Chapter 4 Objectives

- Eliciting requirements from the customers
- Modeling requirements
- Reviewing requirements to ensure their quality
- Documenting requirements for use by the design and test teams
4.1 The Requirements Process

- A requirement is an expression of desired behavior.

- A requirement deals with:
  - objects or entities
  - the state they can be in
  - functions that are performed to change states or object characteristics

- Requirements focus on the customer needs, not on the solution or implementation:
  - designate **what behavior**, without saying how that behavior will be realized.
Sidebar 4.1 Why Are Requirements Important?

- Top factors that caused project to fail
  - Incomplete requirements
  - Lack of user involvement
  - Unrealistic expectations
  - Lack of executive support
  - Changing requirements and specifications
  - Lack of planning
  - System no longer needed

- Some part of the requirements process is involved in almost all of these causes

- Requirements error can be expensive if not detected early
4.1 The Requirements Process

- Performed by the req. analyst or system analyst
- The final outcome is a Software Requirements Specification (SRS) document
4.1 The Requirements Process

Sidebar 4.2 Agile Requirements Modeling

● If requirements are tightly coupled and complex, we may be better off with a “heavy” process that emphasizes up-front modeling.

● If the requirements are uncertain, agile methods are an alternative approach.

● Agile methods gather and implement the requirements in increments.

● Extreme Programming (XP) is an agile process
  ▪ The requirements are defined as we build the system
  ▪ No planning or designing for possible future requirements
  ▪ Encodes the requirements as test cases that eventually implementation must pass.
4.2 Requirements Elicitation

- Customers do not always understand what their needs and problems are.
- It is important to discuss the requirements with everyone who has a stake in the system.
- Come up with agreement on what the requirements are.
  - If we can not agree on what the requirements are, then the project is doomed to fail.
4.2 Requirements Elicitation

Stakeholders

- **Clients**: pay for the software to be developed
- **Customers**: buy the software after it is developed
- **Users**: use the system
- **Domain experts**: familiar with the problem that the software must automate
- **Market Researchers**: conduct surveys to determine future trends and potential customers
- **Lawyers or auditors**: familiar with government, safety, or legal requirements
- **Software engineers** or other technology experts
4.2 Requirements Elicitation
Means of Eliciting Requirements

- Interviewing stakeholders
- Reviewing available documentations
- Observing the current system (if one exists)
- Apprenticing with users to learn about user's task in more details
- Interviewing user or stakeholders in groups
- Using domain specific strategies, such as Joint Application Design, or PIECES
- Brainstorming with current and potential users
4.3 Types of Requirements

- **Functional requirement**: describes required behavior in terms of required activities

- **Quality requirement** or **nonfunctional requirement**: describes some quality characteristic that the software must posses

- **Design constraint**: a design decision such as choice of platform or interface components

- **Process constraint**: a restriction on the techniques or resources that can be used to build the system
4.3 Types of Requirements

Resolving Conflicts

- Different stakeholder has different set of requirements
  - potential conflicting ideas

- Need to prioritize requirements

- Prioritization might separate requirements into three categories
  - essential: absolutely must be met
  - desirable: highly desirable but not necessary
  - optional: possible but could be eliminated
4.3 Types of Requirements

Two Kinds of Requirements Documents

- **Requirements definition**: a complete listing of everything the customer wants to achieve
  - Describing the entities in the environment where the system will be installed

- **Requirements specification**: restates the requirements as a specification of how the proposed system shall behave
4.4 Characteristics of Requirements

- Correct
- Consistent
- Unambiguous
- Complete
- Feasible
- Relevant
- Testable
- Traceable
4.5 Modeling Notations

- It is important to have standard notations for Modeling, documenting, and communicating decisions.

- Modeling helps us to understand requirements thoroughly:
  - Holes in the models reveal unknown or ambiguous behavior.
  - Multiple, conflicting outputs to the same input reveal inconsistencies in the requirements.
4.5 Modeling Notations

ER Diagrams Example: UML Class Diagram

- **UML (Unified Modeling Language)** is a collection of notations used to document software specifications and designs.

- It represents a system in terms of
  - objects: akin to entities, organized in classes that have an inheritance hierarchy
  - methods: actions on the object's variables

- The **class diagram** is the flagship model in any UML specification
  - A sophisticated ER diagram relating the classes (entities) in the specification
4.6 Requirements and Specification Languages
Unified Modeling Language (UML)

- Combines multiple notation paradigms
- Eight graphical modeling notations, and the OCL constrain language, including
  - Use-case diagram (a high-level DFD)
  - Class diagram (an ER diagram)
  - Sequence diagram (an event trace)
  - Collaboration diagram (an event trace)
  - Statechart diagram (a state-machine model)
  - OCL properties (logic)
4.6 Requirements and Specification Languages
Other Features of Requirement Notations

● Some techniques include notations
  ▪ for the degree of uncertainty or risk with each requirement
  ▪ for tracing requirements to other system documents such as design or code, or to other systems, such as when requirements are reused

