawing** (size & shape of the external surface: heights, opening aesthetic (no of doors windows and their sizes)
- **Structural drawing** (steel reinforcement, foundation, plinth beam, flooring, framing material, concrete, steel beams)
- **Electrical drawing** (electrical connection, wires, lighting, power circuit tries
- **Plumbing drawings** (water connection and sanitary system, pipes, pumps, drainage system)

9.1 Requirements Analysis (Cont'd)

- In this chapter, focus is on ***scenario-based modeling*** **—*a technique that is* growing increasingly popular.**

- Over the past decade, **flow and data modeling have become less commonly used**, while scenario and class-based methods, supplemented with behavioral approaches and pattern-based techniques have grown in popularity.

9.1 Requirements Analysis (Cont'd)

9.1.1 Overall Objectives and Philosophy

- **Throughout analysis modeling, primary focus is on *what, not how*.**
- What user interaction occurs
- what objects does the system manipulate
- what functions
- what behaviors
- what interfaces
- what constraints apply?

9.1 Requirements Analysis (Cont'd)

9.1.1 Overall Objectives and Philosophy

- Complete specification of requirements may not be possible
  - **Customer** may be unsure of what is required for certain aspects
  - **Developer** may be unsure that a specific approach will properly accomplish function and performance.

- These realities mitigate if iterative approach adopted for requirements analysis and modeling.
- The analyst should model what is known and use that model as the basis for design of the software increment

9.1 Requirements Analysis (Cont'd)

9.1.1 Overall Objectives and Philosophy

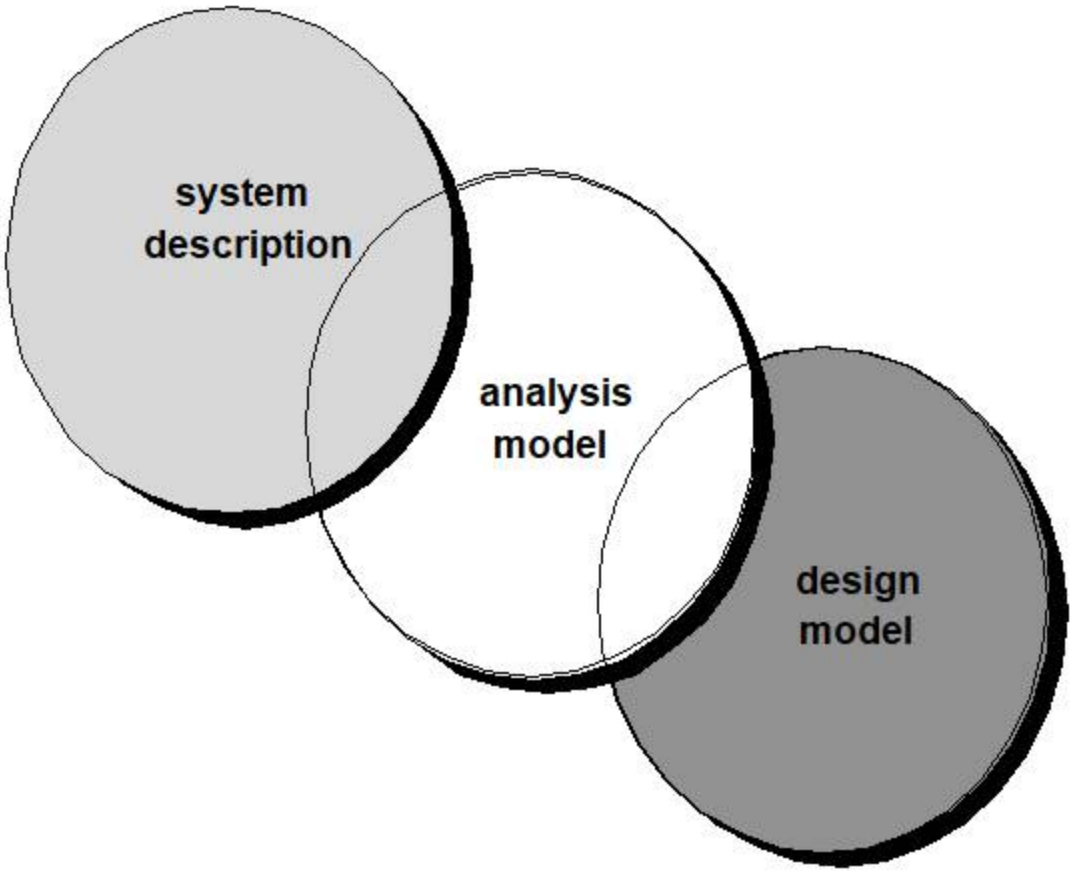
- **Requirements model 3 primary objectives:**
- (1) to describe what the customer requires,
- (2) to establish a basis for the creation of a software design, and
- (3) to define a set of requirements that can be validated once the software is built.

