<Project Name>
Software Requirements Specification [Template]

A. Hornof (ajh) - 5-31-2019 - v1.01

Table of Contents

This document describes the content and general organization to be used for a Software Requirements Specification (SRS) in CIS 422 Software Methodologies. The document is distilled from several primary sources, each of which is cited in the article and listed in the references at the end of the article. Most SRS templates, such as those provided by the IEEE (Institute of Electrical and Electronics Engineers), are longer and more complex than what is shown here. This template distills the SRS down to a set of essential components.

How to use this document: Keep the headings and document structure, and replace or modify the text within each section and subsection with the content described here in that section. (To assist with this, this document will be made available in numerous file formats, such as: HTML, Pages, Text, Word.) Optionally, add another level of structure in Section 2 to organize the specific requirements around major classes of user activities or system behaviors, as described in that section. Note that some of the headings such as "1. The Concept of Operations" need no text other than to briefly describe what is contained within that section. The final SRS should be a self-contained document that clearly describes what the document is, who wrote it, who wrote various sections, and when they wrote them. None of this explanatory text that describes *how to write* the document should be present.

1. SRS Revision History

[This should list every modification to the document. Entries should be ordered chronologically, either in forward or backward chronological order; pick an order and stick to it. The entries listed in the template are for the creation of the template.]

Date	**Author**	**Description**
4-8-2019	ajh	Created the initial document.
4-30-2019	ajh	Added the “revision history” section.
5-6-2019	ajh	Fixed grammar errors pointed out by Prof. Young. (Still need to add the examples he suggested. See “…Michal comments.pages”.)
5-31-2019	ajh	Updated Section 3.5

2. The Concept of Operations (ConOps)

The concept of operations (ConOps) “describes system characteristics for a proposed system from the users’ viewpoint.” (IEEE Std 1362-1998) The ConOps document communicates overall system characteristics to all stakeholders. It should be readable by any stakeholder familiar with the application domain. The ConOps should clarify the context of the software and the capabilities that the system will provide the user (Faulk, 2017).

The sections described here should all be included in the ConOps. They are distilled from IEEE Std 1362-1998. Note that, for a small system, Sections 1.1 through 1.5 could potentially be completed with one or two paragraphs each. The Uses Cases in Section 1.6 should be longer, though.

You can see more sections that can go into a ConOps in Section 4 “Elements of a ConOps document” of IEEE Std 1362-1998 at https://ieeexplore.ieee.org/document/761853 .

2.1. Current System or Situation

This section ‘describes the real-world situation or system (either automated or manual), as it currently exists, that motivates development of the proposed system. This section provides readers with an introduction to the problem domain, which enables readers to better understand the motivation to develop the system that is being specified. This section discusses interfaces, systems, or procedures that are external to the system but essential to understand the current system or situation.' (IEEE Std 1362-1998)

For example, if the system being built will create a new way for students to access required-reading materials for college classes, this section would briefly describe the purpose of assigned reading, the various ways that students are currently informed of reading assignments (course web pages, in-class announcements, etc.), and the way that reading materials are made available to students (bookstore, online handouts, etc.).

2.2. Justification for a New System

This section 'describes the *shortcomings* of the current system or situation that motivate development of a new system or modification of an existing system. This provides a transition from the current system or situation to the description of the proposed system. If there is no current system on which to base changes, this section should indicate as much' (IEEE Std 1362-1998).

This section should provide an objectively measurable justification for the system, based on real-world data, not opinion, hunches, or unsupported statements. For example, a rigorous student survey might indicate that half of all students on campus own a physical or digital copy of required-reading, and that the majority of the digital copies technically violate copyright laws. Similarly, data from reliable sources could note that the cost of college textbooks over the last twenty years has risen at a much higher rate than the cost of other books, and the cost of tuition. Such objective data could support a claim that student-access to required-reading materials needs to be improved.

This section, or possibly the next section, should probably review systems that are currently available that could potentially solve the shortcomings of the existing system or situation, and explain how these systems do not.

Note that sections 1.1 and 1.2 together identify and describe an unmet need. The next section, 1.3, describes at a high level how this need will be met

2.3. Operational Features of the Proposed System

This is where you briefly describe what the new system will do. 'This section describes the key operational features—the essential services and constraints—of the proposed system, at a high-level and without specifying design details. The description should be sufficient to explain how the system fulfills the users' needs, and solves the shortcomings of the current system or situation. In some cases, to clarify the operational details of the proposed system, it may be necessary to suggest some possible design details, such as possible design strategies or specific implementations, but it should be made clear that these are not committed design specifications.' (IEEE Std 1362-1998)

For example, for a required-reading-distribution system, the key operational features could include some means of assisting students with automatically taking high-resolution screenshot after screenshot of an existing shared online textbook viewing system (presuming that such a system does not currently exist), and combining these screenshots into a compete PDF of the book. (Though such a system would seem to violate copyright laws, and it could be argued to be unethical to specify such a system.)

