Chapter Four

Requirements Specification

Introduction

- A software requirements specification (SRS) is a description of a software system to be developed.
  - its defined after business requirements specification (CONOPS):
    - a document describing the characteristics of a proposed system from the viewpoint of an individual who will use that system.
  - also called **stakeholder requirements specification** (StRS).
  - other document related is the **system requirements specification** (SyRS).
- The SRS lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

Cont'd…

- **Main aim of requirements specification:**
  - systematically organize the requirements arrived during requirements analysis.
  - document requirements properly.
- The SRS document is useful in various contexts:
  - statement of user needs
  - contract document
  - reference document
  - definition for implementation

Software Requirements Specification : A Contract Document

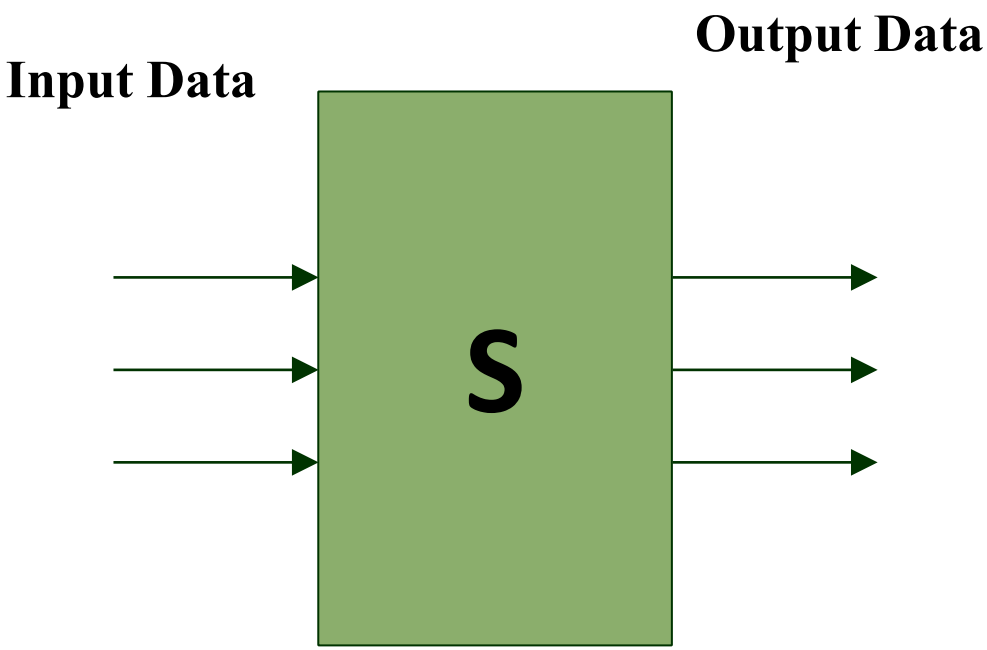
- Requirements document is a reference document.
- SRS document is a contract between the development team and the customer.
  - Once the SRS document is approved by the customer,
    - any subsequent controversies are settled by referring the SRS document.
- Once customer agrees to the SRS document:
  - development team starts to develop the product according to the requirements recorded in the SRS document.

❑The final product will be acceptable to the customer:

- as long as it satisfies all the requirements recorded in the SRS document.

Cont'd…

- The SRS document is known as <u>black-box specification:</u>
  - the system is considered as a black box whose internal details are not known.
  - only its visible external (i.e. input/output) behaviour is documented.



Cont'd…

- SRS document concentrates on:
  - what needs to be done.
  - carefully avoids the solution ("how to do") aspects.
- The SRS document serves as a contract
  - between development team and the customer.
  - Should be carefully written.
- The requirements at this stage:
  - written using end-user terminology.
- If necessary:
  - later a formal requirement specification may be developed from it.

Cont'd…

- SRS document, normally contains three important parts:
  - Functional requirements,
  - Non-functional requirements,
  - Constraints on the system.

Organization of SRS Document

- Introduction
- Functional Requirements
- Non-functional Requirements
  - External interface requirements
  - Performance requirements
- Constraints

Requirements smells/Code Smells

- **Requirements smells** has been proposed to describe issues in requirements specification where **the requirement** is not necessarily wrong but **could be problematic**.

