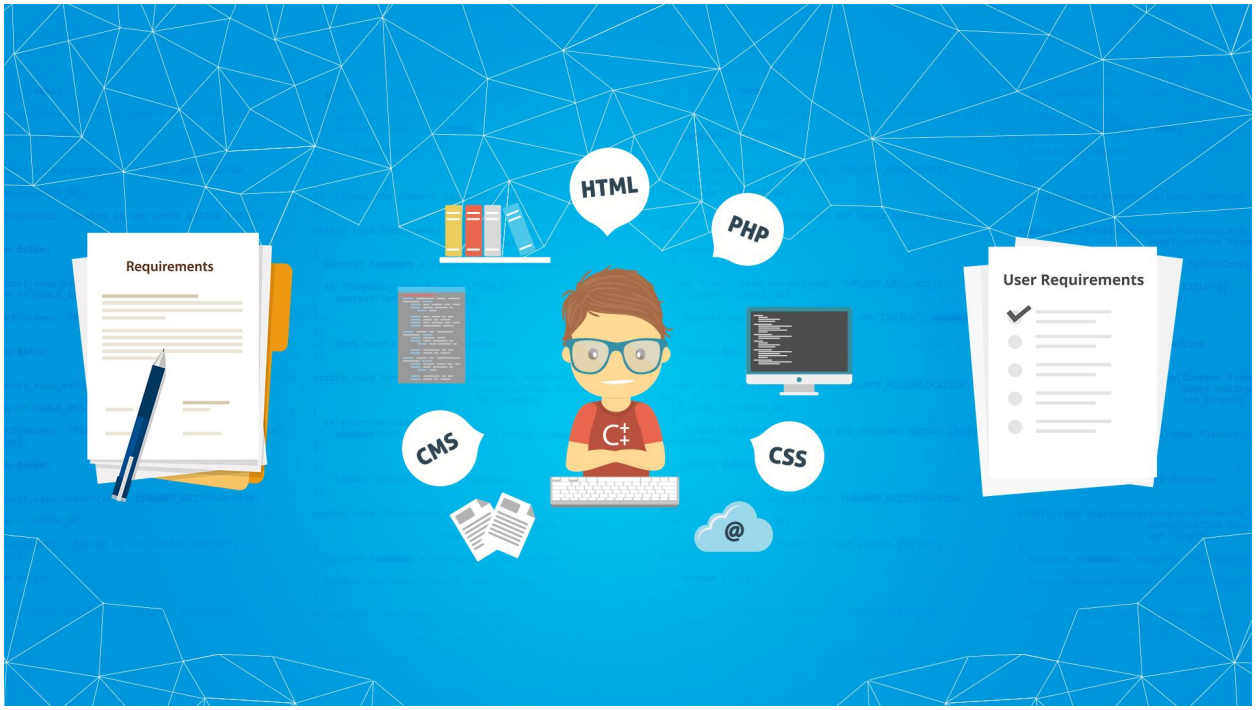
IEEE Standard
for
Software Requirements Specifications

Report



Jayaweera W.M.P.N.
TG/2018/392
Department of ICT
Faculty of Technology
University of Ruhuna

What is IEEE?

The Institute of Electrical and Electronics Engineers Standards Association (IEEE-SA) is an organization within IEEE that develops global standards in a broad range of industries, including: power and energy, biomedical and health care, information technology and robotics, telecommunication and home automation, transportation, nanotechnology, information assurance, and many more.

IEEE-SA has developed standards for over a century, through a program that offers balance, openness, fair procedures, and consensus. Technical experts from all over the world participate in the development of IEEE standards.

IEEE-SA is not a body formally authorized by any government, but rather a community.

What is Software Requirements Specifications?

SRS is a specification for a particular software product, program, or set of programs that perform certain functions in a specific environment. The SRS can be written by one or more representatives of the supplier, one or more representatives of the customer, or both.

The basic issues that the SRS writer shall address are the following:

1. Functionality. What is the software supposed to do?
2. External interfaces. How does the software interact with people, the system's hardware, other hard-ware, and other software?
3. Performance. What is the speed, availability, response time, recovery time of various software func-tions, etc.?
4. Attributes. What are the portability, correctness, maintainability, security, etc. considerations?
5. Design constraints imposed on an implementation. Are there any required standards in effect, simple-mentation language, policies for database integrity, resource limits, operating environments etc.?