2.4. User Classes

This section provides a brief description of each user class for the proposed system. A user class is group of people that will interact with the system in a roughly similar manner, such as students, instructors, system administrators, book publishers, programmers, or software installers. "Factors that distinguish a user class include common responsibilities, skill levels, work activities, and modes of interaction with the system. Different user classes may have distinct operational scenarios [discussed below] for their interactions with the system. In this context, a user is anyone who interacts with the existing system, including operational users, data entry personnel, system operators, operational support personnel, software maintainers, and trainers." (IEEE Std 1362-1998)

2.5. Modes of Operation

This section describes the various modes of operation for the proposed system. For example, there could be a different mode for each of the following: user, administrator, normal usage, backup, degraded, maintenance, training, student, instructor. Include all of the modes that apply to all user classes. Each user class might have a different mode of operation. Some user classes will have multiple modes. (IEEE Std 1362-1998)

2.6. Operational Scenarios (Also Known as "Use Cases")

"An operational scenario [also known as "Use Cases"] is a step-by-step description of how the proposed system should operate and interact with its users and its external interfaces under a given set of circumstances. Scenarios should be described in a manner that will allow readers to walk through them and gain an understanding of how all the various parts of the proposed system function and interact. The scenarios tie together all parts of the system, the users, and other entities by describing how they interact." (IEEE Std 1362-1998)

'Operational scenarios should describe operational sequences that illustrate the roles of the system, its interactions with users, and interactions with other systems. Operational scenarios should ideally be described for all operational modes and all classes of users identified for the proposed system. Each scenario should include events, actions, stimuli, information, and interactions as appropriate to provide a comprehensive understanding of the operational aspects of the proposed system. Prototypes, storyboards, and other media, such as video or hypermedia presentations, may be used to provide part of this information.' (IEEE Std 1362-1998)

Include, in each operational-scenario (or use-case), a one-sentence description of the scenario, a list of the users or "actors" (from the user classes) involved in the scenario, the preconditions for starting the scenario, and the postconditions (the relevant status of the system and world) after the scenario is completed.

Structure the writing of operational scenarios, or use cases, so that they are easy to read, with headings and numbered steps rather than in paragraph form. For example, note how the following is well-structured to make it easy to read.

Use Case: Get a copy of an online book.
Brief description: This use case describes how a student would make a digital copy of the required reading for a course, for a textbook that is made available to student online, but which the student cannot download to his or her computer.
Actors: A student.
Preconditions:

1. The student has access to an online digital copy of the assigned reading.
2. The reading material can be viewed one full and sufficiently-high-resolution page at a time.
3. The page can be advanced one page with a consistent onscreen button, mouse command, or keyboard command.
4. Each page of reading material is positioned in the exact same rectangle on the computer screen.
5. The student can load applications on the machine accessing the online digital copy of the reading, and has access to substantial (such as 100MB) hard drive space on that machine.

Steps to Complete the Task:

1. The student gains access to the online digital copy of the assigned reading.
2. The user figures out:
    (a) How to display the reading material one page at a time, with the page filling as much of the screen as possible (perhaps even turning the computer screen 90°).
    (b) What commands are available to advance through the reading one page at a time.
3. The user starts up the required-reading-assistant software.

...

Postconditions:
The user has a PDF of all of the required reading, with OCR (optical character recognition) applied, and with a separate chapter within the PDF for each reading assignment or chapter in the book.

You can read more on use cases in the Oracle (2007) White Paper on "Getting Started With Use Case Modeling", available at: https://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf

Note that diagrams can assist with communication.

"Graphical tools should be used wherever possible, especially since ConOps documents should be understandable by several different types of readers. Useful graphical tools include, but are not limited to, work breakdown structures (WBS), N2 charts, sequence or activity charts, functional flow block diagrams, structure charts, allocation charts, data flow diagrams (DFD), object diagrams, context diagrams, storyboards, and entity-relationship diagrams." (IEEE Std 1362-1998).

3. Specific Requirements

This is where the actual requirements are specified. A requirement is a description of a behavior or property that a computer program must have, independent of how that behavior or property is achieved. Requirements must be complete, unambiguous, consistent, and objectively verifiable (see van Vliet, 2008, pp.241-242, for a discussion of these terms). Requirements describe what the system will do, but do not commit to specific *design* details of how the system will do it.

Requirements should be organized in a hierarchy. A good organization (a) makes requirements easier to read and understand because related requirements will be near each other in the document, (b) makes requirements easier to modify and update, and (c) makes it easier to find a specific requirement. There are a number of ways that requirements can be organized to help achieve these goals.

