Software Testing

Unit: 1

Lecture-6

By: Monika Singh
Guest Faculty (B.Voc Robotics & AI)
DEI

Software Requirement Specification (SRS) Format

In order to form a good SRS, there are some points which can be used and should be considered to form a structure of good SRS. These are as follows :

1. Introduction
(i) Purpose of this document
(ii) Scope of this document
(iii) Overview
2. General description
3. Functional Requirements
4. Interface Requirements
5. Performance Requirements
6. Design Constraints
7. Non-Functional Attributes
8. Preliminary Schedule and Budget
9. Appendices

Software Requirement Specification (SRS) Format as name suggests, is complete specification and description of requirements of software that needs to be fulfilled for successful development of software system. These requirements can be functional as well as non-functional depending upon type of requirement. The interaction between different customers and contractor is done because its necessary to fully understand needs of customers.

Depending upon information gathered after interaction, SRS is developed which describes requirements of software that may include changes and modifications that is needed to be done to increase quality of product and to satisfy customer's demand.

1. Introduction :

(i) Purpose of this Document –

At first, main aim of why this document is necessary and what's purpose of document is explained and described.

(ii) Scope of this document –

In this, overall working and main objective of document and what value it will provide to customer is described and explained. It also includes a description of development cost and time required.

(iii) Overview –

In this, description of product is explained. It's simply summary or overall review of product.

General description :

- In this, general functions of product which includes objective of user, a user characteristic, features, benefits, about why its importance is mentioned. It also describes features of user community.

Functional Requirements :

- In this, possible outcome of software system which includes effects due to operation of program is fully explained. All functional requirements which may include calculations, data processing, etc. are placed in a ranked order.

Interface Requirements :

- In this, software interfaces which mean how software program communicates with each other or users either in form of any language, code, or message are fully described and explained. Examples can be shared memory, data streams, etc.

Performance Requirements :

- In this, how a software system performs desired functions under specific condition is explained. It also explains required time, required memory, maximum error rate, etc.

Design Constraints :

- In this, constraints which simply means limitation or restriction are specified and explained for design team. Examples may include use of a particular algorithm, hardware and software limitations, etc.

Non-Functional Attributes :

- In this, non-functional attributes are explained that are required by software system for better performance. An example may include Security, Portability, Reliability, Reusability, Application compatibility, Data integrity, Scalability capacity, etc.

Preliminary Schedule and Budget :

- In this, initial version and budget of project plan are explained which include overall time duration required and overall cost required for development of project.

Appendices :

- In this, additional information like references from where information is gathered, definitions of some specific terms, acronyms, abbreviations, etc. are given and explained.

Quality Characteristics of a good SRS:

Following are the characteristics of a good SRS document:

- **Correctness:**

User review is used to ensure the correctness of requirements stated in the SRS. SRS is said to be correct if it covers all the requirements that are actually expected from the system.

- **Completeness:**

Completeness of SRS indicates every sense of completion including the numbering of all the pages, resolving the to be determined parts to as much extent as possible as well as covering all the functional and non-functional requirements properly.

- **Consistency:**

Requirements in SRS are said to be consistent if there are no conflicts between any set of requirements. Examples of conflict include differences in terminologies used at separate places, logical conflicts like time period of report generation, etc.

- **Unambiguousness:**

A SRS is said to be unambiguous if all the requirements stated have only 1 interpretation. Some of the ways to prevent unambiguousness include the use of modelling techniques like ER diagrams, proper reviews and buddy checks, etc.

- **Ranking for importance and stability:**

There should a criterion to classify the requirements as less or more important or more specifically as desirable or essential. An identifier mark can be used with every requirement to indicate its rank or stability.

- **Modifiability:**

SRS should be made as modifiable as possible and should be capable of easily accepting changes to the system to some extent. Modifications should be properly indexed and cross-referenced.

- **Verifiability:**
  A SRS is verifiable if there exists a specific technique to quantifiably measure the extent to which every requirement is met by the system. For example, a requirement starting that the system must be user-friendly is not verifiable and listing such requirements should be avoided.

- **Traceability:**
  One should be able to trace a requirement to design component and then to code segment in the program. Similarly, one should be able to trace a requirement to the corresponding test cases.

- **Design Independence:**
  There should be an option to choose from multiple design alternatives for the final system. More specifically, the SRS should not include any implementation details.

- **Testability:**
  A SRS should be written in such a way that it is easy to generate test cases and test plans from the document.

- **Understandable by the customer:**
  An end user maybe an expert in his/her specific domain but might not be an expert in computer science. Hence, the use of formal notations and symbols should be avoided to as much extent as possible. The language should be kept easy and clear.

- **Right level of abstraction:**
  If the SRS is written for the requirements phase, the details should be explained explicitly. Whereas, for a feasibility study, fewer details can be used. Hence, the level of abstraction varies according to the purpose of the SRS.

Software Quality Assurance

- Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.

- Software Quality Assurance is a process which works parallel to development of software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

- Software Quality Assurance has:

- A quality management approach
- Formal technical reviews
- Multi testing strategy
- Effective software engineering technology
- Measurement and reporting mechanism

Major Software Quality Assurance Activities:

SQA Management Plan:

Make a plan for how you will carry out the sqa through out the project. Think about which set of software engineering activities are the best for project. check level of sqa team skills.

Set The Check Points:

SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.

Multi testing Strategy:

Do not depend on a single testing approach. When you have a lot of testing approaches available use them.

Measure Change Impact:

The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to change check the compatibility of this fix with whole project.

Manage Good Relations:

In the working environment managing good relations with other teams involved in the project development is mandatory. Bad relation of sqa team with programmers team will impact directly and badly on project. Don’t play politics.

Benefits of Software Quality Assurance (SQA):

SQA produces high quality software.
High quality application saves time and cost.
SQA is beneficial for better reliability.
SQA is beneficial in the condition of no maintenance for a long time.
High quality commercial software increase market share of company.
Improving the process of creating software.
Improves the quality of the software.

Disadvantage of SQA:

There are a number of disadvantages of quality assurance. Some of them include adding more resources, employing more workers to help maintain quality and so much more.

THANK YOU