CHAPTER 7 REQUIREMENTS ENGINEERING

Overview

- Requirements engineering helps software engineers better understand the problems they are trying to solve.
- Building an elegant computer solution that ignores the customer's needs helps no one.
- It is very important to understand the customer's wants and needs before you begin designing or building a computer-based solution.
- The requirements engineering process begins with inception, moves on to elicitation, elaboration, negotiation, problem specification, and ends with review or validation of the specification.
- The intent of requirements engineering is to produce a written understanding of the customer's problem.
- Several different work products might be used to communicate this understanding (user scenarios, function and feature lists, analysis models, or specifications).

Requirements Engineering

- Must be adapted to the needs of a specific process, project, product, or people doing the work.
- Begins during the software engineering communication activity and continues into the modeling activity.
- In some cases requirements engineering may be abbreviated, but it is never abandoned.
- It is essential that the software engineering team understand the requirements of a problem before the team tries to solve the problem.

Requirements Engineering Tasks

- Inception (software engineers use context-free questions to establish a basic understanding of the problem, the people who want a solution, the nature of the solution, and the effectiveness of the collaboration between customers and developers)
- Elicitation (find out from customers what the product objectives are, what is to be done, how the product fits into business needs, and how the product is used on a day to day basis)
- Elaboration (focuses on developing a refined technical model of software functions, features, and constraints using the information obtained during inception and elicitation)
- Negotiation (requirements are categorized and organized into subsets, relations among requirements identified, requirements reviewed for correctness, requirements prioritized based on customer needs)
- Specification (written work products produced describing the function, performance, and development constraints for a computer-based system)
• Requirements validation (formal technical reviews used to examine the specification work products to ensure requirement quality and that all work products conform to agreed upon standards for the process, project, and products)

Requirements Management

• Set of activities that help project team to identify, control, and track requirements and changes as project proceeds
• Many of these activities are identical to those that make up the software configuration management (SCM) process
• Requirements are first identified, tagged with a unique identifier and classified by type (functional, data, behavioral, interface, or output)
• Traceability tables (e.g., features, source, dependency, subsystem, interface) are developed and updated any time a requirement is modified
• Database systems are invaluable in helping software teams track requirement changes

Traceability Tables

• Features traceability table (shows how requirements relate to customer observable features)
• Source traceability table (identifies source of each requirement)
• Dependency traceability table (indicate relations among requirements)
• Subsystem traceability table (requirements categorized by subsystem)
• Interface traceability table (shows requirement relations to internal and external interfaces)

Initiating Requirements Engineering Process

• Identify stakeholders
• Recognize the existence of multiple stakeholder viewpoints
• Work toward collaboration among stakeholders
• These context-free questions focus on customer, stakeholders, overall goals, and benefits of the system
  o Who is behind the request for work?
  o Who will use the solution?
  o What will be the economic benefit of a successful solution?
  o Is there another source for the solution needed?
• The next set of questions enable developer to better understand the problem and the customer's perceptions of the solution
  o How would you characterize good output form a successful solution?
  o What problem(s) will this solution address?
  o Can you describe the business environment in which the solution will be used?
  o Will special performance constraints affect the way the solution is approached?
- The final set of questions focuses on communication effectiveness
  - Are you the best person to give "official" answers to these questions?
  - Are my questions relevant to your problem?
  - Am I asking too many questions?
  - Can anyone else provide additional information?
  - Should I be asking you anything else?

**Eliciting Requirements**

- Collaborative requirements gathering
  - Meetings attended by both developers and customers
  - Rules for preparation and participation are established
  - Flexible agenda is used
  - Facilitator controls the meeting
  - Definition mechanism (e.g., stickers, flip sheets, electronic bulletin board) used to gauge group consensus
  - Goal is to identify the problem, propose solution elements, negotiate approaches, and specify preliminary set of solutions requirements

- Quality function deployment (QFD)
  - Identifies three types of requirements (normal, expected, exciting)
  - In customer meetings function deployment is used to determine value of each function that is required for the system
  - Information deployment identifies both data objects and events that the system must consume or produce (these are linked to functions)
  - Task deployment examines the system behavior in the context of its environment
  - Value analysis is conducted to determine relative priority of each requirement generated by the deployment activities

- User-scenarios
  - Also known as use-cases, describe how the system will be used
  - Developers and users create a set of usage threads for the system to be constructed

**Elicitation Work Products**

- Statement of need and feasibility
- Bounded statement of scope for system or product
- List of stakeholders involved in requirements elicitation
- Description of system's technical environment
- List of requirements organized by function and applicable domain constraints
- Set of usage scenarios (use-cases) that provide use insight into operation of deployed system
- Prototypes developed to better understand requirements

**Developing Use-Cases**
Each use-case tells stylized story about how end-users interact with the system under a specific set of circumstances.

First step is to identify **actors** (people or devices) that use the system in the context of the function and behavior of the system to be described:

- Who are the primary or secondary actors?
- What preconditions must exist before story begins?
- What are the main tasks or functions performed by each actor?
- What extensions might be considered as the story is described?
- What variations in actor interactions are possible?
- What system information will the actor acquire, produce, or change?
- Will the actor need to inform the system about external environment changes?
- What information does the actor desire from the system?
- Does the actor need to be informed about unexpected changes?

**Use-case template**

- Name
- Primary actor
- Goal in context
- Preconditions
- Trigger
- Scenario details
- Extensions
- Priority
- When available
- Frequency of use
- Channels to secondary actors
- Open issues

**Use-case Drawbacks**

- Lack of formality in use-case descriptions
- Not all systems have explicitly defined actors
- Use-cases are not inherently object-oriented
- Developers have a tendency to functionally decompose use-cases.

**Analysis Model**

- Intent is to provide descriptions of required information, functional, and behavioral domains for computer-based systems

**Analysis Model Elements**

- Scenario-based elements (describe system from user perspective)
- Class-based elements (relationships among objects manipulated by actors and their attributes)
- Behavioral elements (depict system and class behavior as states and transitions between states)
- Flow-oriented elements (shows how information flows through the system and is transformed by the system functions)

Many different representations can be used to depict the analysis model.
- Use-case diagrams
- Activity diagrams
- Class diagrams
- State diagram
- Data flow diagram (DFD)

- Analysis Pattern Template
  - Name
  - Intent
  - Motivation
  - Forces and context
  - Solution
  - Consequences
  - Design
  - Known use examples
  - Related patterns

Negotiating Requirements

- Negotiation activities
  - Identification of system key stakeholders
  - Determination of stakeholders' "win conditions"
  - Negotiate to reconcile stakeholders' win conditions into "win-win" result for all stakeholders (including developers)

- Key points
  - It's not a competition
  - Map out a strategy
  - Listen actively
  - Focus on other party's interests
  - Don't let it get personal
  - Be creative
  - Be ready to commit

Requirement Review (Validation)

- Is each requirement consistent with overall project or system objective?
- Are all requirements specified at the appropriate level off abstraction?
- Is each requirement essential to system objective or is it an add-on feature?
- Is each requirement bounded and unambiguous?
- Do you know the source for each requirement?
- Do requirements conflict with one another?
- Is the requirement achievable in the proposed technical environment for the system or product?
- Is each requirement testable?
- Does the requirements model reflect the information, function, and behavior of the system to be built?
• Has the requirements model been partitioned in a way that exposes more detailed system information progressively?
• Have all the requirements patterns been properly validated and are they consistent with customer requirements?