

Impact of Software Requirement Specification in Reducing Software Risk

Dr. Mohit Kumar Sharma

Associate Professor & Head, Post Graduate Department of Computer Science, J.C. D.A.V. College, Dasuya, Punjab, India

Abstract: *Software Requirements Specification (SRS) is a vital document in the Software Development Life Cycle for creation of quality software. It is mainly prepared during the requirements analysis phase and describes what the software system should do and what requirements it must satisfy. It describes the functional and non-functional requirements of a software. It describes what the system should do, its features, constraints, and the expectations of users. The purpose of an SRS is to provide a clear and detailed understanding of the software requirements to developers, testers and clients. Software Requirements Specification (SRS) plays an important role in reducing risks during software development. It clearly defines what the software should do and provides a common understanding between different users. This paper presents an impact of software requirement specification in reducing different software risk for developing well-engineered software.*

Keywords: SRS, Structure, Risk Management, Quality

1. Introduction

Software Requirements Specification (SRS) is a vital document in the Software Development Life Cycle (SDLC). It is mainly prepared during the requirements analysis phase and describes what the software system should do and what requirements it must satisfy. Software Requirements Specification (SRS) is a document that describes the functional and non-functional requirements of a software system. It explains what the system should do, its features, constraints, and the expectations of users. The main aim of an SRS is to provide a clear and detailed understanding of the software requirements to developers, testers and clients [1].

The output of the requirements phase of the software development process is Software Requirements Specification (SRS) as this document lays a basis for software engineering activities and is created when entire requirements are produced and scrutinized. SRS is a formal document, which acts as a representation of software that enables the users to review whether it is according to their requirements. It includes all user requirements for a system as well as detailed specifications of the system requirements [2]. This document completely describes what the proposed software should do without describing how the software will do it.

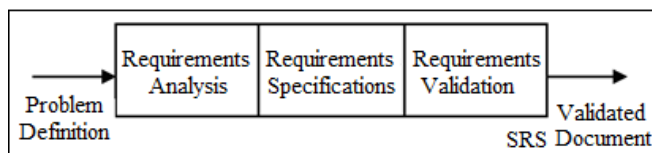


Figure 1: Phases in SRS Document

Different objectives supported by SRS are as:

- 1) **Reduce Software Risks:** It significantly reduces software risk by clarifying requirements, preventing misunderstandings, controlling scope, reducing rework, and improving planning and testing.
- 2) **Design Input:** Covers sufficient detail in the functional system requirements to plan a design solution.

- 3) **Basis for understanding:** Offers a complete description of the functions to be performed by the system. It helps the users to determine whether the specified requirements are accomplished.
- 4) **Assessing costs:** Controls the requirements of the system and thus enables the developer to have estimate of the total cost and scheduling of the software.
- 5) **Validation:** Apply validation strategies applied to the requirements to acknowledge that requirements are specified properly.
- 6) **Decomposition:** Arranges the information and divides the problem into modules in a proper manner.
- 7) **Response:** It ensures to the user that the organization understands problems to be solved by feedback and the software performance necessary to solve problems.

Software Requirements Specification (SRS) is used by various users in the organization for software design. In addition, it enables the managers to plan for the system development processes. Software engineers need a requirements document to understand what system is to be developed. These engineers also require this document to develop validation tests for the required system. In the end, requirements document is needed by system maintenance engineers to use the requirement and the relationship between its modules. Software requirements specification should be accurate, complete, efficient, and consistent, so that it does not affect the entire project design. SRS is correct when all user requirements are stated in the requirements document and stated requirements should be according to the desired system [3]. Correctness ensures that all specified requirements are performed correctly. It should be unambiguous when every stated requirement has only one interpretation. This suggests that each requirement is uniquely interpreted. It should be complete when the requirements clearly define what the software is required to do. The requirements of the user can change; hence requirements document should be created in such a manner that those changes can be modified easily, consistently maintaining the structure and style of the SRS. It should be verifiable when the specified requirements can be verified with a cost-effective process to check whether the final software meets those requirements [4]. It should be

consistent when the subsections of requirements defined do not conflict with each other.

2. Review of Structure of SRS

SRS document is devised in a manner that is easier to write, review, and maintain. It is organized into autonomous sections and each section is organized into modules or units. The level of detail to be included in the SRS depends on the type of the system to be developed and the process model chosen for its development. Each SRS fits a certain pattern; thus, it is essential to standardize the structure of the requirements document to make it easier to understand [5]. This document comprises the following sections as:

- 1) **Introduction:** This provides an overview of the entire information described in SRS. This involves purpose and the scope of SRS, which states the functions to be performed by the system.
- 2) **Project perspective:** It determines whether the project is an independent product or an integral part of the larger product. It determines the interface with hardware, software, system, and communication.
- 3) **Goals of Implementation:** It provides a summary of the objectives and functions to be performed by the software. The functions are organized in a list so that they are easily understandable by the user:
- 4) **Restrictions:** It provides the general description of the constraints such as regulatory policies, audit functions, reliability requirements, and so on.
- 5) **Precise necessities:** These determine all requirements in detail so that the designers can design the system in accordance with them. The requirements include description of every input and output of the system and functions performed in response to the input provided.
- 6) **Functional requirements:** It determines the functional capabilities of the system. For each functional requirement, the accepting and processing of inputs in order to generate outputs are specified.
- 7) **Database requirements:** It determines logical requirements to be stored in the database. This includes type of information used, frequency of usage, data entities and relationships among them, and so on.
- 8) **Non-Functional requirements:** It determines the non-functional of the system that cannot be expressed as functions. It determines the interface of the software with other systems, which can include interface with operating system and so on. External interface also specifies the interaction of the software with users, hardware, or other software.
- 9) **Design and implementation constraint:** It determines all design constraints that are imposed by standards, hardware limitations, and so on. Standard compliance determines requirements for the system, which are in compliance with the specified standards.
- 10) **User Documentation:** It provides information such as user manuals, on-line help, and so on. This is necessary especially when SRS is prepared for large and complex projects.

