Proceedings of the Informatics Conference Vol. 11 No. 22 (2025): Requirements Engineering Methodologies ISSN: 2477-5894

The Importance of Comprehensive Software Requirements: Developing an Effective Software Requirements Specification (SRS) Document

Aziz Zahran[1], Setyawan Widyarto[2]
Universitas Budi Luhur, Jakarta[1], Universiti Selangor, Malaysia[2]
swidyarto@unisel.edu.my[2]

***Abstract*—The Software Requirement Specification (SRS) is a comprehensive document that outlines the functional and non-functional requirements of a software application. It serves as a contract between the client and the development team, detailing what the software must accomplish and the constraints under which it must operate. The SRS includes descriptions of the software's intended purpose, user interfaces, system interactions, performance standards, and security considerations. It is crucial for ensuring that all stakeholders have a clear and shared understanding of the project's goals and deliverables. By providing a detailed blueprint, the SRS minimizes the risk of misunderstandings and errors during the development process, ensuring that the final product meets the client's needs and expectations. Additionally, the SRS serves as a reference point for validation and verification activities, helping to maintain project scope and quality standards throughout the software development lifecycle.**

Keyword : Client, Deliverables, Development team, Functional requirements, Non-functional requirements, Software Requirement Specification (SRS)

I. Introduction

The Software Requirement Specification (SRS) is a critical document in the software development lifecycle, serving as the foundational blueprint for any software project. It meticulously outlines both functional and non-functional requirements, providing a clear and comprehensive description of what the software should achieve and the constraints within which it must operate. The SRS is designed to ensure that all stakeholders, including clients, developers, and project managers, have a unified understanding of the project's objectives, scope, and deliverables. This document not only sets the expectations for the development team but also acts as a reference point for future project phases, including design, implementation, testing, and maintenance.

In addition to defining the software's intended functionality, the SRS addresses various aspects such as user interfaces, system interactions, performance criteria, and security measures. It plays a crucial role in risk management by identifying potential challenges and constraints early in the development process, thereby reducing the likelihood of costly errors and misunderstandings. By providing a detailed and structured approach to requirement gathering and documentation, the SRS helps in aligning the project's outcomes with the client's needs and expectations. Ultimately, the SRS is indispensable for ensuring that the final software product is of high quality, meets the specified requirements, and delivers value to its users.

II. Literature Review

The importance of Software Requirement Specifications (SRS) in software development has been widely recognized in academic and professional literature. Kurniawan et al., n.d. (2023) highlights that an SRS is essential for establishing a mutual understanding between stakeholders and the development team. It defines the functional and non-functional requirements clearly, which is crucial for successful project execution. Githa Ananta (2022), a well-drafted SRS reduces ambiguity and ensures that the final product meets the users' needs and expectations. This is supported by studies showing that projects with detailed SRS documents have higher success rate (Susilowati & Kusuma, 2019).

A significant body of research emphasizes the role of SRS in risk management. Aprilia & Achsin Samas (2024) argues that early identification of potential risks and challenges through a comprehensive SRS can mitigate the chances of project

Proceedings of the Informatics Conference Vol. 11 No. 22 (2025): Requirements Engineering Methodologies ISSN: 2477-5894

failure. The detailed documentation helps in anticipating possible issues and planning accordingly. This perspective is reinforced by Nasrullah et al., n.d.(2023), who state that an SRS serves as a critical tool for managing scope creep and ensuring that the project remains on track. Their findings suggest that projects with thorough requirement specifications are more likely to stay within budget and on schedule.

Moreover, the clarity provided by an SRS aids in improving communication among stakeholders. Kurniawan et al., n.d.-b, (2023) point out that an SRS acts as a communication bridge, facilitating clear and precise discussions between clients and developers. This improved communication leads to a better understanding of user requirements and expectations. Similarly,(Nugraha, n.d. 2022) highlight that effective communication through an SRS can prevent misunderstandings and ensure that the development team fully comprehends the project's goals. This aspect is critical for aligning the project outcomes with client needs.