❖In particular, the requirements smell:

- is an indicator for a quality problem of a requirements artifact.
- does not necessarily lead to a defect and, thus, has to be judged by the context.
- has a concrete location in the requirements artifact itself, e.g. a word or a sequence.
- has a concrete detection mechanism (which can be automatic or manual and more or less accurate).

Cont'd…

- Examples of requirements smells are :
  - Subjective Language (feel, believe, think) , Ambiguous Adverbs (Quickly, gently) and Adjectives (Small, sharp) , Superlatives and Negative Statements.

❖In computer programming, **code smell**, (**software smell** or **bad smell**) is **any symptom in the source code of a program** that possibly indicates a deeper problem.

- One way to look at smells is with respect to principles and quality:
  - "**smells** are certain structures in the code that indicate violation of fundamental design principles and negatively impact design quality".

Cont'd…

- **Code smells** are usually not bugs—they are not technically incorrect and do not currently prevent the program from functioning.
  - Instead, they indicate weaknesses in design that may be slowing down development or increasing the risk of bugs or failures in the future.

Techniques to write high-quality requirements

- Many software requirements specifications (SRS) are filled with badly written requirements.
  - Because the quality of any product depends on the **quality of the raw materials** fed into it,
    - poor requirements cannot lead to excellent software.
- Sadly, few software developers have been educated about how to elicit, analyze, document, and verify the quality of requirements.

❑ No matter how much you scrub, analyze, review, and refine the requirements, **they will never be perfect.**

➢ However, if you keep the following characteristics in mind, you will produce better requirements documents and you will build better products

Characteristics of Quality Requirement Statements

- How can we distinguish good software requirements from those that have problems?

❖ Six characteristics individual requirement statements should exhibit:

1) Correct
2) Feasible
3) Necessary
4) Prioritized
5) Unambiguous
6) verifiable

Cont'd…

I. Correct

- Each requirement must accurately describe the functionality to be delivered.
- **The reference for correctness** is the source of the requirement, such as an actual customer or a higher-level system requirements specification.

➢ Only user representatives can determine the correctness of user requirements, which is why it is essential to include them, or their close surrogates, in inspections of the requirements.

- Requirements inspections that do not involve users can lead to developers saying, "**That doesn't make sense**."
  - This is probably what they meant." This is also known as "guessing."

Cont'd…

II. Feasible

- It must be possible to implement each requirement within the known capabilities and limitations of the system and its environment.
  - To avoid infeasible requirements, have a developer work with the requirements analysts or marketing personnel throughout the elicitation process.

III. Necessary

- Each requirement should document something the customers really need.

Cont'd…

IV. Prioritized

- Assign an implementation priority to each requirement, feature, or use case to indicate how essential it is to include it in a particular product release.

➢ **Three levels of priority:**

- **High priority:-** means the requirement must be incorporated in the next product release.
- **Medium priority:-** means the requirement is necessary but it can be deferred to a later release if necessary.
- **Low priority:-** means it would be nice to have, but we realize it might have to be dropped if we have insufficient time or resources.

Cont'd…

V. Unambiguous

- The reader of a requirement statement should be able to draw **only one interpretation** of it.
- Also, multiple readers of a requirement should arrive at the same interpretation.
- Natural language is highly prone to ambiguity.
  - Avoid subjective words like user friendly, easy, simple, rapid, efficient, several, state-of-the-art, improved, maximize, and minimize.

➢ Words that are clear to the SRS author may not be clear to readers.

Cont'd…

- Write each requirement in concise, simple, straightforward language of the user domain, not in computers.

➢ **Effective ways to reveal ambiguity include:**

- Formal inspections of the requirements specifications,
- Writing test cases from requirements and
- Creating user scenarios that illustrate the expected behavior of a specific portion of the product.

Cont'd…

VI. Verifiable

- See whether you can devise tests or use other verification approaches, such as inspection or demonstration, to determine whether each requirement is properly implemented in the product.
- If a requirement is not verifiable, determining whether it was correctly implemented is a matter of opinion.

❖Requirements that are not consistent, feasible, or unambiguous also are not verifiable.

- Any requirement that says the product shall "support" something is not verifiable.

Characteristics of Quality Requirements Specifications

Characteristics of a high quality SRS:

A. Complete

- No requirements or necessary information should be missing.
- Completeness is also a desired characteristic of an individual requirement.