The following section headings provide one way to organize the requirements. You can adapt this organization. For example, sections 2.1, 2.2, 2.3, and 2.4 (see headings below) describe "behavioral requirements" (Faulk, 2013). If a system supports two major user activities, it might be best to describe the behavioral requirements for each activity separately. For example, in a digital deejaying system, the two major activities could be (a) load songs into the system and (b) use the system to play songs. It might be best to fully specify everything in sections 2.1 through 2.4 first for the song-loading activity, and then for the song-playing activity.

Though there is a tradition of distinguishing between "functional" and "non-functional" requirements, with the former describing services provided by the system, and the latter describing constraints on the system and its development, in actual practice this is not a very useful distinction, and is not a good basis for structuring requirements (Faulk, 2013).

Requirements should be prioritized, with each classified as (a) must have, (b) should have, (c) could have, and (d) won't have. These can be recalled with the memory aid of MoSCoW (vanVliet 2008, p.237). When reading requirements, it should be very easy to see how each requirement is classified, such as by having them grouped by priority.

3.1. External Interfaces (Inputs and Outputs)

This section should describe inputs into and outputs from the software system. (ISO/IEC/IEEE 29148:2011)

Each interface description should include the following:

1. Name of item.
2. Description of purpose.
3. Source of input or destination of output.
4. Valid ranges of inputs and outputs.
5. Units of measure.
6. Data formats.

3.2. Functions

Define the actions that must take place in the software to accept and process inputs, and to generate outputs (ISO/IEC/IEEE 29148:2011). This should include:

1. Validity checks on the inputs.
2. Sequence of operations in processing inputs.
3. Responses to abnormal situations, including error handling and recovery.
4. Relationship of outputs to inputs, including
   (a) input/output sequences
   (b) formulas for input to output conversion

3.3. Usability Requirements

Define usability requirements and objectives for the software system, include measurable effectiveness, efficiency, and satisfaction criteria in specific contexts of use. (ISO/IEC/IEEE 29148:2011)

3.4. Performance Requirements

Specify the static and the dynamic numerical requirements placed on the software or on human interaction with the software. For example: (a) Static numerical requirements may include the amount and type of information to be handled. (b) Dynamic numerical requirements may include the amount of data to be processed within certain time periods.

Performance requirements should be stated in measurable terms. For example,
"95% of the transactions shall be processed in less than 1 second"

rather than

> “An operator shall not have to wait for the transaction to complete.”
> (ISO/IEC/IEEE 29148:2011)

3.5. Software System Attributes

Specify the required attributes of the software product, such as reliability, security, privacy, maintainability, or portability. (ISO/IEC/IEEE 29148:2011) Review a comprehensive list of software attributes, or software qualities, such as are provided in van Vliet (2008) Chapter 6. Decide on a relatively small number of attributes that are most important for this system. Explain why each attribute is important, and what steps or plan will be taken to achieve those attributes. This could include constraints on attributes of the system’s static construction, such as testability, changeability, maintainability, and reusability. (Faulk, 2013)

4. References

This section lists the sources cited in the creation of this template document. An SRS should reference all of the sources that it draws from. If sufficient citations are provided “in line” (at the point of reference) in the document, this section may not be necessary.

IEEE Std 1362-1998 (R2007). (2007). IEEE Guide for Information Technology–System Definition–Concept of Operations (ConOps) Document. https://ieeexplore.ieee.org/document/761853

IEEE Std 830-1998. (2007). IEEE Recommended Practice for Software Requirements Specifications. https://ieeexplore.ieee.org/document/720574

ISO/IEC/IEEE Intl Std 29148:2011. (2011). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/6146379

ISO/IEC/IEEE Intl Std 29148:2018. (2018). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/8559686

Faulk, Stuart. (2013). *Understanding Software Requirements*. https://projects.cecs.pdx.edu/attachments/download/904/Faulk_SoftwareRequirements_v4.pdf

Oracle. (2007). White Paper on “Getting Started With Use Case Modeling”. Available at: https://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf

van Vliet, Hans. (2008). *Software Engineering: Principles and Practice*, 3nd edition, John Wiley & Sons.

5. Acknowledgements

All sources used in the creation of the document should be listed here. This template builds slightly on a similar document produced by Stuart Faulk in 2017, and heavily on the publications cited within the document, such as IEEE Std 1362-1998 and ISO/IEC/IEEE Intl Std 29148:2018.

Possible Minor Future Revision

The ConOps portion of this template was written based on IEEE Std 1362-1998 before realizing that this standard has been subsumed by IEC/IEEE Intl Std 29148:2018. A future exercise could, in short, determine if the references to the former could be replaced with references to the latter; that is, if the ConOps description in each of the two documents is sufficiently identical.