Software Requirements Specification (SRS) Template

In the template, the parts in *italic* are parts that you are supposed to expand on. The parts in ***bold and italics*** are explanatory comments and are provided just for your understanding of the document.

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993).

Complete and tailor the document by expanding the relevant parts and removing explanatory comments as you go along. If you decide to omit a section, you might keep the header and insert a comment saying why you omitted the data.

(Team Number)

(Team Name)

(Team Logo)

Software Requirements Specification Document

Version: (n)Date: (mm/dd/yyyy)

Table of Contents

1. Introduction

The following subsections of the Software Requirements Specifications (SRS) document should provide an overview of the entire SRS. The thing to keep in mind as you write this document is that you are telling what the system must do, so that designers can ultimately build it. Do not use this document for design.

1.1 Purpose

In this subsection, describe the purpose of the particular SRS and specify the intended audience for the SRS.

1.2 Scope

In this subsection:

(1) *Identify the software product(s) to be produced by name*
(2) *Explain what the software product(s) will, and, if necessary, will not do*
(3) *Describe the application of the software being specified, including relevant benefits, objectives, and goals*
(4) *Be consistent with similar statements in higher-level specifications if they exist*

This should be an executive-level summary. Do not enumerate the whole requirements list here.

1.3 Definitions, Acronyms, and Abbreviations

Provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS.

1.4 References

Note any references or related materials here. For instance, if your application uses specific protocols or RFC's, then reference them here so designers can find them.
In this section:

(1) *Provide a complete list of all documents referenced elsewhere in the SRS*
(2) *Identify each document by title, report number (if applicable), date, and publishing organization*
(3) *Specify the sources from which the references can be obtained*

This information can be provided by reference to an appendix or to another document.

1.5 Overview

In this subsection:

(1) *Describe what the rest of the SRS contains*
(2) *Explain how the SRS is organized*

Don't rehash the table of contents here. Point people to the parts of the document they are most concerned with. Customers and potential users should care about Section 2, whereas developers should care about Section 3.

2. The Overall Description

The following subsections describe the general factors that affect the product and its requirements. They do not state specific requirements, but rather provide background for those requirements, which are defined in Section 3. Roughly speaking, this section states the requirements in plain English, for the customer, whereas Section 3 provides a specification written for the developers.

2.1 Product Perspective

Put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection relates the requirements of the larger system to functionality of the software and identifies interfaces between that system and the software.

If you are building a real system, compare its similarity and differences to other systems in the marketplace.

A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful. This diagram should not be a design/architecture picture. It should just provide context if your system will interact with external actors. The system you are building should be shown as a black box. Let the design document present the internals.

2.1.1 Software Interfaces

Specify the use of other required software products. These are external systems/libraries that you have to interact with.

For each required software product, include (if applicable):

(1) *Name*
(2) *Mnemonic*
(3) *Version number*

(4) *Source*
(5) *Define what kind of data is going to be exchanged with the external software*

Here we document the parts of the software that we do not have to write, but that our system has to use. For instance, if your customer uses SQL Server 7 and you are required to use it, then you need to specify that. For example, you may state something like:

Microsoft SQL Server 7
SQL Server
Version 7.2
The system must use SQL Server as its database component. Communication with the DB is through ODBC connections. The system must provide SQL data table definintions to be supplied to the company DBA for setup.

2.1.2 User Interfaces

Specify the logical characteristics of each interface between the software product and its users. This is a description of how the system will interact with its users. Is there a GUI, a command line, or some other type of interface? Are there special interface requirements?

2.1.3 Communications Interfaces

Specify the various interfaces to communications such as local network protocols, etc. These are protocols you will need to directly interact with. If you happen to use web services transparently to your application then do not list it here. If you are using a custom protocol to communicate between systems, then document that protocol here, so designers know what to design. If it is a standard protocol, you can reference an existing document or RFC.

2.1.4 Memory Constraints

Specify any applicable characteristics and limits on primary and secondary memory. Don't just make up something here. If all the customer's machines have only 128K of RAM, then your target must be under 128K, so there is an actual requirement. You could also cite market research here for shrink-wrap type applications "Focus groups have determined that our target market has between 256-512M of RAM, therefore the design footprint should not exceed 256M." If there are no memory constraints, just say so.

2.2 Product Functions

This is the real meat of Section 2. It should provide, in the language of the user, a high-level description of the functionality of the system in terms of the major functions that the software will perform.