B. **Consistent**

- Consistent requirements do not conflict with other software requirements.

➢ Disagreements among requirements must be resolved before development can proceed.

❑ Be careful when modifying the requirements, as inconsistencies can slip in undetected if you review only the specific change and not any related requirements.

Cont'd…

C. Modifiable

- You must be able to revise the SRS when necessary and maintain a history of changes made to each requirement.
- You can make an SRS more modifiable by organizing it so that related requirements are grouped together, and by creating a table of contents, index, and cross-reference listing.

D. **Traceable**

- You should be able to link each software requirement to its source, which could be a higher-level system requirement, a use case, or a voice-of-the-customer statement.

Modelling

- **System modeling** is the process of developing abstract models of a system,
  - with each model presenting a different view or perspective of that system.
- It is about representing a system using some kind of graphical notation, which is now almost always based on notations in the **Unified Modeling Language** (**UML**).
- Models help the analyst to understand the functionality of the system;
  - they are used to communicate with customers.

Cont'd…

- Models can explain the system from **different perspectives**:
  - An **external** perspective, where you model the context or environment of the system.
  - An **interaction** perspective, where you model the interactions between a system and its environment, or between the components of a system.
  - A **structural** perspective, where you model the organization of a system or the structure of the data that is processed by the system.
  - A **behavioral** perspective, where you model the dynamic behavior of the system and how it responds to events.

Cont'd…

- System models describe a particular aspect of a system, such as:
  - the way the system is decomposed into subsystems,
  - the way that data is processed, the structure of the data, etc.
- They are used in a requirements document:
  - ✓ to add information to natural language descriptions of the system requirements.
- ➢ They may be developed during the requirements elicitation and analysis to help understand the requirements.
- Sometimes, they can be considered as a detailed system specification of what is required.

Cont'd…

- System models supplement natural language descriptions of requirements.

❑The requirements description and the associated system models are usually developed at the same time.

- The development of one helps with the development of the other as exploring aspects of the requirements through modelling reveals inconsistencies and incompleteness.

❑In some organizations, it is normal practice to develop very detailed and complete system models as part of the system specification.

Cont'd…

- The models which you need for your system depend on the **type of system** and the **people** who will use the models.

❑ However, it is recommend that two high-level models should always be included in a requirements specification.

  - These are as follows.

1. A model which shows the **environment** in which the system being specified will operate.
2. An **architectural model** which shows how the system is decomposed into sub-systems.

Assignment-#1 (10%)

- Read about the following UML Diagrams that are the most useful for system modeling and prepare a short note on these models with examples.

  1. Activity diagram
  2. Use case diagram
  3. Sequence diagram
  4. Class diagram
  5. State diagram

Writing requirement documents

- **General requirements of all software documentation:**
  - ✓Should provide for communication among team members
  - ✓Should act as an information repository to be used by maintenance engineers
  - ✓Should provide enough information to management to allow them to perform all program management related activities
  - ✓Should describe to users how to operate and administer the system
  - ✓In all software projects, some amount of documentation should be created prior to any code being written.
    - Design docs, etc.
  - ✓Documentation should continue after the code has been completed
    - User's manuals, etc.

Cont'd…

- The two main types of documentation created are ***Process*** and ***Product*** **documents**

❖**Process Documentation**

- Used to record and track the development process.
  - Planning documentation, Cost, Schedule, Fund tracking, Standards etc.
- This documentation is created to allow for successful management of a software product and has a relatively short lifespan.
- Only important to internal development process except in cases where the customer requires a view into this data.

❑Some items, such as papers that describe design decisions should be extracted and moved into the ***product*** **documentation** category when they become implemented

Cont'd…

Software Documentation Standards

- Documentation standards in a software project are important
  - because documents are the only tangible way of representing the software and the software process.
- Standardized documents have a consistent appearance, structure and quality,
  - and should therefore be easier to read and understand.

Cont'd…

2. Product Documentation

- Describes the **delivered** product
- Must evolve with the development of the software product

- **Two main categories:**
  - System Documentation
  - User Documentation

Cont'd…

❖System Documentation

- Describes how the system works, but not how to operate it.

