How to write a software requirements specification

by Robert Japenga

What Makes a Great Software Requirements Specification?

There are many good definitions of System and Software Requirements Specifications that will provide us a good basis upon which we can both define a great specification and help us identify deficiencies in our past efforts. There is also a lot of great stuff on the web about writing good specifications. The problem is not lack of knowledge about how to create a correctly formatted specification or even what should go into the specification. The problem is that we don't follow the definitions out there.

We have to keep in mind that the goal is not to create great specifications but to create great products and great software. Can you create a great product without a great specification? Absolutely! You can also make your first million through the lottery – but why take your chances? Systems and software these days are so complex that to embark on the design before knowing what you are going to build is foolish and risky.

The IEEE (www.ieee.org) is an excellent source for definitions of System and Software Specifications. As designers of real-time, embedded system software, we use IEEE STD 830-1998 as the basis for all of our Software Specifications unless specifically requested by our clients. Essential to having a great Software Specification is having a great System Specification. The equivalent IEEE standard for that is IEEE STD 1233-1998. However, for most purposes in smaller systems, the same templates can be used for both.

What are the benefits of a Great SRS?

The IEEE 830 standard defines the benefits of a good SRS:

Establish the basis for agreement between the customers and the suppliers on what the software product is to do. The complete description of the functions to be performed by the software specified in the SRS will assist the potential users to determine if the software specified meets their needs or how the software must be modified to meet their needs. [NOTE: We use it as the basis of our contract with our clients all the time].

Reduce the development effort. The preparation of the SRS forces the various concerned groups in the customer's organization to consider rigorously all of the requirements before design begins and reduces later redesign, recoding, and retesting. Careful review of the requirements in the SRS can reveal omissions, misunderstandings, and inconsistencies early in the development cycle when these problems are easier to correct.

Provide a basis for estimating costs and schedules. The description of the product to be developed as given in the SRS is a realistic basis for estimating project costs and can be used to obtain approval for bids or price estimates. [NOTE: Again, we use the SRS as the basis for our fixed price estimates]

Provide a baseline for validation and verification. Organizations can develop their validation and Verification plans much more productively from a good SRS. As a part of the development contract, the SRS provides a baseline against which compliance can be measured. [NOTE: We use the SRS to create the Test Plan].

*Facilitate transfer.*The SRS makes it easier to transfer the software product to new users or new machines. Customers thus find it easier to transfer the software to other parts of their organization, and suppliers find it easier to transfer it to new customers.

Serve as a basis for enhancement. Because the SRS discusses the product but not the project that developed it, the SRS serves as a basis for later enhancement of the finished product. The SRS may need to be altered, but it does provide a foundation for continued production evaluation. [NOTE: This is often a major pitfall – when the SRS is not continually updated with changes]

What should the SRS address?

Again from the IEEE standard:

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

a) *Functionality.* What is the software supposed to do?
b) *External interfaces.* How does the software interact with people, the system's hardware, other hardware, and other software?
c) *Performance.* What is the speed, availability, response time, recovery time of various software functions, etc.?
d) *Attributes.* What are the portability, correctness, maintainability, security, etc. considerations?
e) *Design constraints imposed on an implementation.* Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

What are the characteristics of a great SRS?

Again from the IEEE standard:

An SRS should be

a) Correct
b) Unambiguous
c) Complete
d) Consistent
e) Ranked for importance and/or stability
f) Verifiable
g) Modifiable
h) Traceable

Correct - This is like motherhood and apple pie. Of course you want the specification to be correct. No one writes a specification that they know is incorrect. We like to say - "Correct and Ever Correcting." The discipline is keeping the specification up to date when you find things that are not correct.

Unambiguous - An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation. Again, easier said than done. Spending time on this area prior to releasing the SRS can be a waste of time. But as you find ambiguities - fix them.

Complete - A simple judge of this is that is should be all that is needed by the software designers to create the software.

Consistent - The SRS should be consistent within itself and consistent to its reference documents. If you call an input "Start and Stop" in one place, don't call it "Start/Stop" in another.

Ranked for Importance - Very often a new system has requirements that are really marketing wish lists. Some may not be achievable. It is useful provide this information in the SRS.

Verifiable - Don't put in requirements like - "It should provide the user a fast response." Another of my favorites is - "The system should never crash." Instead, provide a quantitative requirement like: "Every key stroke should provide a user response within 100 milliseconds."

Modifiable - Having the same requirement in more than one place may not be wrong - but tends to make the document not maintainable.