The use of SRS also extends to enhancing project validation and verification processes. According to Kotonya and Sommerville (1998), an SRS provides a benchmark against which the software can be tested. This ensures that the developed software aligns with the specified requirements and performs as expected. The authors argue that without a well-defined SRS, it is challenging to carry out effective testing and validation. Pohl (2010) further elaborates that an SRS facilitates traceability, allowing for systematic verification of each requirement throughout the development process (Hadi Waryanto, 2012).

Finally, the evolution of SRS practices reflects the changing dynamics of software development methodologies. Agile and iterative development models have introduced new approaches to requirement documentation. As noted by Beck (2000), while traditional SRS documents are detailed and comprehensive, Agile methodologies favor more flexible and dynamic requirement gathering techniques. However, even in Agile environments, the core principles of clearly defining requirements and maintaining stakeholder alignment remain vitaL, supports this view by suggesting that user stories in Agile can serve a similar purpose to traditional SRS, ensuring that requirements are well-understood and implemented effectivel (Baskoro et al., 2021).

III. System Architecture

The architecture of the Software Requirement Specification (SRS) system is designed to ensure robustness, scalability, and ease of maintenance. At the core of the system is a multi-tier architecture comprising the presentation layer, application layer, and data layer. The presentation layer, implemented using modern web technologies such as React.js or Angular, provides a user-friendly interface for stakeholders to interact with the system. This layer is responsible for rendering the SRS documents, gathering user inputs, and ensuring responsive design across various devices. It communicates with the application layer through RESTful APIs, ensuring a clear separation of concerns.

The application layer forms the backbone of the SRS system, handling the business logic and processing user requests. Built using a combination of Node.js and Express.js, this layer is designed to be highly modular, allowing for easy integration of additional features and services. It manages the creation, modification, and validation of SRS documents, implementing business rules and workflows necessary for comprehensive requirement gathering. The application layer also incorporates authentication and authorization mechanisms to ensure secure access control. For enhanced performance and scalability, microservices architecture can be employed, where each service is independently deployable and can communicate via lightweight protocols such as gRPC or HTTP/2.

The data layer is responsible for persistent storage and retrieval of SRS documents and associated metadata. PostgreSQL is chosen as the primary database management system due to its robustness and support for complex queries and transactions. Additionally, an Object-Relational Mapping (ORM) tool like Sequelize is used to facilitate database interactions, ensuring a clean and maintainable codebase. To ensure data integrity and availability, the database is set up with replication and backup strategies. Furthermore, for efficient full-text search capabilities, Elasticsearch can be integrated, allowing users to perform quick and precise searches across SRS documents. The overall system architecture is designed to support continuous integration and continuous deployment (CI/CD) pipelines, enabling automated testing, deployment, and monitoring to maintain high system reliability and performance.

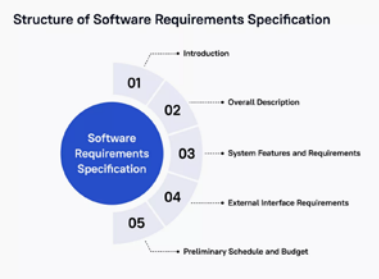

Fig.1 Structure Software Requirement Spesification

The diagram illustrates the structure of a Software Requirements Specification (SRS), which comprises five key sections. The Introduction provides an overview of the document, including its purpose, scope, definitions, and references. The Overall Description outlines the system context, including product perspective, functionality, primary users, constraints, assumptions, and dependencies. The System Features and Requirements section details all functional

Proceedings of the Informatics Conference Vol. 11 No. 22 (2025): Requirements Engineering Methodologies ISSN: 2477-5894

requirements, describing the inputs, processes, and outputs associated with each feature. The External Interface Requirements specify the necessary external interfaces, including user, hardware, software, and communication interfaces, ensuring proper interaction with external components. Finally, the Preliminary Schedule and Budget offers initial estimates of the project timeline and budget, including development milestones and cost projections. This well-organized SRS structure ensures that all critical aspects of the software project are clearly documented, reducing the risk of misunderstandings and ensuring that the final system meets user needs.