3. Software Risk Management

Risk management involves all actions pertaining to identification, analyzing and making provision for

predictable and non-predictable risks in the project. Risk is an unwanted situation, expectation of loss, a potential problem that may or may not occur in the future. It is generally caused due to lack of information, control or time. A possibility of suffering from loss in software development process is called a software risk. Loss can be anything, increase in production cost, development of poor-quality software, not being able to complete the project on time. Software risk exists because the future is uncertain and there are many known and unknown things that cannot be incorporated in the project plan [6]. A software risk can be of two types as internal risks that are within the control of the project manager and external risks that are beyond the control of project manager. Risk Management comprises of following processes:

- 1) **Risk Identification** - In order to identify the risks that project may be subjected to; it is important to first study the problems faced by previous projects. The best ways of analyzing a project plan are by converting it to a flowchart and examine all essential parts. In this phase of risk management, you have to define processes that are important for risk identification. All the details of the risk such as exclusive ID, date on which it was recognized, description and so on should be clearly stated.
- 2) **Risk Analysis** - In this phase the risk is identified and then considered. After the categorization of risk, the level, impact of the risk is analyzed. Possibility is defined in percentage after examining what are the chances of risk to occur due to various technical conditions.
- 3) **Risk Planning** - Risk planning is all about defining preventive measure that would lower down the likelihood or probability of various risks. Define measures that would reduce the impact in case a risk happens. Persistent monitoring of processes to identify risks as early as possible.
- 4) **Risk Monitoring** - Risk monitoring is integrated into project activities and regular checks are conducted on top risks. Risk monitoring comprises of tracking of risk plans for any major changes in actual plan, attribute etc. Preparation of status reports for project management. Review risks and risks whose impact has reached and regularly search for new risks.

4. Impact of SRS in Reducing Software Risk

Software Requirements Specification (SRS) plays an important role in reducing risks during software development. It clearly defines what the software should do and provides a common understanding between the client, users, developers, and testers. Major impacts of SRS on reducing software risk as: -

- 1) **Better understanding:** A well-written SRS clearly describes user and system requirements, reducing confusion between clients and developers.
- 2) **Software Quality:** Functional and non-functional requirements provide clear criteria for designing and testing the software.
- 3) **Prevents unnecessary updates:** Clearly defined requirements help prevent unnecessary features and changes from being added during development.
- 4) **Development costs:** Identifying requirements early reduces expensive changes and revise at later stages.

- 5) **Project planning and scheduling:** The SRS helps developers estimate time, resources, cost, and effort more accurately.
- 6) **Better testing:** Testers can use the SRS as a basis for creating test cases and verifying whether the system meets its requirements.
- 7) **Better communication:** The SRS acts as a common reference document for all users.
- 8) **Good Maintenance:** When requirements change, the documented SRS helps teams evaluate the impact of those changes.
- 9) **Decreases technical risks:** Clearly documented requirements help identify technical constraints, dependencies, and potential problems early.
- 10) **Project documentation:** The SRS serves as a reference throughout development and maintenance, reducing dependency on individual team members.

5. Conclusion

A well-written SRS clearly describes user and system requirements, reducing confusion between clients and developers. SRS significantly reduces software risk by clarifying requirements, preventing misunderstandings, controlling scope, reducing rework, and improving planning and testing. A complete and accurate SRS therefore increases the chances of delivering the software on time, within budget, and according to user expectations. Software Requirements Specification (SRS) plays an important role in reducing risks during software development. It clearly defines what the software should do and provides a common understanding between users.

References

- [1] Roger S. Pressman and B.R. Maxim, Software Engineering, Tata-McGraw Hill Publishing House, 9th edition, 2023
- [2] Ali Behforooz and Frederick J.H., Software Engineering Fundamentals, Oxford University Press. 1996
- [3] Nasib S. Gill, Software Engineering-Software Reliability and Testing, K.B. Publishing, 2020
- [4] Sommerville, I: Software Engineering, Pearson Education.
- [5] Rajib Mall: Fundamentals of Software Engineering, PHI Learning Pvt. Limited
- [6] Jalota, Pankaj: An integrated approach to Software Engineering, Narosa Publishing Company.

Author Profile



Dr. Mohit Kumar Sharma (Ph.D., M.Phil., M.Sc.) is working as Associate Professor & Head, Post Graduate Department of Computer Science, J.C. D.A.V. College, Dasuya, Punjab, India. He has 18 years teaching and research experience. He has 25 research publications with book chapters in international refereed journals/publishers, 20 paper presentations at international/national level conferences/ seminars, 2 books authored and 3 edited books. He is also member of undergraduate and postgraduate board of studies, Computer Science and Applications in Panjab University, Chandigarh. His research interests are in Software Engineering and Object-Oriented Programming.