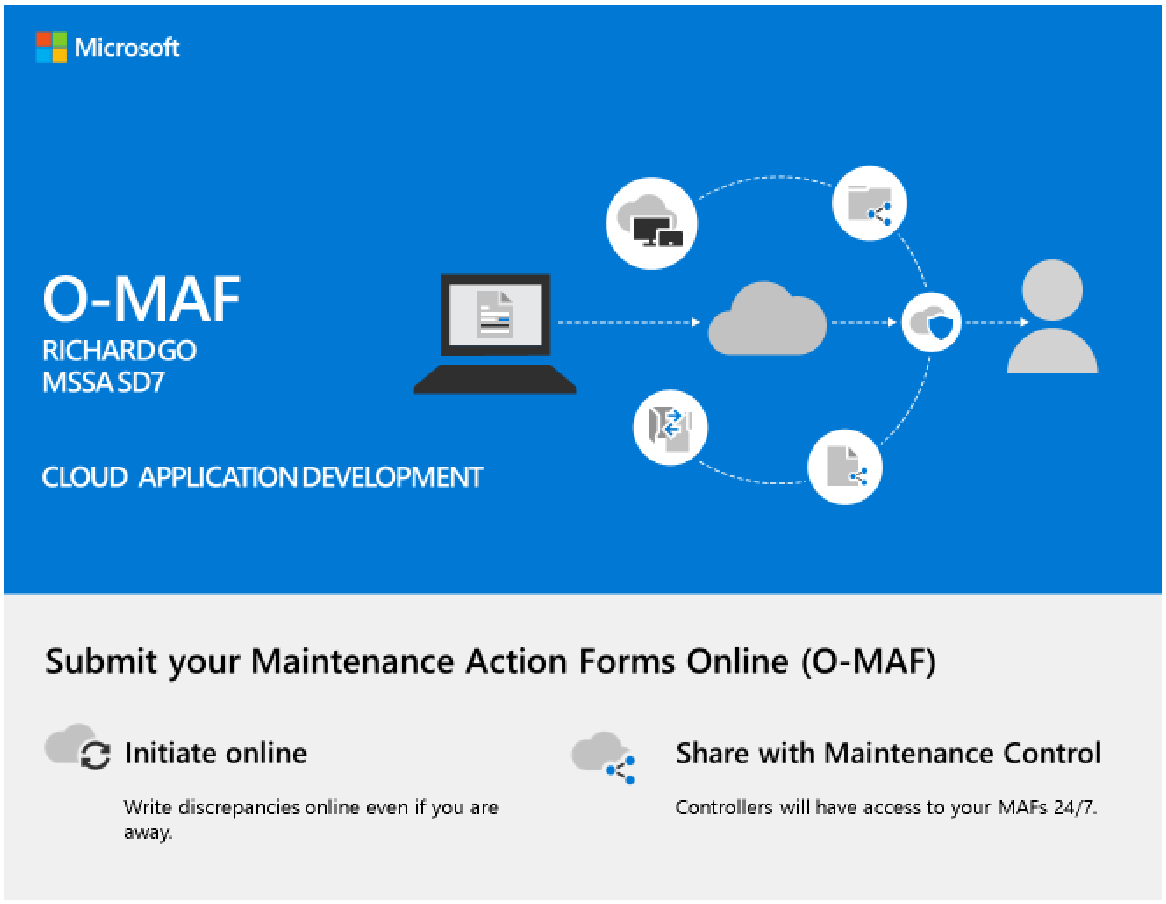
CS330 Software Engineering

Software Requirements Specification (SRS) Template

Items that are intended to stay in as part of your document are in **bold**; explanatory comments are in *italic* text. Plain text is used where you might insert wording about your project.

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).

Tailor this to your needs, removing explanatory comments as you go along. Where you decide to omit a section, keep the header, but insert a comment saying why you omit the data.


Microsoft
O-MAF
RICHARD GO
MSSA SD7
CLOUD APPLICATION DEVELOPMENT
Submit your Maintenance Action Forms Online (O-MAF)
Initiate online
Write discrepancies online even if you are away.
Share with Maintenance Control
Controllers will have access to your MAFs 24/7.

Software Requirements Specification Document

Version: (1) **Date: (27 Feb 2020)**

Table of Contents

1. Introduction

Online Maintenance Action Form (O-MAF) will be used to run the maintenance in case the main system goes down. The issue underlying aviation maintenance is the use of computer-based software that we call Optimized Organizational Maintenance Activity (OOMA).