IV. Methods

To develop a comprehensive Software Requirements Specification (SRS) document, the PICO framework can be effectively utilized, ensuring a systematic and thorough approach. The first component, Population (P), involves identifying the key stakeholders such as clients, end-users, project managers, and developers. This step is crucial for tailoring the requirements to meet the diverse needs and expectations of those who will interact with or be affected by the software. Developing detailed user personas helps in understanding the different perspectives and specific requirements of the user population, ensuring that the final product is user-centric.

Next, the Intervention (I) phase involves implementing various requirement elicitation techniques like interviews, surveys, workshops, and brainstorming sessions to gather comprehensive information from stakeholders. Additionally, creating prototypes and mockups of the proposed system features and user interfaces allows stakeholders to visualize the system early on, facilitating feedback and ensuring that requirements are well-understood and agreed upon. This hands-on approach helps in refining the requirements and aligning them with stakeholder expectations.

The Comparison (C) component focuses on analyzing alternative solutions and benchmarking against similar systems or industry standards. By evaluating different technologies, frameworks, and design options, the most effective solution for the project can be determined. This comparison helps set realistic and achievable goals for the software project. Finally, the Outcome (O) phase involves documenting the gathered and analyzed requirements in a detailed and structured SRS document, including both functional and non-functional requirements. Validation and verification activities, such as peer reviews and requirement-based testing, ensure the accuracy, completeness, and feasibility of the documented requirements. Obtaining formal stakeholder approval is the final step, confirming that the requirements align with the project goals and stakeholder needs. Using the PICO framework, the requirement gathering and analysis process becomes more structured and focused, enhancing the overall quality and success of the software development project.

V. Discussions

The development of comprehensive software requirements is a cornerstone of successful software engineering. Software Requirements Specification (SRS) documents play a vital role in bridging the gap between stakeholders' expectations and the technical implementation by the development team. A well-crafted SRS ensures that all stakeholders have a unified understanding of what the software should achieve, thus minimizing the risk of misunderstandings and project failure.

A key aspect of developing effective software requirements is stakeholder engagement. Requirements must be gathered from a diverse range of stakeholders, including end-users, clients, project managers, and developers. This ensures that the software will meet the needs of its users and fulfill the business objectives. Techniques such as interviews, surveys, workshops, and brainstorming sessions are essential for collecting detailed and accurate requirements. Moreover, developing user personas and use cases helps in visualizing the needs and interactions of different user groups, which is crucial for creating user-centric software.

The analysis of requirements involves not only understanding what the software needs to do (functional requirements) but also the conditions under which it must operate (non-functional requirements). Functional requirements define specific behaviors or functions, such as data processing, user interactions, and system operations. Non-functional requirements, on the other hand, specify criteria such as performance, security, usability, and scalability. Both types of requirements are critical for ensuring that the software performs well in real-world conditions and delivers a satisfactory user experience.

One of the challenges in defining software requirements is dealing with change. Requirements can evolve due to changes in business processes, user needs, or technological advancements. Thus, it is essential to have a flexible and iterative approach to requirement management. Agile methodologies, for instance, promote continuous stakeholder feedback and iterative development, allowing for adjustments to requirements throughout the project lifecycle. This adaptability helps in addressing changes promptly and maintaining alignment with stakeholder expectations.

Validation and verification of requirements are crucial to ensure their correctness and feasibility. Validation involves checking that the requirements accurately reflect the needs and expectations of stakeholders. Techniques such as prototyping and user testing are effective for validating requirements. Verification, on the other hand, ensures that the requirements are feasible and can be implemented within the given

Proceedings of the Informatics Conference Vol. 11 No. 22 (2025): Requirements Engineering Methodologies ISSN: 2477-5894

constraints. This includes peer reviews, requirement-based testing, and feasibility studies. Together, validation and verification activities help in delivering a reliable and high-quality software product.

In conclusion, the process of developing software requirements is fundamental to the success of a software project. It involves thorough stakeholder engagement, detailed analysis of functional and non-functional requirements, and flexible management of changes. Additionally, rigorous validation and verification ensure that the requirements are accurate and feasible. A well-documented SRS serves as a blueprint for the development team, guiding the project to meet its objectives and deliver value to its users.

