Chapter 8

- **Understanding Requirements**

Slide Set to accompany
Software Engineering: A Practitioner's Approach, 8/e
by Roger S. Pressman and Bruce R. Maxim

What is it?

- Before you begin any technical work, it's a good idea to create a set of requirements for any engineering tasks.
- These tasks lead to an understanding of what the business impact of the software will be, what the customer wants, and how end users will interact with the software.

Who does it?

- Software engineers (sometimes referred to as system engineers or "analysts" in the IT world) and other project stakeholders (managers, customers, and end users) all participate in requirements engineering.

Why is it important?

- Designing and building an elegant computer program that solves the wrong problem serves no one's needs.
- That's why it's important to understand what the customer wants before you begin to design and build a computer-based system.

What are the steps?

- Requirements engineering begins with inception (a task that defines the scope and nature of the problem to be solved).
- It moves onward to elicitation (a task that helps stakeholders define what is required).
- Then elaboration (where basic requirements are refined and modified).
- As stakeholders define the problem, negotiation occurs (what are the priorities, what is essential, when is it required?)
- Finally, the problem is specified in some manner and then reviewed or validated to ensure that your understanding of the problem and the stakeholders' understanding of the problem coincide.

Requirements Engineering-I

- Inception—ask a set of questions that establish …
  - basic understanding of the problem
  - the people who want a solution
  - the nature of the solution that is desired, and
  - the effectiveness of preliminary communication and collaboration between the customer and the developer
- Elicitation—It certainly seems simple enough—ask the customer, the users, and others what the objectives for the system or product are, what is to be accomplished, how the system or product fits into the needs of the business, and finally, how the system or product is to be used on a day-to-day basis. But it isn’t simple— it’s very hard. An important part of elicitation is to establish business goals

Requirements Engineering-II

- **Elaboration**—The information obtained from the customer during inception and elicitation is expanded and refined during elaboration. This task focuses on developing a refined requirements model that identifies various aspects of software function, behavior, and information. Elaboration is driven by the creation and refinement of user scenarios that describe how the end user (and other actors) will interact with the system.
- **Negotiation**—agree on a deliverable system that is realistic for developers and customers

Requirements Engineering-III

- Specification—can be any one (or more) of the following:
  - A written document
  - A set of models
  - A formal mathematical
  - A collection of user scenarios (use-cases)
  - A prototype
- Validation—a review mechanism that looks for
  - errors in content or interpretation
  - areas where clarification may be required
  - missing information
  - inconsistencies (a major problem when large products or systems are engineered)
  - conflicting or unrealistic (unachievable) requirements.

Requirements Engineering-IV

- **Requirements management**

Requirements for computer-based systems change, and the desire to change requirements persists throughout the life of the system.

Requirements management is a set of activities that help the project team identify, control, and track requirements and changes to requirements at any time as the project proceeds.

Inception

- Identify stakeholders
  - “who else do you think I should talk to?”
- Recognize multiple points of view
- Work toward collaboration
- The first questions
  - Who is behind the request for this work?
  - Who will use the solution?
  - What will be the economic benefit of a successful solution

A one or two page “product request” is generated during inception. A meeting place, time, and date are selected; a facilitator is chosen; and attendees from the software team and other stakeholder organizations are invited to participate. The product request is distributed to all attendees before the meeting date

Eliciting Requirements

- meetings are conducted and attended by both software engineers and customers
- rules for preparation and participation are established
- An agenda is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas
- a "facilitator" (can be a customer, a developer, or an outsider) controls the meeting
- a "definition mechanism" (can be work sheets, flip charts, or wall stickers or an electronic bulletin board, chat room or virtual forum) is used
- the goal is
  - to identify the problem
  - propose elements of the solution
  - negotiate different approaches, and
  - specify a preliminary set of solution requirements

Elicitation Work Products

- a statement of need and feasibility.
- a bounded statement of scope for the system or product.
- a list of customers, users, and other stakeholders who participated in requirements elicitation
- a description of the system's technical environment.
- a list of requirements (preferably organized by function) and the domain constraints that apply to each.
- a set of usage scenarios that provide insight into the use of the system or product under different operating conditions.
- any prototypes developed to better define requirements.