● Most specification techniques have been automated to some degree
4.7 Prototyping Requirements
Building a Prototype

- To elicit the details of proposed system
- To solicit feedback from potential users about
  - what aspects they would like to see improve
  - which features are not so useful
  - what functionality is missing
- Determine whether the customer's problem has a feasible solution
- Assist in exploring options for optimizing quality requirements
4.7 Prototyping Requirements

Prototyping Example

- Prototype for building a tool to track how much a user exercises each day

- Graphical representation of first prototype, in which the user must type the day, month and year

![Graphical representation of first prototype]
4.7 Prototyping Requirements

Prototyping Example (continued)

- Second prototype shows a more interesting and sophisticated interface involving a calendar
  - User uses a mouse to select the month and year
  - The system displays the chart for that month, and the user selects the appropriate date in the chart
4.7 Prototyping Requirements

Prototyping Example (continued)

- Third prototype shows that instead of calendar, the user is presented with three slide bars
  - User uses the mouse to slide each bar left or right
  - The box at the bottom of the screen changes to show the selected day, month, and year
4.7 Prototyping Requirements

Approaches to Prototyping

● **Throwaway approach**
  - Developed to learn more about a problem or a proposed solution, and that is never intended to be part of the delivered software
  - Allow us to write “quick-and-dirty”

● **Evolutionary approach**
  - Developed not only to help us answer questions but also to be incorporated into the final product
  - Prototype has to eventually exhibit the quality requirements of the final product, and these qualities cannot be retrofitted

● **Both techniques are sometimes called rapid prototyping**
4.7 Prototyping Requirements

Prototyping vs. Modeling

- **Prototyping**
  - Good for answering questions about the user interfaces

- **Modeling**
  - Quickly answer questions about constraints on the order in which events should occur, or about the synchronization of activities
4.8 Requirements Documentation

Requirement Definition: Steps Documenting Process

- Outline the general purpose and scope of the system, including relevant benefits, objectives, and goals
- Describe the background and the rationale behind proposal for new system
- Describe the essential characteristics of an acceptable solution
- Describe the environment in which the system will operate
- Outline a description of the proposal, if the customer has a proposal for solving the problem
- List any assumptions we make about how the environment behaves
4.8 Requirements Documentation

Requirements Specification: Steps Documenting Process

- Describe all inputs and outputs in detail, including
  - the sources of inputs
  - the destinations of outputs,
  - the value ranges
  - data format of inputs and output data
  - data protocols
  - window formats and organizations
  - timing constraint

- Restate the required functionality in terms of the interfaces' inputs and outputs

- Devise fit criteria for each of the customer's quality requirements
4.8 Requirements Documentation
Sidebar 4.6 Level of Specification

- Survey shows that one of the problems with requirement specifications was the uneven level of specification
  - Different writing styles
  - Difference in experience
  - Different formats
  - Over specifying requirements
  - Underspecifying requirements

- Recommendations to reduce unevenness
  - Write each clause so that it contains only one requirement
  - Avoid having one requirement refer to another requirement
  - Collect similar requirements together
4.8 Requirements Documentation

Sidebar 4.7 Hidden Assumptions

- Two types of environmental behavior of interest
  - desired behavior to be realized by the proposed system
  - existing behavior that is unchanged by the proposed system
    - often called assumptions or domain knowledge

- Most requirements writers consider assumptions to be simply the conditions under which the system is guaranteed to operate correctly
4.8 Requirements Documentation
IEEE Standard for SRS Organized by Objects

1. Introduction to the Document
1.1 Purpose of the Product
1.2 Scope of the Product
1.3 Acronyms, Abbreviations, Definitions
1.4 References
1.5 Outline of the rest of the SRS

2. General Description of Product
2.1 Context of Product
2.2 Product Functions
2.3 User Characteristics
2.4 Constraints
2.5 Assumptions and Dependencies

3. Specific Requirements
3.1 External Interface Requirements
3.1.1 User Interfaces
3.1.2 Hardware Interfaces
3.1.3 Software Interfaces
3.1.4 Communications Interfaces
3.2 Functional Requirements
3.2.1 Class 1
3.2.2 Class 2
...
3.3 Performance Requirements
3.4 Design Constraints
3.5 Quality Requirements
3.6 Other Requirements

4. Appendices
4.8 Requirements Documentation
Process Management and Requirements Traceability

● Process management is a set of procedures that track
  ▪ the requirements that define what the system should do
  ▪ the design modules that are generated from the requirement
  ▪ the program code that implements the design
  ▪ the tests that verify the functionality of the system
  ▪ the documents that describe the system

● It provides the threads that tie the system parts together
4.8 Requirements Documentation

Development Activities

- Horizontal threads show the coordination between development activities
4.9 Validation and Verification

• In requirements validation, we check that our requirements definition accurately reflects the customer's needs.

• In verification, we check that one document or artifact conforms to another.

