Sofware Requirements Engineeing

Three main tasks in RE:

1. *Elicit* — find out what the customers really want.
   - Identify stakeholders, their goals and viewpoints.
2. *Document* — write it down (Requirements Specification).
   - Understandable
   - Precise
   - Complete
   - Consistent
   - Unambiguous
   - Modifiable
   - Correct
3. *Analyze* — check for above qualities.

Requirements Elicitation

Key problem areas:

Scope — what is the boundary of the system?

Understanding

- What is needed
- What is possible/practical
- Difficulties in communication of concepts/ideas
- Domain knowledge
- Conflicting needs or priorities

Volatility — Requirements change over time.

Collaborative Requirements Gathering

- Meetings with designers, customers and other stakeholders.
- Rules for preparation & participation.
- Agenda.
- Facilitator.
- "Definitions mechanism" (work sheet, flip charts, wall stickers etc.)
- Goals:
  - identify the problem,
  - propose elements of the solution,
  - negotiate different approaches,
  - specify preliminary set of solution requirements.

Example sequence

1. Stakeholders write and distribute "product request"
2. Pre-meeting, each participant identifies
   - objects in system
   - objects in environment
   - services (processes or functions)
   - constraints
   - performance criteria
3. Meeting
   - present lists
   - combine, move etc. to agree on consensus lists
   - refine definitions for each word or phrase (objects etc.)
   - maintain list of issues raised that aren't resolved.

Requirements Specification

Essential parts of an SRS:[1]

1. Introduction
    1. Purpose — of this document, including intended audience
    2. Scope
        1. what will be produced
        2. what will it do
        3. application, benefits, objectives and goals
    3. Definitions, acronyms and abbreviations
    4. References
    5. Overview — what is to follow and how is it organized

[1] From IEEE Std 830-1998 Software Requirements Specifications

SRS Essential parts (cont'd)

2. Overall description — general factors that affect the product (not detailed requirements).
   1. Product perspective — how does it relate to other products?
      - Interfaces (system, user, hardware, other software, communications)
      - Memory constraints
      - Operation constraints (could be part of UI)
      - Site adaption requirements — how will product be adapted for specific sites?
   2. Product functions — description of major functions. Organized in logical way (for customer).
   3. User characteristics — what do we know about the intended users?
   4. Constraints — any other items that will limit design decisions.
   5. Assumptions and dependencies
   6. Apportioning of requirements — what may be delayed until later?

SRS Essential parts (cont'd)

3. Specific Requirements
   - All requirements
   - Detailed enough to enable designers to design a satisfactory system and for testers to test it.
   - All must be externally perceivable (e.g., by users, operators or other systems).
   - Must include description of every input/stimulus and output/response and their relations.
   - Each requirement should be uniquely identifiable.
   - Each requirement should be verifiable.
   - Organized for readability.

Use Cases

- A technique for capturing functional requirements of system.
- Describe typical interactions between users (*actors*[2]) and the system.
- UML does not specify a standard presentation format or notation for Use Cases.
- Note that use cases are quite distinct from use case diagrams, which illustrate relationships between use cases and actors.

[2] An *actor* is actually a role that a user may play w.r.t. the system.

Scenario Example

Consider first a *scenario*—a particular sequence of events in the interaction between the user and the system.
Consider Automated Banking Machine example:

> **Goal**: *Withdraw money.*
> *The customer inserts the bank card in the ATM card slot and is prompted to enter her personal identification number. The system checks that the PIN is correct and then prompts the customer to select a transaction type, account and amount. The user selects a withdrawal. The system verifies that the withdrawal is permitted (e.g., funds are available), dispenses the money and returns the card.*

Senario (cont'd)

- There are many possible variations on this sequence (e.g., PIN is not correct) so there will be many scenarios with the same goal.
- A use case describes a set of scenarios with a common goal.

See: Transaction

Use Case tips

- Always keep in terms of external view (problem domain).
- Name should be short active verb phrase.
- Steps clearly describe action by actor or system.
- Other information:

  entry condition(s) Condition(s) that must be true before a use case can begin. Also known as *pre-condition*.

  exit condition(s) Condition(s) that the system will ensure is true at the end of the use case. Also known as *guarantee* or *post-condition*

  trigger Event that gets the use case started.
- See also useful links on the web page.