UNIT 3 :: REQUIREMENT ANALYSIS AND SPECIFICATION

The requirement analysis and specification phase starts after the feasibility study phase is complete and the project has been found to be financially and technically feasible.

The goal of the requirement analysis and specification phase is to clearly understand the customer requirements and to systematically organize the requirements into a specification document. The SRS document is the final outcome of the requirement analysis and specification phase.

The two main activities carried out during requirement analysis and specification phase are of two types

- Requirement Gathering and Analysis
- Requirement Specification

3.1 REQUIREMENT GATHERING AND ANALYSIS:

We can broadly divide the requirements gathering and analysis activity into two separate tasks as requirements gathering and requirements analysis.

Requirement Gathering:

The analyst starts requirements gathering activity by collecting all information that could be useful to develop the system. The important ways in which the analyst gather requirements

1. Study the existing Documentation: The analyst usually studies all existing documents regarding the system to be developed before visiting the customer site.

2. Interview: There are many different categories of users of software. Different categories of users typically require different features from the software. All the different categories of users are interviewed to gather the different functionalities required by them.

For Example, to perform requirement gathering of Library automation software, the analyst must interview the Library members, the librarian, the accountant.

3. Task Analysis: The users usually view software as a box that provides a set of services. A service is also called a Task. For each Task the analyst tries to formulate different steps necessary to realize the service in consultation with the users.

For Issue Book Service, the steps may be

Authenticate Users ,Check the number of Books issued to customer, Check whether the book has been reserved, Issue Book , Print Book issue Slip

4. Scenario Analysis: A task may have many scenarios (situations) of operation. The different scenarios of a task can occur when the task is invoked under different situations.

The different scenario for the Book issue task of a Library automation software may be

Book issue is performed and Issue slip is printed.

The book is reserved and cannot be issued

The maximum number of books that can be issued to the member is exceeded and cannot be issued.

For various tasks, the different types of scenarios may occur are identified and details of each scenario is identified in consultation with the users.

5. Form Analysis: The different forms are analyzed to determine the data input to the system and the data that are output from the system. For different inputs how these are used by the system to produce the corresponding output data are determined from the users.

Requirement Analysis The main purpose of the requirements analysis activity is to analyze the collected information to obtain a clear understanding of the product to be developed with a view to removing all ambiguities, incompleteness, and inconsistencies.

There are 3 main types of problems in the requirements that the analyst needs to identify are resolve

Ambiguity, Inconsistency, Incompleteness

Ambiguity: When a requirement is ambiguous several interpretations of that requirement are possible. Ambiguity in the requirements can lead to the development of incorrect system. Example: When the temperature becomes high, the heater should be switched off. The word High may be interpreted differently by different people.

Inconsistency: Two requirements are said to be inconsistent, if one of the requirements contradicts the other. or two-end users of the system gives inconsistent description of the requirement.
Example: One end-user says that the furnace be switched off when the temperature of the furnace rises above some specified value; whereas another user expressed that the water shower should be switched on instead of furnace being switched off.

Incompleteness: An incomplete set of requirements is one in which some requirements have been overlooked. Incompleteness is caused by the inability of the customer to visualize the system that is to be developed.

Example: One of the requirements is that if the Temperature exceeds $200^{o}C$ then an alarm bell must be sounded. However, there is no provision for resetting the alarm bell in any of the requirements.