For clarity:

(1) *The functions should be organized in a way that makes the list of functions understandable to the customer or to anyone else reading the document for the first time.*

(2) *Textual or graphic methods can be used to show the different functions and their relationships. Such a diagram is not intended to show a design of a product but simply shows the logical relationships among variables.*

2.3 User Characteristics

Describe those general characteristics of the intended users of the product, including educational level, experience, and technical expertise. What is it about your potential user base that will impact the design? Their experience and comfort with technology may drive UI design. Other characteristics might actually influence internal design of the system.

2.4 Constraints

Provide a general description of any other items that will limit the developer's options. This section captures non-functional requirements in the customers language. These can include:

(1) Regulatory policies
(2) Hardware limitations (for example, signal timing requirements)
(3) Interface to other applications
(4) Parallel operation
(5) Audit functions
(6) Control functions
(7) Higher-order language requirements
(8)*Signal handshake protocols (for example, XON-XOFF, ACK-NACK)*
(414482448)*Reliability requirements*
(10) Criticality of the application
(11) Safety and security considerations

2.5 Assumptions and Dependencies

List each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption might be that a specific operating system would be available on the hardware designated for the software product. If, in fact, the operating system were not available, the SRS would then have to change accordingly.

This section is catch-all for everything else that might influence the design of the system and that did not fit in any of the categories above.

2.6 Apportioning of Requirements.

Identify requirements that may be delayed until future versions of the system. After you look at the project plan and hours available, you may realize that you just cannot get everything done. This section divides the requirements into different sections for development and delivery. Remember to check with the customer – they should prioritize the requirements and decide what does and does not get done. This can also be useful if you are using an iterative life cycle model to specify which requirements will map to which interation.

3. Specific Requirements

These subsections contain all the software requirements at 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. The following principles apply:

(1) ***Specific requirements should be stated with all the characteristics of a good SRS***
- ***correct***
- ***clear (unambiguous)***
- ***complete***
- ***consistent***
- ***coherent***
- ***verifiable***
- ***modifiable***
- ***traceable***

(2) ***All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)***

(3) ***Careful attention should be given to organizing the requirements to maximize readability. (Several alternative organizations are discussed in the appendix at the end of this document.)***

Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in the following

subclasses. This section reiterates Section 2, but is for developers not the customer. The customer buys in with Section 2, the designers use Section 3 to design and build the actual application.

Use proper terminology:

The system shall…	***A required, must have feature***
The system should…	***A desired feature, but may be deferred til later***
The system may…	***An optional, nice-to-have feature that may never make it to implementation.***

Avoid imprecise statements like, "The system shall be easy to use." What does that mean? Avoid "motherhood and apple pie" type statements, such as "The system shall be developed using good software engineering practice"

Avoid examples, This is a specification, a designer should be able to read this spec and build the system without bothering the customer again. Don't say things like, "The system shall accept configuration information such as name and address." The designer doesn't know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right system.

3.1 External Interfaces

This section contains a detailed description of all inputs and outputs of the software system as a whole. It should discuss both content and format as follows:

- *Name of item*
- *Description of purpose*
- *Source of input or destination of output*
- *Valid range, accuracy and/or tolerance*
- *Units of measure*
- *Timing*
- *Relationships to other inputs/outputs*
- *Screen formats/organization*
- *Window formats/organization*
- *Data formats*
- *Command formats*
- *End messages*

3.2 Functions

Functional requirements 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 should include:

- *Validity checks on the inputs*
- *Exact sequence of operations*
- *Responses to abnormal situation, including*
    - *Overflow*
    - *Communication facilities*
    - *Error handling and recovery*
- *Effect of parameters*
- *Relationship of outputs to inputs, including*
    - *Input/Output sequences*
    - *Formulas for input to output conversion*

It may be appropriate to partition the functional requirements into sub-functions or sub-processes. This does not imply that the software design will also be partitioned that way.

3.3 Performance Requirements

This subsection specifies both the static and the dynamic numerical requirements placed on the software or on human interaction with the software, as a whole.

Static numerical requirements may include:

(a) The number of terminals to be supported
(b) The number of simultaneous users to be supported
(c) Amount and type of information to be handled

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these 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.

(Note: Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.)

3.4 Logical Data Requirements

This section specifies the logical requirements for any data that is to be handled by the system. This may include:

- *Types of information used by various functions*
- *Frequency of use*
- *Accessing capabilities*
- *Data entities and their relationships*
- *Integrity constraints*
- *Data retention requirements*

If the customer provided you with data models, those can be presented here. ER diagrams (or static class diagrams) can be useful here to show complex data relationships. Remember that often a diagram is worth a thousand words.

