**IEEE Recommended Practice for Software Requires Specifications** **sponsored by the Software Engineering Standards Committee 1998: Summary**

**INTRODUCTION**

IEEE’s guide to software requirement specifications (SRS) is meant to assist the customer and supplier with designing and implementing software. IEEE reviews and updates its recommended practices every five years to ensure its standards are high quality. An SRS should provide five benefits: establish the software’s function, reduce development effort, provide a realistic basis for estimating costs, and create a basis for continual development. An SRS is meant to address the software’s functionality, its interactions with the environment, performance, portability, and implementation constraints. Before writing an SRS, the software’s operating environment should be considered. The operating environment could either be inside a larger system or standalone. If the operating environment is inside another larger system, then the SRS needs to note how the two interface together.

**QUALITIES OF A GOOD SRS**

A good SRS has six distinct qualities: correctness, lack ambiguity, consistency, modifiability, traceability, and verifiability. An SRS is considered complete if it contains all specific requirements; consistency is met when an SRS’s requirements are internally consistent with other documents; verifiability is the ability to reasonably verify all requirements. Modifiability is the ability to edit the structure and style to fit new requirements; traceability is met when the origin of requirements is clear. To meet these qualities an SRS should be jointly prepared by the customer and supplier because both have their strengths and weaknesses. This is because a customer typically does not understand software design, and a supplier does not typically understand the scope of the customer’s problem. Other elements that make a good SRS is having clear and concise definitions for terms, and references to documents used which are outside the SRS.

**SECTIONS OF A GOOD SRS**

According to IEEE, a good SRS has three major sections: introduction, overall description, and specific requirements. A table of contents should be included to improve readability. Section 1 of a good SRS is the introduction, which contains a general overview of the SRS. This section contains background information on the project, such as the project’s scope. Section 2 of the SRS is the overall description. This section should describe the factors that affect the requirements, but not the requirements themselves. Section 2 contains background information for section 3, specific requirements. The specific requirements section should contain information specific enough to enable designers to build a system that satisfies those requirements. It is especially important that the specific requirements section lacks ambiguity.

**SECTION 1: INTRODUCTION**

The introduction section should contain five subsections: purpose, scope, definitions, references, and an overview. The purpose subsection describes the software’s purpose, and intended audience; the scope subsection should identify the software by name, and explain what the software will and will not do. The SRS’s definitions subsection should provide a list of all terms and their definitions that are necessary for full understanding. These items are meant to be referenced later by the appendix. The references subsection should contain a list of all documents referenced throughout the SRS and the overview subsection should describe what the rest of the SRS contains and its organization style.

**SECTION 2: OVERALL DESCRIPTION**

Section 2 is the overall description, which is meant to describe factors that affect requirements. This section contains six distinct subsections: perspective, functions, user characteristics, constraints, assumptions/dependencies, and the organization of requirements. Product perspective contains information on how the product interfaces with aspects outside of the software. This could be how the product interfaces with the hardware, or how the product interacts with a larger system. The next subsection is product functions, which is meant to provide a summary of the software’s functionality. The user characteristics section should describe characteristics of the intended user, such as education level or expertise. The constraints section should provide a list of factors that will limit a developer’s options such as government regulations. Assumptions/dependencies list factors that affect the specific requirements. This is meant to be background information to assist in understanding section 3.

**SECTION 3: SPECIFIC REQUIREMENTS**

Section 3 is entirely composed of specific requirements. There are many types of specific requirements, these include functional requirements, performance requirements, and a multitude of others. These requirements need to be specific enough to allow a system designer to build a system that can support the product. For these requirements to be useful, they need to be organized. There is no one catch-all way to organize an SRS, so multiple methods can be used and combined.

**ANNEXES**

The last section of IEEE’s SRS document is the annexes, which consist of two subsections. Under the first subsection, IEEE outlines multiple SRS templates following different organization styles. These templates can be used to produce a high-quality SRS based on specific organizational needs. The next subsection describes how to comply with another standard outside SRS documentation, called IEEE/EIA 12207.1-1997. This standard is meant to serve as a basis for the life cycle of software. The purpose of this subsection is to provide guidance to those who want to comply with both standards within the SRS.

**CONCLUSION**

The end goal of an SRS is to be understandable by its audience. This is why IEEE’s guide to writing an SRS is useful. Providing a common standard ensures that the SRS will contain all necessary information, organized understandably.

**DISCUSSION**

Documentation standards are very important to computer science. This is because they provide a reasonable framework to document aspects of software. In this case the documentation standards are for software requirements, but it also applies to all other types of documentation. Standards also raise the bar for what should be acceptable for documentation, and increases understanding. This is because when the framework of documentation is understood, a reader only needs to understand its content, not its structure.