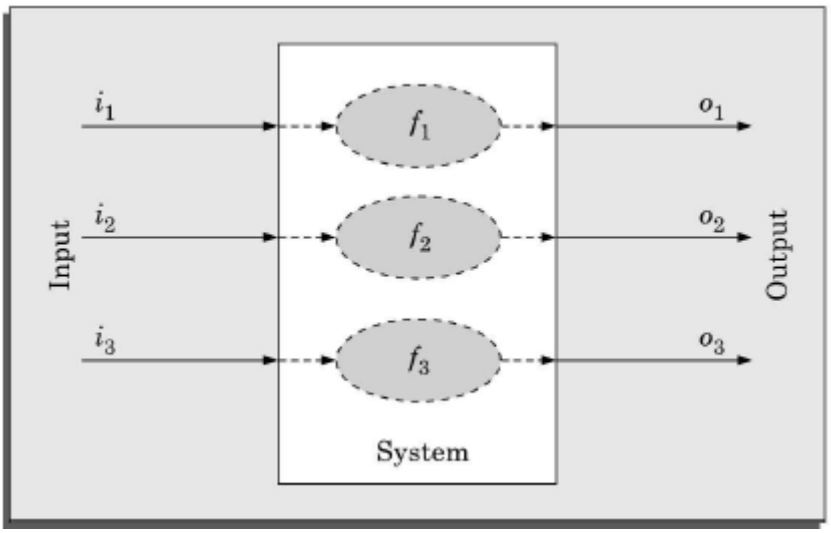
3.2 SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

After all the requirements has been gathered and analyzed the analyst starts to organize the requirements in the form of an SRS document. Among all the documents produced during a software development life cycle, writing the SRS document is probably the toughest. one reason behind this difficulty is that the SRS document is expected to cater to the needs of a wide variety of audience. Some of the users of SRS document are software developers, Test engineers, Project managers and maintenance engineers.

3.2.1 Contents of the SRS Document

The SRS document should clearly document the following three aspects of a system.
Functional Requirements
Non-Functional Requirements
Goals of Implementation

Functional Requirements: Functional requirements discuss the functionalities required by the users from the system. We can consider system as performing a set of functions $\{f_i\}$. These functions can be considered similar to a mathematical function f: I - > O. meaning that a function transforms an element (i_i) in the input domain (I) to a value (o_i) in the output(O)



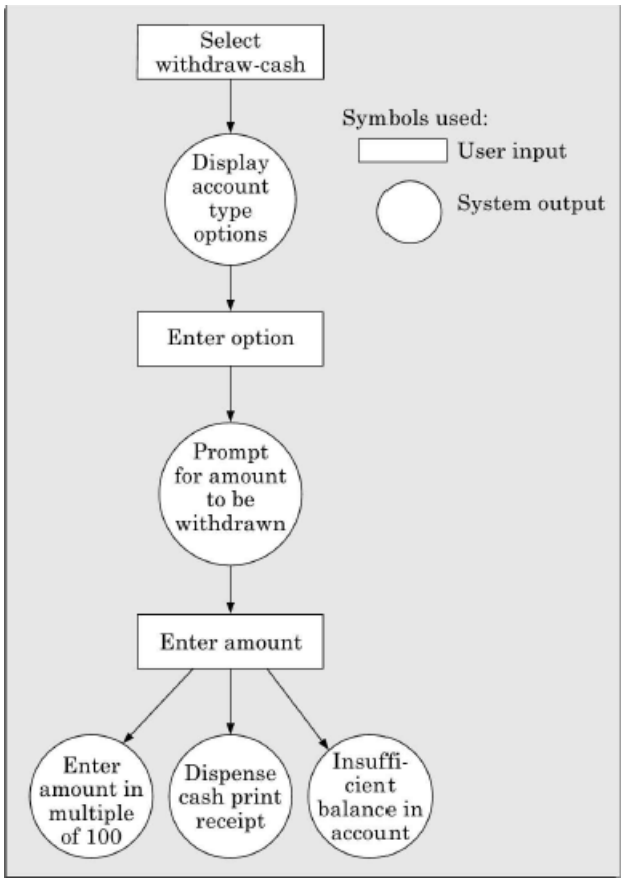
Non – Functional Requirements: The non-Functional requirements deal with the characteristics of a system that cannot be expressed as functions. Non – Functional requirements address aspects concerning maintainability, portability, usability, maximum number of concurrent users, timing and throughput.