3.5 Software System Attributes

There are attributes of the software that can serve as "cross-cutting" requirements. They are also known as non-functional requirements or quality attributes. These requirements have to be testable just like the functional requirements (it is easy to be fuzzy here, but keep it specific). Identify the quality attributes that are important for your system and specify them here. Write a subsection for each relevant attribute, explaining why the attribute is important for your system and how it will be tested and measured. The following subsections provide some examples.

3.5.1 Reliability

Specify the factors required to establish the required reliability of the software system at time of delivery. If you have Mean Time Between Failures (MTBF) requirements, express them here. This does not refer to just having a program that does not crash; this has a specific engineering meaning.

3.5.2 Availability

Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. This is somewhat related to reliability. Some systems run only infrequently and on-demand (e.g., a word processor). Some systems have to run 24/7 (e.g., an e-commerce web site). The required availability will greatly impact the design.

Example: What are the requirements for system recovery from a failure? "The system shall allow users to restart the application after failure with the loss of at most 12 characters of input".

3.5.3 Security

Specify the factors that would protect the software from accidental or malicious access, *use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:*

- *Use certain cryptographic techniques*
- *Keep specific log or history data sets*
- *Assign certain functions to different modules*
- *Restrict communications between some areas of the program*
- *Check data integrity for critical variables*

3.5.4 Maintainability

Specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices.

3.5.5 Portability

Specify attributes of software that relate to the ease of porting the software to other host machines or operating systems. These may include:

- *Percentage of components with host-dependent code*
- *Percentage of code that is host dependent*
- *Use of a proven portable language*
- *Use of a particular compiler or language subset*
- *Use of a particular operating system*

3.5.6 Other quality characteristics

There are many other quality attributes that may be relevant for your system and are not discussed in the paragraphs above:

- *Correctness - extent to which program satisfies specifications, fulfills user's mission objectives*
- *Efficiency - amount of computing resources and code required to perform function*
- *Flexibility - effort needed to modify operational program*
- *Interoperability - effort needed to couple one system with another*
- *Reusability - extent to which it can be reused in another application*
- *Testability - effort needed to test to ensure performs as intended*
- *Usability - effort required to learn, operate, prepare input, and interpret output*

4. Change Management Process

Identify the change management process to be used to identify, log, evaluate, and update the SRS to reflect changes in project scope and requirements. How are you going to *control changes to the requirements. Can the customer just call up and ask for something new? Does your team have to reach consensus? How do changes to requirements get submitted to the team? Formally in writing, email or phone call?*

5. Supporting Information

The supporting information makes the SRS easier to use and can be included in the form of appendices to the document. Note that appendices are not always necessary.

Appendices may include:

- *Sample I/O formats*
- *Descriptions of cost analysis studies*
- *Results of user surveys*
- *Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements*
- *Raw data captured during requirement gathering*
- *Other supporting/background information that can help the readers of the SRS*
- *…*

Appendix: Organizing the requirements
(use one of the proposed organizations in preparing Section 3)

There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements. Some possible organizations are described below and detailed in the following pages.

Choose the one that best fits your system. If more than one organization is appropriate, you can organize the specific requirements aling multiple hierarchies.

- System Mode. **Some systems behave quite differently depending on the mode of operation.**
- User Class. ***Some systems provide different sets of functions to different classes of users.***
- Objects. **Objects are real-world entities that have a counterpart within the system. Associated with each object is a set of attributes and functions. These functions are also called services, methods, or processes. Note that sets of objects may share attributes and services. These are grouped together as classes.**
- Feature. **A feature is an externally visible service provided by the system that may require a sequence of inputs to effect the desired result. Each feature is generally described as sequence eof stimulus-response pairs.**
- Stimulus. **Some systems can be best organized by describing their functions in terms of stimuli.**
- Response. **Some systems can be best organized by describing their functions in support of the generation of a response.**
- Functional Hierarchy. ***When none of he above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by either common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be used to show the relationships between and among the functions and data.***

The following pages provide possible outlines for the different organizations described above. In the outlines, those sections called "Functional Requirement i" may be described in native language, in pseudocode, in a system definition language, or in four subsections titled: Introduction, Inputs, Processing, Outputs.