Traceable - Often, this is not important in a non-politicized environment. However, in most organizations, it is sometimes useful to connect the requirements in the SRS to a higher level document. Why do we need this requirement?

What is the difference between a System Specification and a Software Specification?

Very often we find that companies do not understand the difference between a System specification and a Software Specification. Important issues are not defined up front and Mechanical, Electronic and Software designers do not really know what their requirements are.

The following is a high level list of requirements that should be addressed in a System Specification:

- Define the functions of the system
- Define the Hardware / Software Functional Partitioning
- Define the Performance Specification
- Define the Hardware / Software Performance Partitioning
- Define Safety Requirements
- Define the User Interface (A good user's manual is often an overlooked part of the System specification. Many of our customers haven't even considered that this is the right time to write the user's manual.)
- Provide Installation Drawings/Instructions.
- Provide Interface Control Drawings (ICD's, External I/O)

One job of the System specification is to define the full functionality of the system. In many systems we work on, some functionality is performed in hardware and some in software. It is the job of the System specification to define the full functionality and like the performance requirements, to set in motion the trade-offs and preliminary design studies to allocate these functions to the different disciplines (mechanical, electrical, software).

Another function of the System specification is to specify performance. For example, if the System is required to move a mechanism to a particular position accurate to a repeatability of ± 1 millimeter, that is a System's requirement. Some portion of that repeatability specification will belong to the mechanical hardware, some to the servo amplifier and electronics and some to the software. It is the job of the System specification to provide that requirement and to set in motion the partitioning between mechanical hardware, electronics, and software. Very often the System specification will leave this partitioning until later when you learn more about the system and certain factors are traded off (For example, if we do this in software we would need to run the processor clock at 40 mHz. However, if we did this function in hardware, we could run

the processor clock at 12 mHz). [This implies that a certain level of research or even prototyping and benchmarking needs to be done to create a System spec. I think it is useful to say that explicitly.]

However, for all practical purposes, most of the systems we are involved with in small to medium size companies, combine the software and the systems documents. This is done primarily because most of the complexity is in the software. When the hardware is used to meet a functional requirement, it often is something that the software wants to be well documented. Very often, the software is called upon to meet the system requirement with the hardware you have. Very often, there is not a systems department to drive the project and the software engineers become the systems engineers. For small projects, this is workable even if not ideal. In this case, the specification should make clear which requirements are software, which are hardware, and which are mechanical.

What is the difference between a design requirement and software requirement?

In short, the SRS should not include any design requirements. However, this is a difficult discipline. For example, because of the partitioning and the particular RTOS you are using, and the particular hardware you are using, you may require that no task use more than 1 ms of processing prior to releasing control back to the RTOS. Although that may be a true requirement and it involves software and should be tested – it is truly a design requirement and should be included in the Software Design Document or in the Source code.

Consider the target audience for each specification to identify what goes into what documents.

Marketing/Product Management

Creates a product specification and gives it to Systems. It should define everything Systems needs to specify the product

Systems

Creates a System Specification and gives it to Systems/Software and Mechanical and Electrical Design.

Systems/Software

Creates a Software Specification and gives it to Software. It should define everything Software needs to develop the software.

Thus, the SRS should define everything explicitly or (preferably) by reference that software needs to develop the software. References should include the version number of the target document. Also, consider using master document tools which allow you to include other documents and easily access the full requirements.

Is this do-able? Won't we miss our deadlines if we take the time to do this?

This is a great question. There is no question that there is balance in this process. We have seen companies and individuals go overboard on documenting software that doesn't need to be documented, such as a temporary utility. We have also seen customers kill good products by spending too much time specifying it.

However, the bigger problem is at the other end of the spectrum. We have found that taking the time up front pays dividends down stream. If you don't have time to specify it up front, you probably don't have the time to do the project.

Here are some of our guidelines:

- Spend time specifying and documenting well software that you plan to keep.
- Keep documentation to a minimum when the software will only be used for a short time or has a limited number of users.
- Have separate individuals write the specifications (not the individual who will write the code).
- The person to write the specification should have good communication skills.
- Pretty diagrams can help but often tables and charts are easier to maintain and can communicate the same requirements.
- Take your time with complicated requirements. Vagueness in those areas will come back to bite you later.
- Conversely, watch out for over-documenting those functions that are well understood by many people but for which you can create some great requirements.
- Keep the SRS up to date as you make changes.
- Approximately 20-25% of the project time should be allocated to requirements definition.
- Keep 5% of the project time for updating the requirements after the design has begun.
- Test the requirements document by using it as the basis for writing the test plan.