Goals of Implementation: The goals of implementation part of the SRS document offer some general suggestions regarding development. The developers may take these suggestions into account if possible. A goal is not checked by customer at the time of acceptance testing.

3.2.2 Functional Requirements:

In order to document functional requirements of a system, it is necessary to first learn to identify the high-level functions of a system. A high level function is one using which the user can get some useful piece of work done.

Each high-level requirement typically involves accepting some data from the user, transforming it to the required response, and then outputting the system response to the user. In fact a high level function usually involves a series of interactions between the system and one or more users. Even for the same high-level function, there can be different interaction sequences or scenarios due to users selecting different options or entering different data items. Each high level requirements might consists of several sub requirements.



Interactions between the user and the system in the withdraw_cash high-level functional requirement

3.2.3 How to identify the Functional Requirements

The high level functional requirements often need to be identified either from an informal problem description document or from a conceptual understanding of the problem.
"Each high-Level requirement characterizes a way of system usage (service invocation) by some user to perform some meaningful piece of work."
Remember that there can be many types of users of a system and their requirements from the system may be very different. So, it is often useful to first identify the different types of users who might use the system and then to try to identify the different services expected from the software by different types of users.

3.2.4 How to Document the Functional Requirements

Once all the high level functional requirements have been identified, they can be documented. A function can be documented by identifying the state at which the data is to be input to the system, its input data domain, the output data domain, and the type of processing to be carried on the input data to obtain the output data.

Example Withdraw Cash from ATM

R.1: Withdraw cash

*Description:*The withdraw cash function first determines the type of account that the user has and the account number from which the user wishes to withdraw cash. It checks the balance to determine whether the requested amount is available in the account. If enough balance is available, it outputs the required cash, otherwise it generates an error message.

R.1.1: Select withdraw amount option

Input: "Withdraw amount" option

Output: User prompted to enter the account type

R.1.2: Select account type

Input: User option from any one of the following: savings/checking/deposit.

Output: Prompt to enter amount

R.1.3: Get required amount

Input: Amount to be withdrawn in integer values greater than 100 and less than 10,000 in multiples of 100.

Output: The requested cash and printed transaction statement.

Processing: The amount is debited from the user's account if sufficient balance is available, otherwise an error message is displayed.

Example 2:

. Search Book Availability in Library

R.1. Search book

Description: Once the user selects the search option, he would be asked to enter the keywords. The system would search the book in the book list based on the keywords entered. After making the search, the system should output the details of all books whose title or author name match any of the keywords entered. The book details to be displayed include: title, author name, publisher name, year of publication, ISBN number, catalog number, and the location in the library.

R.1.1: Select search option

Input: "Search" option

Output: User prompted to enter the keywords

R.1.2: Search and display

Input: Keywords

Output: Details of all books whose title or author name matches any of the keywords entered by the user. The book details displayed would include: title of the book, author name, ISBN number, catalog number, year of publication, number of copies available, and the location in the library.

Processing: Search the book list based on the keywords

R.2: Renew book

Description: When the "renew" option is selected, the user is asked to enter his membership number and password. After password validation, the list of the books borrowed by him are displayed. The user can renew any of his borrowed books by indicating them. A requested book cannot be renewed if it is reserved by another user. In this case, an error message would be displayed.

R.2.1: Select renew option

State: The user has logged in and the main menu has been displayed

Input: "Renew" option selection

Output: Prompt message to the user to enter his membership number and password

R.2.2: Login

State: The renew option has been selected

Input: Membership number and password

Output: List of the books borrowed by the user is displayed, and user is prompted to select the books to be renewed, if the password is valid. If the password is invalid, the user is asked to reenter the password.

Processing: Password validation, search the books issued to the user from the borrower's list and display

Next function: R.2.3 if password is valid and R.2.2 if password is invalid

R.2.3: Renew selected books

Input: User choice for books to be renewed out of the books borrowed by him.

Output: Confirmation of the books successfully renewed and apology message for the books that could not be renewed.