Outline for SRS Section 3
Organized by mode: Version 1

3. Specific Requirements
   - 3.1 External interface requirements
     - 3.1.1 User interfaces
     - 3.1.2 Hardware interfaces
     - 3.1.3 Software interfaces
     - 3.1.4 Communications interfaces
   - 3.2 Functional requirements
     - 3.2.1 Mode 1
       - 3.2.1.1 Functional requirement 1.1
       - .....
       - 3.2.1.*n* Functional requirement 1.*n*
     - 3.2.2 Mode 2
       - .....
     - 3.2.*m* Mode *m*
       - 3.2.*m*.1 Functional requirement *m*.1
       - .....
       - 3.2.*m.n* Functional requirement *m.n*
   - 3.3 Performance Requirements
   - 3.4 Design Constraints
   - 3.5 Software system attributes
   - 3.6 Other requirements

Outline for SRS Section 3
Organized by mode: Version 2

3. Specific Requirements
  3.1 Functional Requirements
    3.1.1 Mode 1
      3.1.1.1 External interfaces
          3.1.1.1 User interfaces
          3.1.1.2 Hardware interfaces
          3.1.1.3 Software interfaces
          3.1.1.4 Communications interfaces
      3.1.1.2 Functional Requirement
        3.1.1.2.1 Functional requirement 1
        .....
        3.1.1.2.*n* Functional requirement *n*
      3.1.1.3 Performance
    3.1.2 Mode 2
        .....
    3.1.*m* Mode *m*
  3.2 Design constraints
  3.414482688 Software system attributes
  3.414482689 Other requirements

Outline for SRS Section 3
Organized by user class
(e.g., system adminstrators, managers, clerks, etc.)

3. Specific Requirements
    - 3.1 External interface requirements
        - 3.1.1 User interfaces
        - 3.1.2 Hardware interfaces
        - 3.1.3 Software interfaces
        - 3.1.4 Communications interfaces
    - 3.2 Functional requirements
        - 3.2.1 User class 1
            - 3.2.1.1 Functional requirement 1.1
            - .....
            - 3.2.1.*n* Functional requirement 1.*n*
        - 3.2.2 User class 2
            - .....
        - 3.2.*m* User class *m*
            - 3.2.*m*.1 Functional requirement *m*.1
            - .....
            - 3.2.*m.n* Functional requirement *m.n*
    - 3.3 Performance Requirements
    - 3.4 Design Constraints
    - 3.5 Software system attributes
    - 3.6 Other requirements

Outline for SRS Section 3
Organized by object
(good if you did an object-oriented analysis as part of your requirements)

3 Specific Requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Classes/Objects
  - 3.2.1 Class/Object 1
    - 3.2.1.1 Attributes (direct or inherited)
      - 3.2.1.1.1 Attribute 1
      - .....
      - 3.2.1.1.*n* Attribute *n*
    - 3.2.1.2 Functions (services, methods, direct or inherited)
      - 3.2.1.2.1 Functional requirement 1.1
      - .....
      - 3.2.1.2.*m* Functional requirement 1.*m*
    - 3.2.1.3 Messages (communications received or sent)
  - 3.2.2 Class/Object 2
  - .....
  - 3.2.*p* Class/Object *p*
- 3.3 Performance Requirements
- 3.4 Design Constraints
- 3.5 Software system attributes
- 3.6 Other requirements

Outline for SRS Section 3
Organized by feature
(good when there are clearly delimited feature sets)

3 Specific Requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 System features
  - 3.2.1 System Feature 1
    - 3.2.1.1 Introduction/Purpose of feature
    - 3.2.1.2 Stimulus/Response sequence
      - 3.2.1.3 Associated functional requirements
        - 3.2.1.3.1 Functional requirement 1
        - .....
        - 3.2.1.3.*n* Functional requirement *n*
  - 3.2.2 System Feature 2
    - .....
  - 3.2.*m* System Feature *m*
    - .....
- 3.3 Performance Requirements
- 3.4 Design Constraints
- 3.5 Software system attributes
- 3.6 Other requirements

Outline for SRS Section 3
Organized by stimulus
(good for event driven systems where the events form logical groupings)

3 Specific Requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Functional requirements
  - 3.2.1 Stimulus 1
    - 3.2.1.1 Functional requirement 1.1
    - .....
    - 3.2.1.*n* Functional requirement 1.*n*
  - 3.2.2 Stimulus 2
  - .....
  - 3.2.*m* Stimulus *m*
    - 3.2.*m*.1 Functional requirement *m*.1
    - .....
    - 3.2.*m.n* Functional requirement *m.n*
- 3.3 Performance Requirements
- 3.4 Design Constraints
- 3.5 Software system attributes
- 3.6 Other requirements

Outline for SRS Section 3
Organized by response
(good for event-driven systems where the responses form logical groupings)

3 Specific Requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Functional requirements
  - 3.2.1 Response 1
    - 3.2.1.1 Functional requirement 1.1
    - .....
    - 3.2.1.*n* Functional requirement 1.*n*
  - 3.2.2 Response 2
  - .....
  - 3.2.*m* Response *m*
    - 3.2.*m*.1 Functional requirement *m*.1
    - .....
    - 3.2.*m.n* Functional requirement *m.n*
- 3.3 Performance Requirements
- 3.4 Design Constraints
- 3.5 Software system attributes
- 3.6 Other requirements