VI. Conclusion

The development of comprehensive software requirements is essential for the success of any software project. The Software Requirements Specification (SRS) document serves as a critical tool in this process, ensuring that all stakeholders have a clear and unified understanding of the project's goals, functionalities, and constraints. By engaging stakeholders through various requirement gathering techniques such as interviews, surveys, workshops, and brainstorming sessions, diverse perspectives are incorporated, leading to more accurate and user-centric requirements.

A thorough analysis of both functional and non-functional requirements is crucial for creating a robust and reliable software system. Functional requirements outline the specific actions the software must perform, while non-functional requirements define the conditions under which the software operates. Addressing both aspects ensures the software not only meets its intended purpose but also performs well in real-world scenarios.

Adopting a flexible and iterative approach to requirement management, such as Agile methodologies, allows for continuous stakeholder feedback and the ability to adapt to changes throughout the project lifecycle. This adaptability is essential for maintaining alignment with evolving business processes and user needs.

Validation and verification activities, including prototyping, user testing, peer reviews, and feasibility studies, are critical for ensuring the accuracy, completeness, and feasibility of the requirements. These activities help in mitigating risks, reducing misunderstandings, and ensuring that the final software product aligns with stakeholder expectations and delivers high quality.

In conclusion, a well-crafted SRS document, developed through a structured and thorough process, is fundamental to the successful execution of a software project. It serves as a guiding blueprint for the development team, ensuring that the project stays on track, meets its objectives, and ultimately provides value to its users.

References

Aprilia, T., & Achsin Samas, M. (2024). Implementasi Software Requirement Specification dan Waterfall Model pada SIPODANG berbasis Android. Indonesian Journal on Software Engineering (IJSE), 10(1). Retrieved from http://ejournal.bsi.ac.id/ejurnal/index.php/ijse

Baskoro, F., Andrahsmara, R. A., Darnoto, B. R. P., & Tofan, Y. A. (2021). A systematic comparison of software requirements classification. IPTEK The Journal for Technology and Science, 32(3), 184. https://doi.org/10.12962/j20882033.v32i3.13005

Githa Ananta, V. (2022). Analisis kebutuhan perangkat lunak sistem informasi back office (Studi kasus Hublang dan Teknik Perumdam Among Tirto Kota Batu). Retrieved from https://jurnal.poliwangi.ac.id/index.php/session

Hadi Waryanto, N. (2012). Software Requirements Specification SINAPRA berbasis sistem informasi terpadu. Jurnal Ilmiah, 7(2).

Kurniawan, Y., & Paulus Lucky T. I. (n.d.-a). Software Requirement Specification Sistem Informasi Manajemen dan Geografis Pemetaan Sumber Daya Air. Kurawal: Jurnal Teknologi, Informasi dan Industri. Retrieved from https://jurnal.machung.ac.id/index.php/kurawal

Kurniawan, Y., & Paulus Lucky T. I. (n.d.-b). Software Requirement Specification Sistem Informasi Manajemen dan Geografis Pemetaan Sumber Daya Air. Kurawal: Jurnal Teknologi, Informasi dan Industri. Retrieved from https://jurnal.machung.ac.id/index.php/kurawal

Nasrullah, M., Dwi Angresti, N., Harits Suryawan, S., & Faizal Mahananto, D. (n.d.). Requirement engineering terhadap virtual team pada proyek software engineering. Journal of Advances in Information and Industrial Technology (JAIIT), 3(1).

Nugraha, D. W. (n.d.). Software Requirement dalam Membangun Sistem Informasi Pelayanan Publik.

Proceedings of the Informatics Conference Vol. 11 No. 22 (2025): Requirements Engineering Methodologies ISSN: 2477-5894

Susilowati, M., & Kusuma, A. A. (2019). Software Requirement Specification Sistem Informasi Manajemen. SMARTICS Journal, 5(1), 27–33. https://doi.org/10.21067/smartics.v5i1.3444