• Verification ensures that we build the system right, whereas validation ensures that we build the right system.
### 4.9 Validation and Verification

List of techniques to validate requirements

<table>
<thead>
<tr>
<th>Validation</th>
<th>Walkthroughs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Readings</td>
</tr>
<tr>
<td></td>
<td>Interviews</td>
</tr>
<tr>
<td></td>
<td>Reviews</td>
</tr>
<tr>
<td></td>
<td>Checklists</td>
</tr>
<tr>
<td></td>
<td>Models to check functions and relationships</td>
</tr>
<tr>
<td></td>
<td>Scenarios</td>
</tr>
<tr>
<td></td>
<td>Prototypes</td>
</tr>
<tr>
<td></td>
<td>Simulation</td>
</tr>
<tr>
<td></td>
<td>Formal inspections</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verification</th>
<th>Cross-referencing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simulation</td>
</tr>
<tr>
<td></td>
<td>Consistency checks</td>
</tr>
<tr>
<td></td>
<td>Completeness checks</td>
</tr>
<tr>
<td></td>
<td>Check for unreachable states of transitions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Checking</th>
<th>Model checking</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mathematical proofs</td>
</tr>
</tbody>
</table>
4.9 Validation and Verification

Requirements Review

- Review the stated goals and objectives of the system
- Compare the requirements with the goals and objectives
- Review the environment in which the system is to operate
- Review the information flow and proposed functions
- Assess and document the risk, discuss and compare alternatives
- Testing the system: how the requirements will be revalidated as the requirements grow and change
4.9 Validation and Verification

Sidebar 4.8 Number of Requirements Faults

- Jone and Thayes's studies show that
  - 35% of the faults to design activities for project of 30,000-35,000 delivered source instructions
  - 10% of the faults to requirements activities and 55% of the faults to design activities for projects of 40,000-80,000 delivered source instructions
  - 8% to 10% of the faults to requirements activities and 40% to 55% of the faults to design activities for project of 65,000-85,000 delivered source instructions

- Basili and Perricone report
  - 48% of the faults observed in a medium-scale software project were attributed to “incorrect or misinterpreted functional specification or requirements”

- Beizer attributes 8.12% of the faults in his samples to problems in functional requirements
4.9 Validation and Verification

Verification

- Check that the requirements-specification document corresponds to the requirements-definition
- Make sure that if we implement a system that meets the specification, then the system will satisfy the customer's requirements
- Ensure that each requirement in the definition document is traceable to the specification
Model checking is an exhaustive search for a specification's execution space, to determine whether some temporal-logic property holds of the execution

- Atlee (1996) used the SMV model checker to verify five properties of an SCR specification of the A-7 naval aircraft

A theorem prover uses a collection of built-in theories, inference rules, and decision procedures for determining whether a set of asserted facts logically entails some unasserted fact

- Dutertre and Stavridou (1997) used theorem prover PVS to verify some of the functional and safety requirements of an avionic system
4.10 Measuring Requirements

- Measurements focus on three areas
  - product
  - process
  - resources

- Number of requirements can give us a sense of the size of the developed system

- Number of changes to requirements
  - Many changes indicate some instability or uncertainty in our understanding of the system

- Requirement-size and change measurements should be recorded by requirements type
4.10 Measuring Requirements

Rating Scheme on Scale from 1 to 5

1. You understand this requirement completely, have designed systems from similar requirements, and have no trouble developing a design from this requirement.

2. Some elements of this requirement are new, but they are not radically different from requirements that have been successfully designed in the past.

3. Some elements of this requirement are very different from requirements in the past, but you understand the requirement and can develop a good design from it.

4. You cannot understand some parts of this requirement, and are not sure that you can develop a good design.

5. You do not understand this requirement at all, and cannot develop a design.
4.10 Measuring Requirements
Testers/Designers Profiles

- Figure (a) shows profiles with mostly 1s and 2s
  - The requirements are in good shape

- Figure (b) shows profiles with mostly 4s and 5s
  - The requirements should be revised
4.11 Choosing a Specification Technique
Criteria for Evaluating Specification Methods

- Applicability
- Implementability
- Testability/Simulation
- Checkability
- Maintainability
- Modularity
- Level of abstraction
- Soundness
- Veriability
- Run time safety
- Tools maturity
- Looseness
- Learning curve
- Technique maturity
- Data modeling
- Discipline
Steps

1. Determine which of the criteria are especially important
2. Evaluate each of the candidate techniques with respect to the criteria
3. Choose a specification technique that best supports the criteria that are most important to the problem
## Important of Specification Criteria During Reactive-System Life Cycle

- **R**=Requirements, **D**=Design, **I**=Implementation, **T**=Testing, **M**=Maintenance, **O**=Other

<table>
<thead>
<tr>
<th>R</th>
<th>D</th>
<th>I</th>
<th>T</th>
<th>M</th>
<th>O</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td>Applicability</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td>Implementability</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td>Testability</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td>Checkability</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td>Maintainability</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td>Modularity</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Level of abstraction</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Soundness</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Verifiability</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Runtime safety</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tools maturity</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Looseness</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Learning curve</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Technique maturity</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data modeling</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Discipline</td>
</tr>
</tbody>
</table>