

# Chapter 5

---

## ■ Understanding Requirements

*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.

# 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**—elicit requirements from all stakeholders
- **Elaboration**—create an analysis model that identifies data, function and behavioral requirements
- **Negotiation**—agree on a deliverable system that is realistic for developers and customers

# Requirements Engineering-II

- **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 management**

# Inception

---

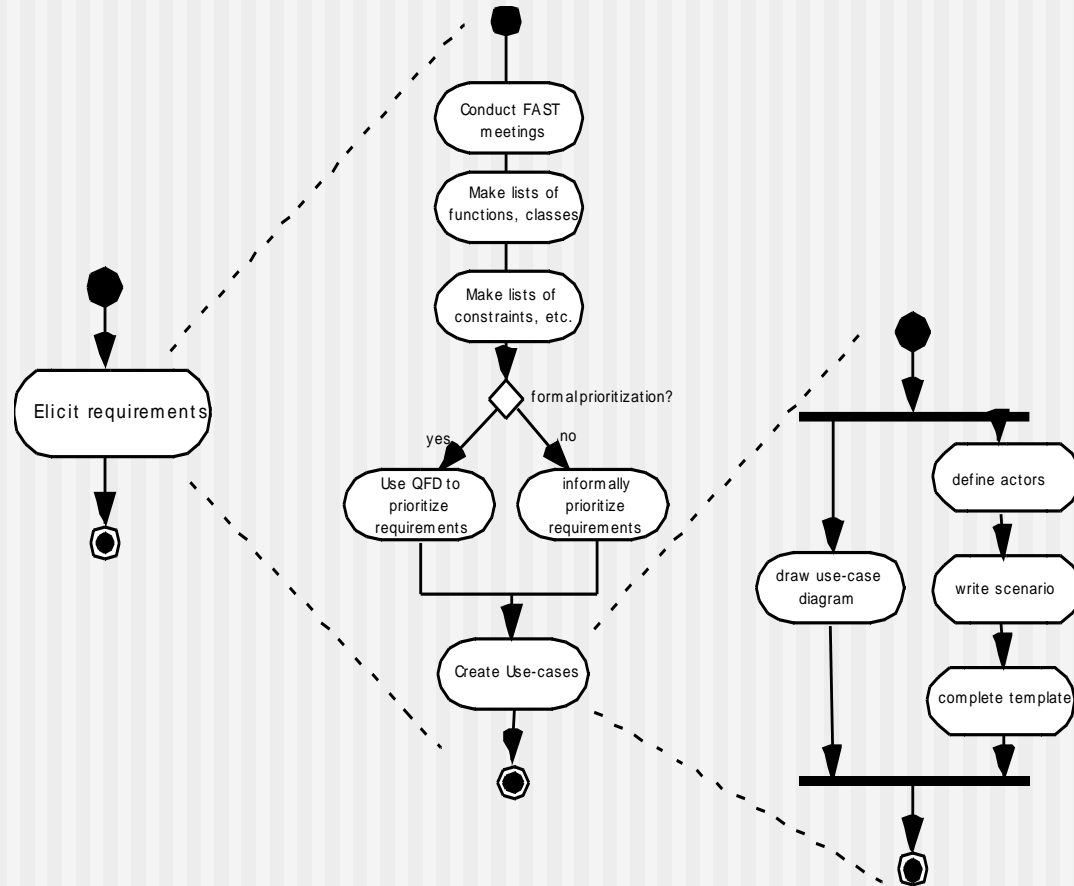
- Identify stakeholders
  - “who else do you think I should talk to?”
- Recognize multiple points of view
- Work toward collaboration
- The first questions
  - Who is behind the request for this work?
  - Who will use the solution?
  - What will be the economic benefit of a successful solution
  - Is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

# Eliciting Requirements

---

- meetings are conducted and attended by both software engineers and customers
- rules for preparation and participation are established
- an agenda is suggested
- 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