The analysis model bridges the gap between a system-level description that describes overall system or business functionality as it is achieved by applying software, hardware, data, human, and other system elements and a software design that describes the software's application architecture, user interface, and component-level structure.

This relationship is illustrated in Figure 9.1

9.1.1 Overall Objectives and Philosophy

FIGURE 9.1
The requirements model as a bridge between the system description and the design model



9.1 Requirements Analysis (Cont'd)

9.1.1 Overall Objectives and Philosophy

- It is important to note that all elements of the requirements model will be **directly traceable to parts of the design model**. A clear division of analysis and design tasks between these two important modeling activities is not always possible.

- **Some design invariably occurs as part of analysis, and some analysis will be conducted during design.**

9.1 Requirements Analysis (Cont'd)

9.1.2 Analysis Rules of Thumb: Arlow and Neustadt [Arl02]

1) *Model should focus on requirements that are visible within the problem or business domain. The level of* ***abstraction should be relatively high.*** **"Don't get bogged down in details". How it works.**

2) *Each element of the requirements model should add to an overall understanding of software requirements and provide insight into the* ***information domain, function, and behavior.***

3) ***Delay consideration of infrastructure*** *and other nonfunctional models until design. That is, a database may be required, but the classes to* implement, functions to access, and the behavior should be considered after problem domain analysis.

9.1 Requirements Analysis (Cont'd)

9.1.2 Analysis Rules of Thumb: Arlow and Neustadt [Arl02]

4) ***Minimize coupling*** *throughout the system. Important to represent relationships* between classes and functions. However, if the level of "interconnectedness" is extremely high, should be reduced.

5) *Be certain that the requirements model* ***provides value to all stakeholders***. Each constituency has its own use for the model. Ex: business stakeholders use model to validate requirements; designers use model as a basis for design; QA people use model to help plan acceptance tests.

6) *Keep the* ***model as simple*** *as it can be. Don't add additional diagrams* when they add no new information. Don't use complex notational forms when a simple list will do.

9.1 Requirements Analysis (Cont'd)

9.1.3 Domain Analysis

- **Analysis patterns** often reoccur across many applications within a specific business domain.

- If these **patterns are defined and categorized** in a manner that allows you to recognize and apply them to solve common problems, the creation of the analysis model is expedited.
- **Likelihood** of applying design patterns and executable software components **grows** dramatically.

- This improves time-to-market and reduces development costs.
- **But how are analysis patterns and classes recognized in the first place? Who defines them, categorizes them, and readies them for use on subsequent projects?**

9.1.3 Domain Analysis

Answers to these questions lie in *domain analysis*. *Firesmith [Fir93]* describes domain analysis in the following way:

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

9.1 Requirements Analysis (Cont'd)

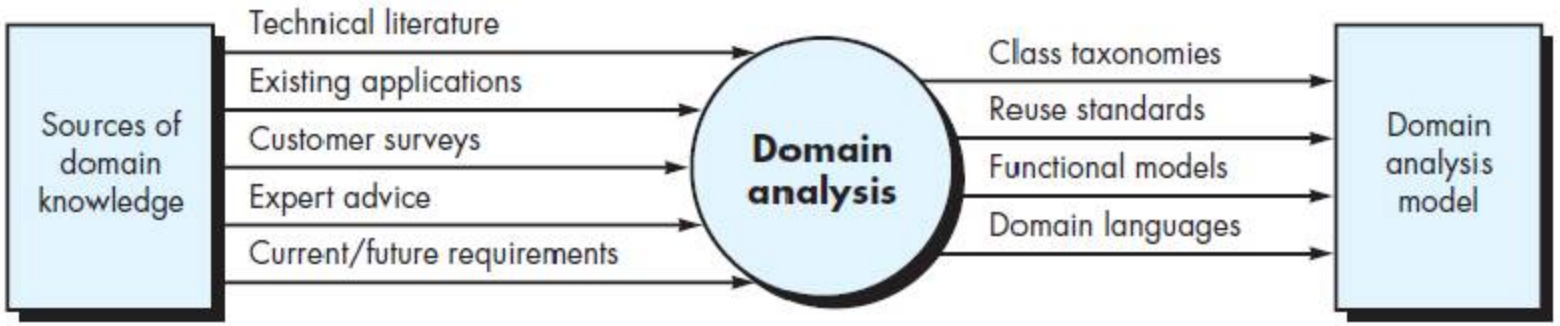
9.1.3 Domain Analysis

- "**Specific application domain**" can range from avionics to banking, from multimedia video games to software embedded within medical devices.
- **Goal:** find or create those **analysis classes and/or analysis patterns** that are broadly applicable so that they may be **reused.**
- Domain analysis may be viewed as an **umbrella activity** for the software process.
- Role of the **domain analyst is to discover and define analysis patterns,** analysis classes, and related information that may be used by many people working on similar but not necessarily the same applications.