❑Examples:

- Requirements Spec
- Architectural Design: framework for sub-system control and communication.
- Detailed Design :implementation details, modules and their implementation.
- Commented Source Code: including output such as JavaDoc
- Test Plan: including test cases
- V&V plan and results
- List of Known Bugs

Cont'd…

- **User Documentation:** has two main types
  1. End User
  2. System Administrator

- There are five important areas that should be documented for a formal release of a software application
  a) Functional Description of the Software
  b) Installation Instructions
  c) Introductory Manual
  d) Reference Manual
  e) System Administrator's Guide

Cont'd…

❖Document Structure

- All documents for a given product should have a similar structure
  - A good reason for *product standards*
- The IEEE Standard for User Documentation lists such a structure
  - The authors "best practices" are:

1. Put a cover page on all documents
2. Divide documents into chapters with sections and subsections
3. Add an index if there is lots of reference information
4. Add a glossary to define ambiguous terms

Cont'd…

❖**Standards**

- Standards play an important role in the development, maintenance and usefulness of documentation.
- Standards can act as a basis for quality documentation but are not good enough on their own.
- Usually define high level content and organization

1. Process standards

- Define the approach that is to be used when creating the documentation
- Don't actually define any of the content of the documents

Cont'd…

2. Product standards

- Goal is to have all documents created for a specific product attain a consistent structure and appearance
  - Can be based on organizational or contractually required standards
- **Four main types:**
  1. Documentation Identification Standards
  2. Document Structure Standards
  3. Document Presentation Standards
  4. Document Update Standards

Cont'd…

3. Interchange standards

- Deals with the creation of documents in a format that allows others to effectively use.
- PDF may be good for end users who don't need to edit
- Word may be good for text editing

➢Specialized CASE tools need to be considered.

Cont'd…

❖Other standards

➢IEEE

- Has a published standard for user documentation
- Provides a structure and superset of content areas

- Many organizations probably won't create documents that completely match the standard.

➢Writing Style

- Ten "best practices" when writing are provided
- Author proposes that
  - group edits of important documents should occur in a similar fashion to software walkthroughs.

Ten best practices

10 Best Practices for Writing and Editing Technical Documents

1. Know Your Audience and Write Exclusively for Them and to Them.
2. Organize and Outline Your Technical Writing Before You Write.
3. Consider Each Document's Layout Before You Write.
4. Always Insert Images, Videos, Data Visualizations, and Other Visual Aids.
5. Be Concise and Use Plain Language.
6. Remain Consistent.
7. Remember That You’re a Human Writing for Other Humans
8. Always Have Others Edit Your Work and Provide Feedback
9. Stay Focused on the Purpose of Each Technical Document
10. Test the Practical Usefulness of Each Technical Document

Reviewing Requirements for Quality

- What do good requirements really look like?
- Following are several requirements adapted from actual projects.

❑**Example #1:** *"The product shall provide status messages at regular intervals **not** less than **every** 60 seconds."*

- This requirement is incomplete:
  - what are the status messages and how are they supposed to be displayed to the user?
  - The requirement contains several ambiguities.
    - What part of "the product" are we talking about?
  - Is the interval between status messages really supposed to be at least 60 seconds, so showing a new message every 10 years is okay? Perhaps the intent is to have no more than 60 seconds elapse between messages; would 1 millisecond be too short? The word "**every**" just confuses the issue.

➢As a result of these problems, the requirement is not verifiable.

Cont'd…

- Here is one way we could rewrite the requirement to address those shortcomings:
- "**1. Status Messages.**

1.1. The Background Task Manager shall display status messages in a designated area of the user interface at intervals of 60 plus or minus 10 seconds.

1.2. If background task processing is progressing normally, the percentage of the background task processing that has been completed shall be displayed.

1.3. A message shall be displayed when the background task is completed.

1.4. An error message shall be displayed if the background task has halted."

Guidelines for Writing Quality Requirements

- **There is no formulaic way to write excellent requirements.**
- It is largely a matter of experience and learning from the requirements problems you have encountered in the past.

❖Here are a few guidelines to keep in mind as you document software requirements.

- Keep sentences and paragraphs short.
- Use the active voice.
- Use proper grammar, spelling, and punctuation.
- Use terms consistently and define them in a glossary or data dictionary.
- Avoid long narrative paragraphs that contain multiple requirements.
- Never use "**and/or**" in a requirement statement.
- Write requirements at a consistent level of detail throughout the document.
- Avoid stating requirements redundantly in the SRS.

Thank You!

?