# Eliciting Requirements



# 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

# 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.



# Building the Analysis Model

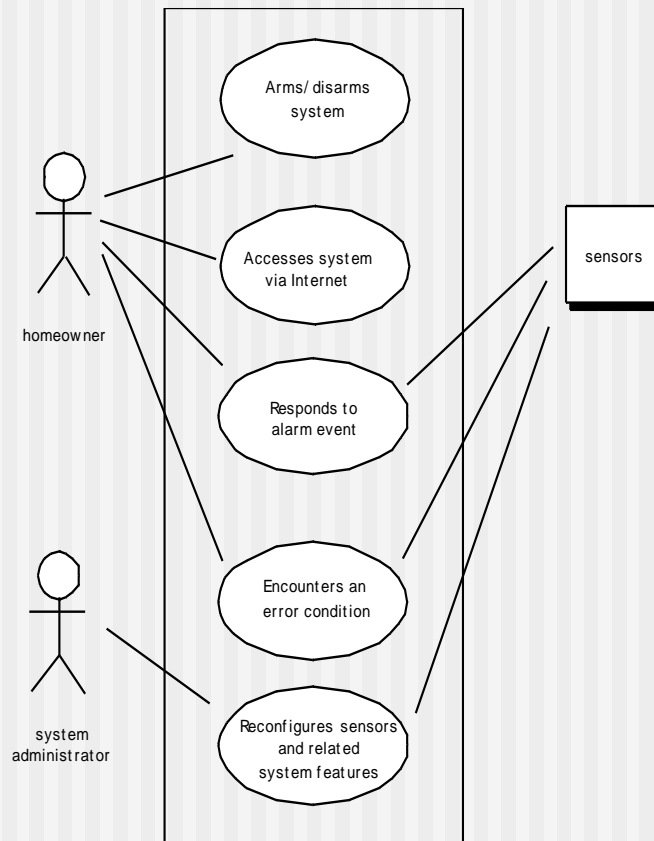
- Elements of the analysis model
  - Scenario-based elements
    - Functional—processing narratives for software functions
    - Use-case—descriptions of the interaction between an “actor” and the system
  - Class-based elements
    - Implied by scenarios
  - Behavioral elements
    - State diagram
  - Flow-oriented elements
    - Data flow diagram

# 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?

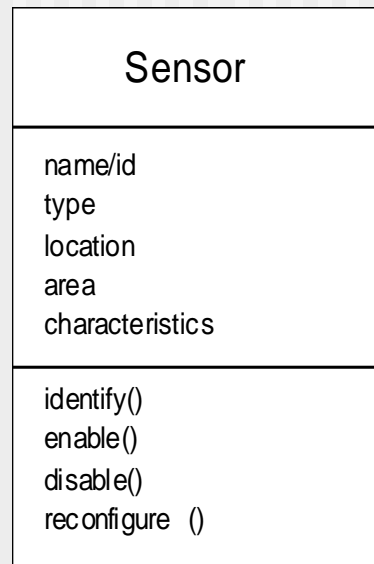
# Use-Case Diagram



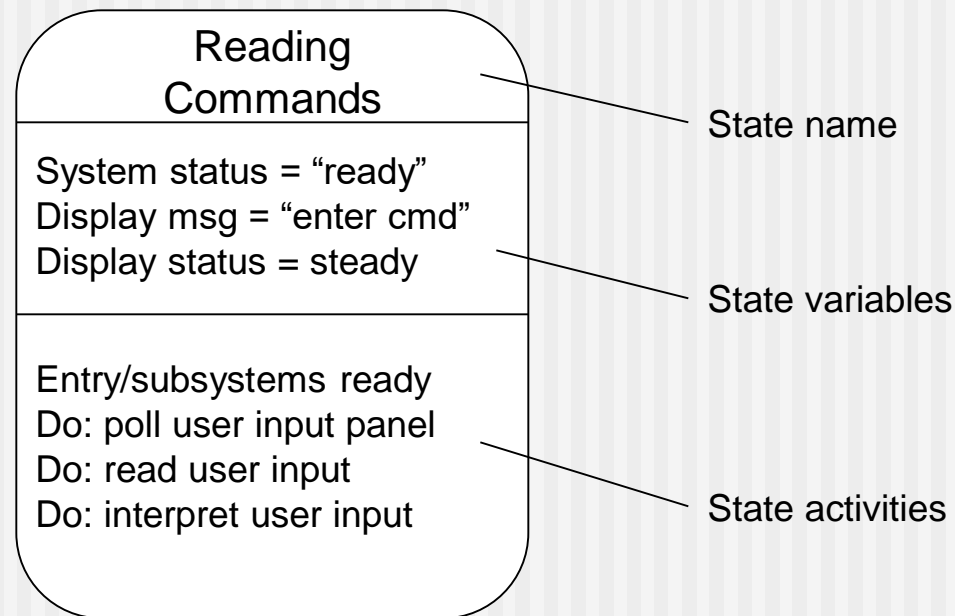
# Class Diagram

---

**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:** One 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”

# Validating Requirements - I

---

- Is each requirement consistent with the overall objective for the system/product?
- Have all requirements been specified at the proper level of abstraction? That is, do some requirements provide a level of technical detail that is inappropriate at this stage?
- Is the requirement really necessary or does it represent an add-on feature that may not be essential to the objective of the system?
- Is each requirement bounded and unambiguous?
- Does each requirement have attribution? That is, is a source (generally, a specific individual) noted for each requirement?
- Do any requirements conflict with other requirements?



# Validating Requirements - II

---

- Is each requirement achievable in the technical environment that will house the system or product?
- Is each requirement testable, once implemented?
- Does the requirements model properly reflect the information, function and behavior of the system to be built.
- Has the requirements model been “partitioned” in a way that exposes progressively more detailed information about the system.
- Have requirements patterns been used to simplify the requirements model. Have all patterns been properly validated? Are all patterns consistent with customer requirements?