Considerations for producing a good SRS

Environment of the SRS

Since the SRS has a specific role to play in the software development process, the SRS writers should be careful not to go beyond the bounds of that role. This means the SRS

1. Should correctly define all of the software requirements. A software requirement may exist because of the nature of the task to be solved or because of a special characteristic of the project.
2. Should not describe any design or implementation details. These should be described in the design stage of the project.
3. Should not impose additional constraints on the software. These are properly specified in other documents such as a software quality assurance plan.Therefore, a properly written SRS limits the range of valid designs, but does not specify any particular design.

Characteristics of a good SRS

An SRS should be

1. Correct;
2. Unambiguous;
3. Complete;
4. Consistent;
5. Ranked for importance and/or stability;
6. Verifiable;
7. Modifiable;
8. Traceable.

Joint preparation of the SRS

The software development process should begin with supplier and customer agreement on what the completed software must do. This agreement, in the form of an SRS, should be jointly prepared. This is important because usually neither the customer nor the supplier is qualified to write a good SRS alone.

1. Customers usually do not understand the software design and development process well enough to write a usable SRS.

2. Suppliers usually do not understand the customer's problem and field of endeavor well enough to specify requirements for a satisfactory system.

Therefore, the customer and the supplier should work together to produce a well-written and completely understood SRS. A special situation exists when a system and its software are both being defined concurrently. Then the func-tionality, interfaces, performance, and other attributes and constraints of the software are not predefined, but rather are jointly defined and subject to negotiation and change. In particular, an SRS that does not comply with the requirements of its parent system specification is incorrect.

This recommended practice does not specifically discuss style, language usage, or techniques of good writ-ing. It is quite important, however, that an SRS be well written. General technical writing books can be used for guidance.

SRS evolution

The SRS may need to evolve as the development of the software product progresses. It may be impossible to specify some details at the time the project is initiated (e.g., it may be impossible to define all of the screen formats for an interactive program during the requirements phase). Additional changes may ensue as defi-ciencies, shortcomings, and inaccuracies are discovered in the SRS.Two major considerations in this process are the following:

1. Requirements should be specified as completely and thoroughly as is known at the time, even if evolutionary revisions can be foreseen as inevitable. The fact that they are incomplete should be noted.
2. A formal change process should be initiated to identify, control, track, and report projected changes.Approved changes in requirements should be incorporated in the SRS in such a way as to
    a. Provide an accurate and complete audit trail of changes;
    b. Permit the review of current and superseded portions of the SRS.

Prototyping

Prototyping is used frequently during the requirements portion of a project. Many tools exist that allow a prototype, exhibiting some characteristics of a system, to be created very quickly and easily. Prototypes are useful for the following reasons:

1. The customer may be more likely to view the prototype and react to it than to read the SRS and react to it. Thus, the prototype provides quick feedback.
2. The prototype displays unanticipated aspects of the systems behavior. Thus, it produces not only answers but also new questions. This helps reach closure on the SRS.

3. An SRS based on a prototype tends to undergo less change during development, thus shortening development time.

A prototype should be used as a way to elicit software requirements. Some characteristics such as screen or report formats can be extracted directly from the prototype. Other requirements can be inferred by running experiments with the prototype.

Embedding design in the SRS

A requirement specifies an externally visible function or attribute of a system. A design describes a particu-lar subcomponent of a system and/or its interfaces with other subcomponents. The SRS writers should clearly distinguish between identifying required design constraints and projecting a specific design. Note That every requirement in the SRS limits design alternatives. This does not mean, though, that every require-ment is design.

The SRS should specify what functions are to be performed on what data to produce what results at what location for whom. The SRS should focus on the services to be performed. The SRS should not normally specify design items such as the following:

1. Partitioning the software into modules;
2. Allocating functions to the modules;
3. Describing the flow of information or control between modules;
4. Choosing data structures.

Embedding project requirements in the SRS

The SRS should address the software product, not the process of producing the software product.
Project requirements represent an understanding between the customer and the supplier about contractual matters pertaining to production of software and thus should not be included in the SRS. These normally include items such as

1. Cost;
2. Delivery schedules;
3. Reporting procedures;
4. Software development methods;
5. Quality assurance;
6. Validation and verification criteria;
7. Acceptance procedures.