Quality Function Deployment

- *Quality function deployment* (QFD) is a quality management technique that translates the needs of the customer into technical requirements for software.
- QFD "concentrates on maximizing customer satisfaction from the software engineering process".
- To accomplish this, QFD emphasizes an understanding of what is valuable to the customer and then deploys these values throughout the engineering process.
- "Normal requirements" identify the objectives and goals that are stated for a product or system during meetings with the customer
- "Expected requirements" are implicit to the product or system and may be so fundamental that the customer does not explicitly state them.
- "Exciting requirements" go beyond the customer's expectations and prove to be very satisfying when present.

Quality Function Deployment

- **Function deployment** determines the “value” (as perceived by the customer) of each function required of the system
- **Information deployment** identifies data objects and events
- **Task deployment** examines the behavior of the system
- **Value analysis** determines the relative priority of requirements

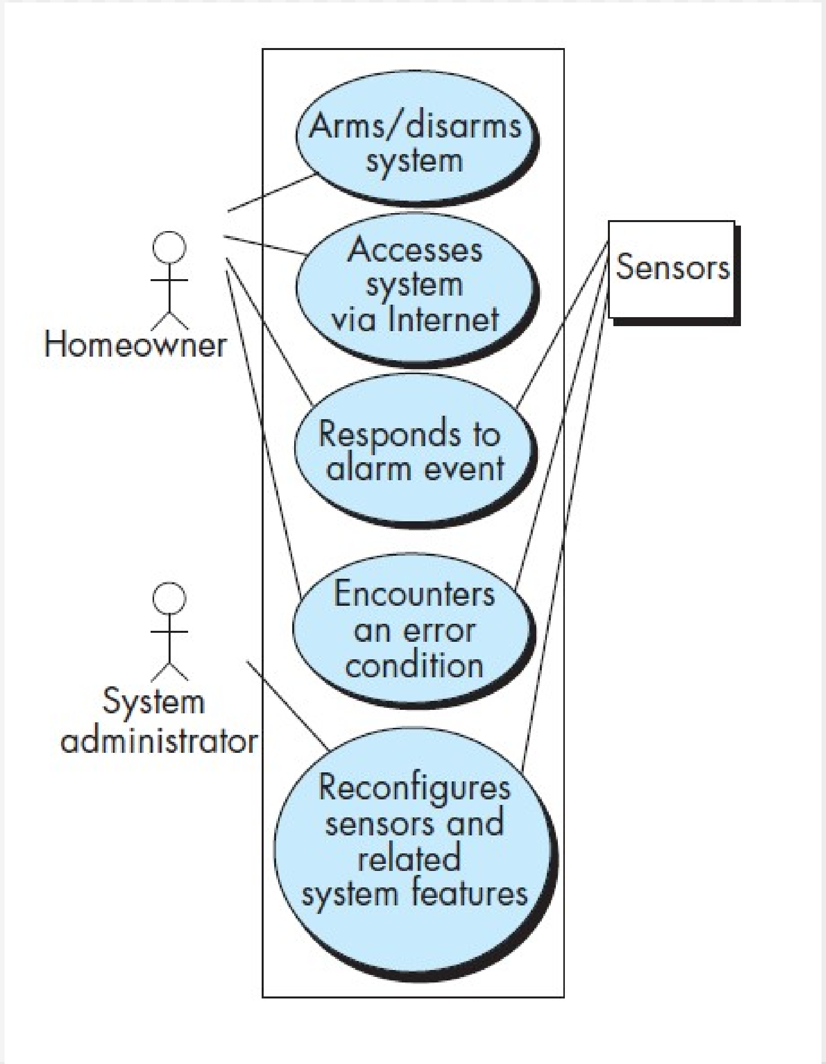
Non-Functional Requirements

- **Non-Functional Requirment (NFR)** – quality attribute, performance attribute, security attribute, or general system constraint. A two phase process is used to determine which NFR's are compatible:
  - The first phase is to create a matrix using each NFR as a column heading and the system SE guidelines a row labels
  - The second phase is for the team to prioritize each NFR using a set of decision rules to decide which to implement by classifying each NFR and guideline pair as complementary, overlapping, conflicting, or independent