9.1.3 Domain Analysis

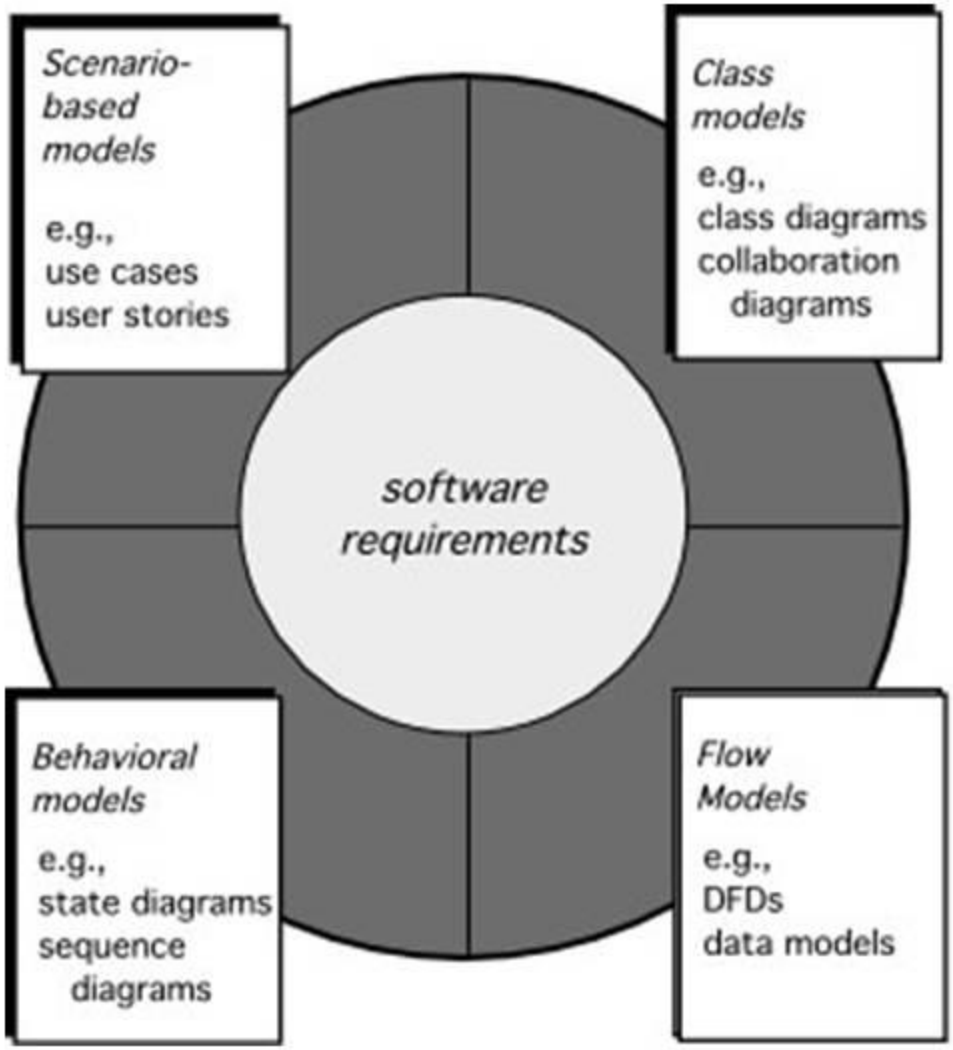
Figure 9.2 [Arn89] illustrates key inputs and outputs for the domain analysis process. Sources of domain knowledge are surveyed in an attempt to identify objects that can be reused across the domain

FIGURE 9.2 Input and output for domain analysis



9.1 Requirements Analysis (Cont'd)

9.1.4 Requirements Modeling Approaches



9.1 Requirements Analysis (Cont'd)

9.1.4 Requirements Modeling Approaches

A. ***Structured analysis, considers data (state)*** and the processes that transform the data as separate entities. Data objects are modeled in a way that defines their attributes and relationships. Processes that manipulate data objects are modeled in a manner that shows how they transform data as data objects flow through the system.

B. ***Object-oriented analysis,*** focuses on the definition of classes and the manner in which they collaborate with one another to effect customer requirements. UML and the Unified Process are predominantly object oriented.

9.1 Requirements Analysis (Cont'd)

9.1.4 Requirements Modeling Approaches

C. **Scenario-based elements** depict how the user interacts with the system and the specific sequence of activities that occur as the software is used. Class-based elements model the objects that the system will manipulate, the operations that will be applied to the objects to effect the manipulation, relationships (some hierarchical) between the objects, and the collaborations that occur between the classes that are defined. Behavioral elements depict how external events change the state of the system or the classes that reside within it.

D. Finally, **flow-oriented elements** represent the system as an information transform, depicting how data objects are transformed as they flow through various system functions.

Analysis modeling leads to the derivation of one or more of these modeling elements.