Outline for SRS Section 3
Organized by functional hierarchy
(good if you have done structured analysis as part of your design)

3 Specific Requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Functional requirements
- 3.2.1 Information flows
  - 3.2.1.1 Data flow diagram 1
    - 3.2.1.1.1 Data entities
    - 3.2.1.1.2 Pertinent processes
    - 3.2.1.1.3 Topology
  - 3.2.1.2 Data flow diagram 2
    - 3.2.1.2.1 Data entities
    - 3.2.1.2.2 Pertinent processes
    - 3.2.1.2.3 Topology
    - .....
  - 3.2.1.*n* Data flow diagram *n*
    - 3.2.1.*n*.1 Data entities
    - 3.2.1.*n*.2 Pertinent processes
    - 3.2.1.*n*.3 Topology
  - 3.2.2 Process descriptions
    - 3.2.2.1 Process 1
      - 3.2.2.1.1 Input data entities
      - 3.2.2.1.2 Algorithm or formula of process
      - 3.2.2.1.3 Affected data entities
    - 3.2.2.2 Process 2
      - 3.2.2.2.1 Input data entities
      - 3.2.2.2.2 Algorithm or formula of process
      - 3.2.2.2.3 Affected data entities
      - …..
    - 3.2.2.*m* Process *m*
      - 3.2.2.*m*.1 Input data entities
      - 3.2.2.*m*.2 Algorithm or formula of process
      - 3.2.2.*m*.3 Affected data entities
  - 3.2.3 Data construct specifications
    - 3.2.3.1 Construct 1
      - 3.2.3.1.1 Record type
      - 3.2.3.1.2 Constituent fields
    - 3.2.3.2 Construct 2
      - 3.2.3.2.1 Record type

3.2.3.2.2 Constituent fields
…..
3.2.3.*p* Construct *p*
3.2.3.*p*.1 Record type
3.2.3.*p*.2 Constituent fields
3.2.4 Data dictionary
3.2.4.1 Data element 1
3.2.4.1.1 Name
3.2.4.1.2 Representation
3.2.4.1.3 Units/Format
3.2.4.1.4 Precision/Accuracy
3.2.4.1.5 Range
3.2.4.2 Data element 2
3.2.4.2.1 Name
3.2.4.2.2 Representation
3.2.4.2.3 Units/Format
3.2.4.2.4 Precision/Accuracy
3.2.4.2.5 Range
…..
3.2.4.*q* Data element *q*
3.2.4.*q*.1 Name
3.2.4.*q*.2 Representation
3.2.4.*q*.3 Units/Format
3.2.4.*q*.4 Precision/Accuracy
3.2.4.*q*.5 Range
3.3 Performance Requirements
3.4 Design Constraints
3.5 Software system attributes
3.6 Other requirements

Outline for SRS Section 3
Showing multiple organizations
(can't decide? glob it all together)

3 Specific Requirements
- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 Functional requirements
  - 3.2.1 User class 1
    - 3.2.1.1 Feature 1.1
      - 3.2.1.1.1 Introduction/Purpose of feature
      - 3.2.1.1.2 Stimulus/Response sequence
      - 3.2.1.1.3 Associated functional requirements
    - 3.2.1.2 Feature 1.2
      - 3.2.1.2.1 Introduction/Purpose of feature
      - 3.2.1.2.2 Stimulus/Response sequence
      - 3.2.1.2.3 Associated functional requirements
      - …..
    - 3.2.1.*m* Feature 1.*m*
      - 3.2.1.*m*.1 Introduction/Purpose of feature
      - 3.2.1.*m*.2 Stimulus/Response sequence
      - 3.2.1.*m*.3 Associated functional requirements
  - 3.2.2 User class 2
  - .....
  - 3.2.*n* User class *n*
  - .....
- 3.3 Performance Requirements
- 3.4 Design Constraints
- 3.5 Software system attributes
- 3.6 Other requirements

Outline for SRS Section 3
Organized by Use Case
(good when following UML development)

3. Specific Requirements
   - 3.1 External Actor Descriptions
     - 3.1.1 Human Actors
     - 3.1.2 Hardware Actors
     - 3.1.3 Software System Actors
   - 3.2 Use Case Descriptions
     - 3.2.1 Use Case 1
     - 3.2.2 Use Case 2

     - 3.2.n Use Case n
   - 3.3 Performance Requirements
   - 3.4 Design Constraints
   - 3.5 Software system attributes
   - 3.6 Other requirements