Use-Cases

- A collection of user scenarios that describe the thread of usage of a system
- Each scenario is described from the point-of-view of an “actor”—a person or device that interacts with the software in some way
- Each scenario answers the following questions:
  - Who is the primary actor, the secondary actor (s)?
  - What are the actor’s goals?
  - What preconditions should exist before the story begins?
  - What main tasks or functions are performed by the actor?
  - What extensions might be considered as the story is described?
  - What variations in the actor’s interaction are possible?
  - 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


Arms/disarms system
Accesses system via Internet
Responds to alarm event
Encounters an error condition
Reconfigures sensors and related system features
Homeowner
System administrator
Sensors

Building the Analysis Model

- The intent of the analysis model is to provide a description of the required informational, functional, and behavioral domains for a computer-based system.
- The model changes dynamically as you learn more about the system to be built, and other stakeholders understand more about what they really require.
- For that reason, the analysis model is a snapshot of requirements at any given time.
- As the analysis model evolves, certain elements will become relatively stable, providing a solid foundation for the design tasks that follow

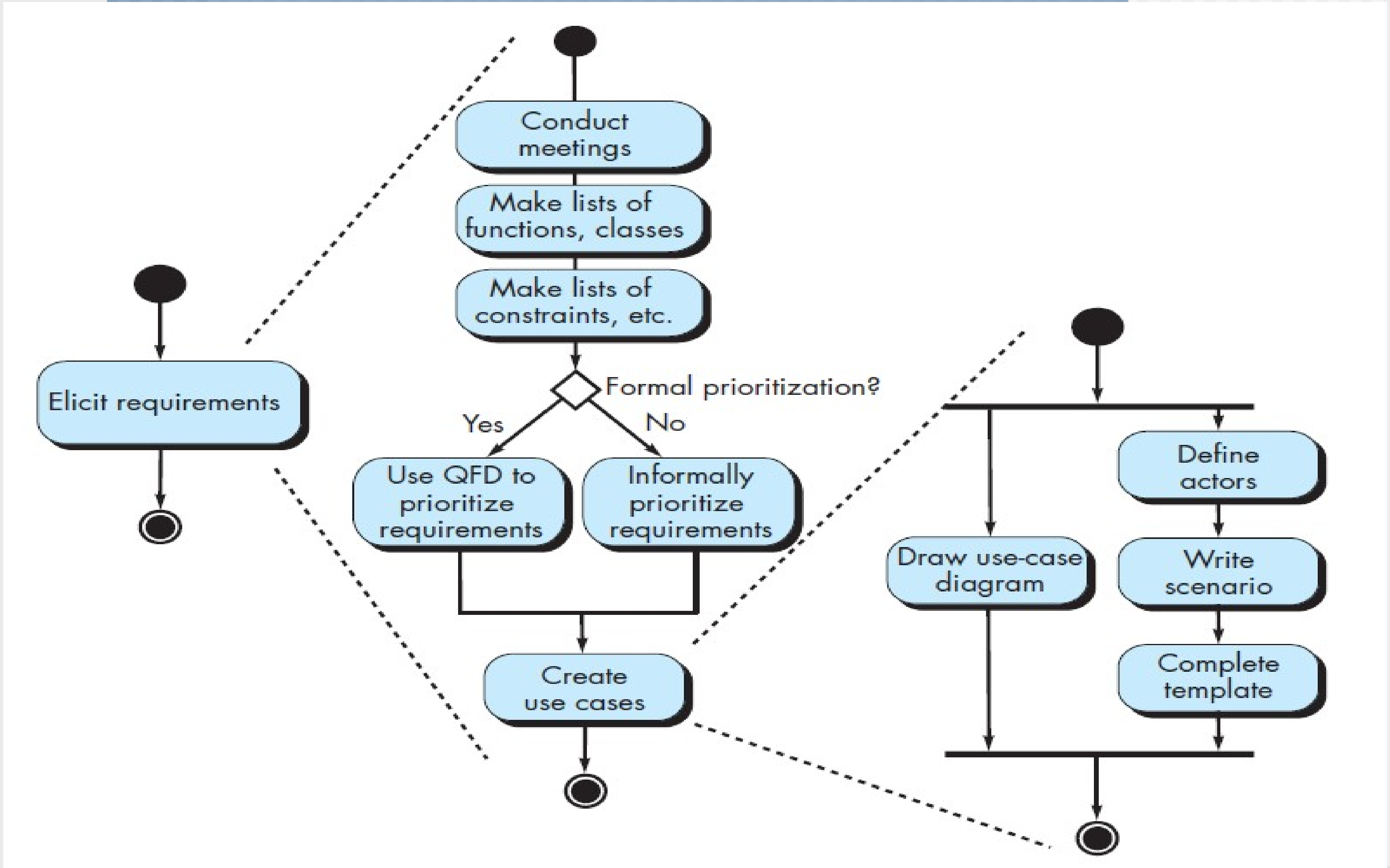
Building the Analysis Model

- Elements of the analysis model
  - Scenario-based elements
    - The system is described from the user's point of view using a scenario-based approach.
    - Use-case—descriptions of the interaction between an "actor" and the system
  - Class-based elements
    - Each usage scenario implies a set of objects that are manipulated as an actor interacts with the system. These objects are categorized into classes—a collection of things that have similar attributes and common behaviors

Building the Analysis Model

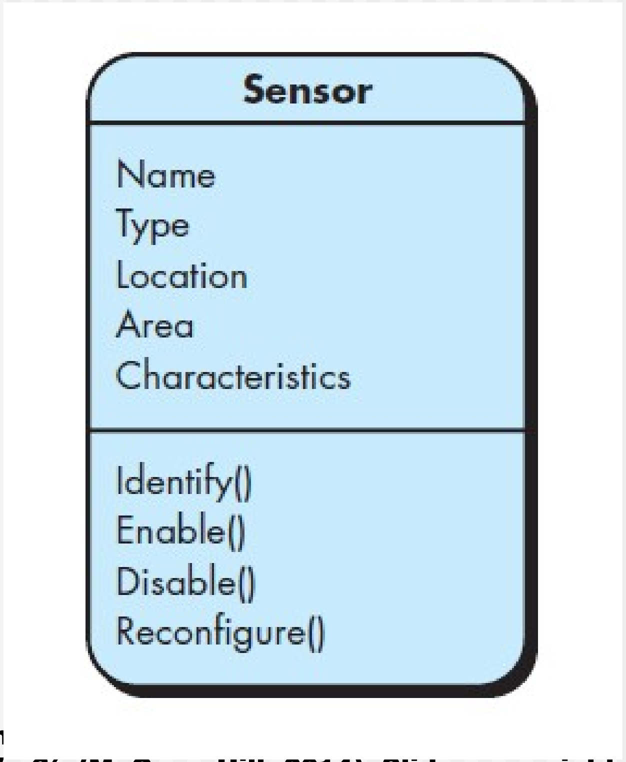
- Elements of the analysis model
  - Behavioral elements
    - State diagram
    - The behavior of a computer-based system can have a profound effect on the design that is chosen and the implementation approach that is applied.
    - The state diagram is one method for representing the behavior of a system by depicting its states and the events that cause the system to change state. A state is any observable mode of behavior. In addition, the state diagram indicates what actions (e.g., process activation) are taken as a consequence of a particular event.
  - Flow-oriented elements
    - Data flow diagram