Project requirements are specified in other documents, typically in a software development plan, a software quality assurance plan, or a statement of work.

The parts of an SRS

This clause discusses each of the essential parts of the SRS. While an SRS does not have to follow this outline or use the names given here for its parts, a good SRS should include all the information discussed here.

Introduction

The introduction of the SRS should provide an overview of the entire SRS. It should contain the following subsections:

1. Purpose;
2. Scope;
3. Definitions, acronyms, and abbreviations;
4. References;
5. Overview.

Overall description

This section of the SRS should describe the general factors that affect the product and its requirements. This Section does not state specific requirements. Instead, it provides a background for those requirements, and makes them easier to understand.

This section usually consists of six subsections, as follows:

1. Product perspective;
2. Product functions;
3. User characteristics;
4. Constraints;
5. Assumptions and dependencies;
6. Apportioning of requirements.

Specific requirements

This section of the SRS should contain all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated

requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system, and all functions performed by the system in response to an input or in support of an output. As this is often the largest and most impor-tant part of the SRS, the following principles apply:

1. Specific requirements should be stated in conformance with all the characteristics described in Characteristics of a good SRS.
2. Specific requirements should be cross-referenced to earlier documents that relate.
3. All requirements should be uniquely identifiable.
4. Careful attention should be given to organizing the requirements to maximize readability.

External interfaces

This should be a detailed description of all inputs into and outputs from the software system.

It should include both content and format as follows:

1. Name of item;
2. Description of purpose;
3. Source of input or destination of output;
4. Valid range, accuracy, and/or tolerance;
5. Units of measure;
6. Timing;
7. Relationships to other inputs/outputs;
8. Screen formats/organization;
9. Window formats/organization;
10. Data formats;
11. Command formats;
12. End messages.

Functions

Functional requirements should define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as “shall” statements starting with “The system shall...”

These include

1. Validity checks on the inputs
2. Exact sequence of operations

3. Responses to abnormal situations, including
    a. Overflow
    b. Communication facilities
    c. Error handling and recovery
4. Effect of parameters
5. Relationship of outputs to inputs, including
    a. Input/output sequences
    b. Formulas for input to output conversionIt may be appropriate to partition the functional requirements into subfunctions or subprocesses. This does not imply that the software design will also be partitioned that way.

Performance requirements

This subsection should specify both the static and the dynamic numerical requirements placed on the soft-ware or on human interaction with the software as a whole. Static numerical requirements may include the following:

1. The number of terminals to be supported;
2. The number of simultaneous users to be supported;
3. Amount and type of information to be handled.

Logical database requirements

This should specify the logical requirements for any information that is to be placed into a database. This May include the following:

1. Types of information used by various functions;
2. Frequency of use;
3. Accessing capabilities;
4. Data entities and their relationships;
5. Integrity constraints;
6. Data retention requirements.

Standards compliance

This should specify design constraints that can be imposed by other standards, hardware limitations, etc.

Standards compliance

This subsection should specify the requirements derived from existing standards or regulations. They may include the following:

1. Report format;
2. Data naming;
3. Accounting procedures;
4. Audit tracing.

For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.

Software system attributes

There are a number of attributes of software that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified.

1. Reliability
2. Availability
3. Security
4. Maintainability
5. Portability

Organizing the specific requirements

1. System mode
2. User class
3. Objects
4. Feature
5. Stimulus
6. Response
7. Functional hierarchy
8. Additional comments

Additional comments

In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification. Any additional requirements may be put in a separate section at the end of the SRS.

There are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts may prove helpful; when organizing by object, object-oriented analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; and when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.

Supporting information

The supporting information makes the SRS easier to use. It includes the following:

1. Table of contents;
2. Index;
3. Appendixes.

Appendixes

The appendixes are not always considered part of the actual SRS and are not always necessary. They may include

1. Sample input/output formats, descriptions of cost analysis studies, or results of user surveys;
2. Supporting or background information that can help the readers of the SRS;
3. A description of the problems to be solved by the software;
4. Special packaging instructions for the code and the media to meet security, export, initial loading, orother requirements.

When appendixes are included, the SRS should explicitly state whether or not the appendixes are to be considered part of the requirements.

-END-