Processing: Check if any one has reserved any of the requested books. Renew the books selected by the user in the borrower's list, if no one has reserved those books.

Traceability: Traceability means that it would be possible to tell which design component corresponds to which requirement, which code part corresponds to which design component, and which test case corresponds to which requirement etc., Thus a given code component can be traced to the corresponding design component, and a design component can be traced to a specific requirement that it implements vice versa.

That is traceability makes it easy to identify which parts of the design and code would be affected, when certain requirements change occurs. It can also be used to study the impact of a bug on various requirements etc.,

To achieve traceability it is necessary that each functional requirement should be numbered uniquely and consistently. Proper numbering of the requirements makes it possible for different documents to uniquely refer to specific requirements.

3.2.5 Characteristics of a Good SRS Document

The skill of writing a good SRS document usually comes from the experience gained from writing SRS document for many problems. Some of the identified desirable qualities of the SRS document are the following.

Concise: The SRS document should be concise and at the same time unambiguous, consistent and complete. Wordy and irrelevant descriptions reduce readability and also increase error possibilities.
Structured: It should be well structured. A well structured document is easy to understand and modify.
Black Box View: It should only specify what the system should do and refrain (avoid doing) how to do these. This means that the SRS document should specify the external behavior of the system and not discuss the implementation issues.
Traceable: It should be possible to trace a specific requirement to the design elements that implement it and vice versa. Similarly it should be possible to trace a requirement to the code segments, test cases that test requirements and vice versa.
Response to undesirable events: It should characterize acceptable responses to undesired events. These are called system response to exceptional conditions.
Verifiable: All requirements of the system as documented in the SRS document should be verifiable. This means that it should be possible to determine whether or not requirements have been met in an implementation. Requirements such as the system should be user-friendly are not verifiable. Any requirements that is not verifiable should be listed separately in the goals of implementation.

3.2.6 Examples of Bad SRS Document

The important problems that one needs to watch out in SRS Document are incompleteness, ambiguity and contradictions. There are many other problems that SRS document might suffer from. The important categories of problems that many SRS documents suffer from are
Over Specification: Over specification occurs when you try to address the how to aspects in the SRS documents. Over specification restricts the freedom of the designers in arriving at the good design solution.
Forward References: You should not refer to aspects that are discussed much latter in the SRS document. Forward referencing seriously reduce readability of the specification.
Wishful Thinking: This type of problems concern description of aspects which would be difficult to implement.
Noise: The term noise refers to presence of material not directly relevant to the software development.

3.2.7 Organization of the SRS Document

1. **Introduction**
   - (a) Purpose
   - (b) Overview
   - (c) Environmental Characteristics
     - i. Hardware
     - ii. Peripherals
     - iii. People
2. **Goals of Implementation**
3. **Functional Requirements**
   - (a) User class 1
     - i. Functional requirement 1.1
     - ii. Functional requirement 1.2
   - (b) User class 2
     - i. Functional requirement 2.1
     - ii. Functional requirement 2.2
4. **Non-functional Requirements**
   - (a) External interfaces
   - (b) User interfaces
   - (c) Software interfaces
   - (d) Communication interfaces
5. **Behavioural Description**
   - (a) System states
   - (b) Events and actions

Organization of the SRS document depends to a large extent on the system analyst himself, often guided by the policies and standards of the company. Also the organization of the document to a large extent depends on the type of the product being developed. However irrespective of the company principles and product type the 3 basic issues that any SRS document should discuss are: Functional requirements, non-Functional requirements and guidelines for system Implementation.
The following is the organization of SRS document as prescribed by the IEEE 830 standard. IEEE 830 serves only as guidelines for SRS. Depending on the specific problem being specified, some sections can be omitted, introduced or interchanged.

The introduction section describes the context in which the system is being developed.

The environment characteristics subsection describes the properties of the environment with which the system will interact.

Specification of the behavior may not be necessary for all systems. It is necessary for system transits among a set of states depending on some condition.