Jaewon Heo

CS 325

1/17/2022

Lab2

Summary – Lab 2

This is a summary of IEEE Recommended Practice for Software Requirements Specifications sponsored by Software Engineering Standards Committee of the IEEE Computer Society. This document helps Software customers describe their requirements accurately and helps software suppliers understand what customers’ needs are. This practice describes recommended approaches for the specification of software requirements into five sections which are the scope of this recommended practice, lists of references of other standards, definitions of specific terms used, background information for writing a good software requirements specification (SRS), and discussion of essential parts of an SRS. References of this practice provided and could be seen at the original document.An SRS is useful for software that has yet to be developed. However, applying IEEE’s recommendations to developed software is ineffective.

There are terms which provided to readers should consider when writing an SRS. The word contract is a legal document agreed between the customer and supplier. A contract includes technical and organizational requirements, cost, and schedule for building a software. The word customer means a person who are purchasing the product. Supplier is a person who make the software for customer, and users are people who are interacting the software.

The SRS should specify the performance of functions in a specific environment in the software. These are reflecting representatives of customers, or suppliers. The SRS should consist of the program’s functionalities, external interfaces, performances, attributes, and implementations. Design or project requirements should be avoided placing in the SRS.

Considering the SRS’s role in theproject plan is important. If the software is essential to a larger system’s functionality, then the SRS will need to indicate the software’s relationship with the system. The SRS should agree with the requirements of the larger system. SRS writers should consist with requirements assigned. This means that the SRS should define every software requirement correctly, avoid describing any design or implementations, and avoid adding extra restrictions on the software which are out of system requirements.

There are number of characteristics of a good SRS. The SRS must be correct, and its contents should match with its targeted software. A good SRS ensure users or customers that it correctly reflected their needs. An SRS should be clear to read. Every requirement should have one meaning to avoid ambiguity. It should be written in natural language such as English. However, since natural language is naturally ambiguous, an SRS written in natural language should revised and be corrected. The SRS should also be written in a requirements specification language.

An SRS should include every significant requirement which is related to a software’s functionality, performance and other attributes. It is important to verify both valid input and invalid input values of a particular program. Labeling and referencing all content in an SRS is also important.

The SRS should be consistent. It should agree with higher level documents which are related such as a system requirements specification. A good SRS should not include content which conflicts with specified characteristics of real-world objects, between two specified actions, or between requirements.

Every requirement in SRS is important but not equally. An SRS writer should rank for importance. The writer should also identify the stability of for a functionality of a program. This may be stated as quantitative measure of changes to requirements.

For the other requirements of a good SRS, a writer should avoid using words such as “well”, “good”, or “usually”. These terms introduce unneeded ambiguity. Contents of the document should be coherent, essential, and separated. Redundancy itself is not an error, but it can cause some errors, eventually lead to inconsistency. Lastly, a good SRS should be traceable by both backward and forward.

Prototyping is also part of an SRS’s requirement’s . The prototype of project is preferred to user hence powerful for receiving quick feedback from users. Feedback can save time for SRS writers and for development.

The SRS writers should reveal its outputs of the system, such as partitioning the software into modules, allocating functions to modules, describing the flow of information or control between modules, and choosing data structures. Writers should also specify the certain project’s cost, delivery schedules, report procedures, development methods, quality assurance, validation and verification criteria, and acceptance procedures.

For writing the SRS, the introduction should be included which contains the purpose of writing this SRS, and details of contents of an SRS such as software’s name and its functionality. This section of the SRS is interpreting terms, acronyms, and abbreviations and specify references.

Lastly the SRS should demonstrate descriptions of the product’s functionalities, its internal, external interface, and operations. The description also includes summary of functions and states requirements for users to utilize the software. Constraints, functions delayed being included must be specified.

-Discussion

This article describes how to write the good SRS. I think an SRS is a tool of communication between software customers and developers and for writers. SRS writers should produce a good SRS to ease this communication.