Eliciting Requirements

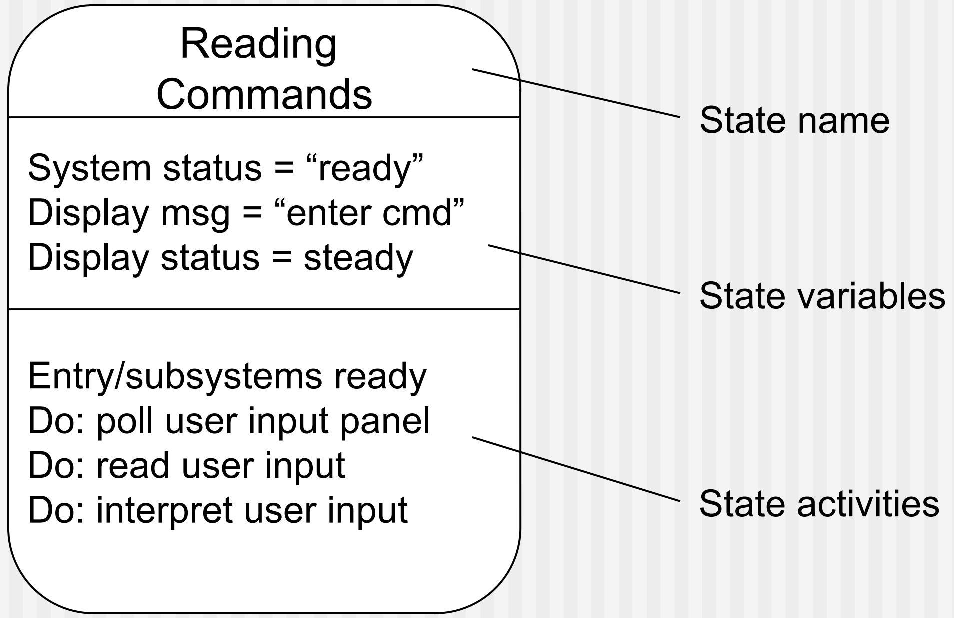


Class Diagram

From the *SafeHome* system …



State Diagram



Analysis Patterns

Pattern name: A descriptor that captures the essence of the pattern.

Intent: Describes what the pattern accomplishes or represents

Motivation: A scenario that illustrates how the pattern can be used to address the problem.

Forces and context: A description of external issues (forces) that can affect how the pattern is used and also the external issues that will be resolved when the pattern is applied.

Solution: A description of how the pattern is applied to solve the problem with an emphasis on structural and behavioral issues.

Consequences: Addresses what happens when the pattern is applied and what trade-offs exist during its application.

Design: Discusses how the analysis pattern can be achieved through the use of known design patterns.

Known uses: Examples of uses within actual systems.

Related patterns: On e or more analysis patterns that are related to the named pattern because (1) it is commonly used with the named pattern; (2) it is structurally similar to the named pattern; (3) it is a variation of the named pattern.

Negotiating Requirements

- **Identify the key stakeholders**
  - These are the people who will be involved in the negotiation
- **Determine each of the stakeholders "win conditions"**
  - Win conditions are not always obvious
- **Negotiate**
  - Work toward a set of requirements that lead to "win-win"

Requirements Monitoring

Especially needes in incremental development

- ***Distributed debugging*** **– uncovers errors and determines their cause.**
- ***Run-time verification*** **– determines whether software matches its specification.**
- ***Run-time validation*** **– assesses whether evolving software meets user goals.**
- ***Business activity monitoring*** **– evaluates whether a system satisfies business goals.**
- ***Evolution and codesign*** **– provides information to stakeholders as the system evolves.**