The problem is whenever the system is down, we are forced to use a paper copy of all our MAF. There are just too many things to update while the maintenance actions are open and being worked on.

(1) Ordering parts
(2) Personnel going In-Work
(3) Completing the job

After the system comes back online, the controllers have to contingency all the paper MAF and type all open and closed MAFs to OOMA. These process is just too time consuming for the workers and for the controllers.

1.1 Purpose

Operate and maintain the O-MAF automated online tools to provide maintenance, material and operations managers with timely, accurate and complete information on which to base daily decisions in the management of assigned aircraft and equipment.

1.2 Scope

O-MAF solves the issue of manually managing workloads and personnel. Study have shown that by Using O-MAF will enable the controllers to effectively manage all MAFs and set priorities. Efficiency will be increased by 67% based on previous studies done by Certified Lean Six Sigma Green Belt.

Future concept will enable controllers to automatically upload all data from O-MAF to OOMA if permitted by the Navy system administrators.

1.3 Definitions, Acronyms, and Abbreviations.

(1) ASM. Training Management System used by Marine Aviation.
(2) CAC. Common Access Card used by military and government contrators to access sensitive FOUO materials from the web or any portal.
(3) CDI. Collateral Duty Inspector.
(4) FOUO. For Official Use Only.
(5) InPro. In Process. CDI acknowledging his inspection and documenting it.
(6) IW. In Work. Job that is currently being worked on.

(7) JC. Job Complete
(8) QA. Quality Assurance.
(9) M3. Awaiting Maintenance
(10) MC. Maintenance Control. Manages all maintenance actions needed to be attended.
(11) MDS. Maintenance Data Specialist. Maintenance Administrator.
(12) MSSM. Microsoft SQL Server Management
(13) O-MAF. Online Mainatenance Action Forms. This is the job ticket.
(14) PN. Part Number
(15) SN. Serial Number
(16) SAR. System Access Request form is used to check eligibility of users on accessing government computers, systems, and portals.
(17) SFF. Safe For Fight
(18) WP. Awaiting parts needed to finish the job.

1.4 References

O-MAf system was inspired by the NALCOMIS OOMA System Administration being used by Marine Tiltrotor Squadron 166 located in San Diego, Ca.

(1) Marine Tiltrotor Squadron 166 Maintenance Control section
(2) https://www.public.navy.mil/netc/centers/cnatt/atsugi/Course-Description.aspx?CDP=7784

1.5 Overview

The concept comprise of many steps in chronological and systematic order or workflow.

(1) Initiate maintenance
(2) Order parts
(3) In Work
(4) In Process
(5) Update Job Status – must be filtered for management (M3, IW, WP, JC)
(6) Complete maintenance
(7) PN/SN removed/installed
(8) Completed time
(9) Sign authority
    a. Initiator / Pilot
    b. Worker
    c. CDI always required
    d. QA if required
    e. MC always required
    f. MDS always required

2. The Process

The system requires daily back up of all data needed to run O-MAF. This daily back up will not add to the efficiency of managing aircraft workloads. This data needs manipulation by O-MAF to run and be used efficiently in a dynamic online platform. This platform will produce data needed for timely, accurate and complete information on which to base daily decisions in the management of assigned aircraft and equipment

(1) Daily back up copies of the data.
  - CSV file downloaded from OOMA
  - After every shift

(2) Input
  - Purged data on O-MAF
  - Upload data to O-MAF

(3) Sorting/Filtering

(4) Sort OPEN MAF by JCN column

(5) Sort OPEN MAF by MCN column
  - Sort OPEN MAF by Shop
  - Sort OPEN MAF by Aircraft
  - Filter Job Status column

(6) Complete and Close MAF.
  - Authorized signer by qualifications

2.1 Product Perspective

O-MAF will mirror some of the functionality of OOMA. It will not update the main servers automatically. Manual input is still needed to update the main servers.

The spelled diragram will show you how the systems are connected.

(1) Data Model
  a. OOMA for creating back up data
  b. Excel for saving .csv file
  c. MSSM using SQL

(2) Methods / Functions / Controllers
  a. Visual Studio using C#

(3) View
  a. Browser

(4) Storage
  a. SharePoint
  b. SQL Server on shareddrive

The following subsections describe how the software operates inside various constraints.

2.1.1 System Interfaces

O-MAF will interface with Microsoft Excel CSV file. The .csv file will be uploaded to the SQL Server database to purge the old data and replaced with the new data coming from the daily backup.

2.1.2 Interfaces

O-MAF will interact with the following data:

(1) Daily back up of data by the the contrllers as system admin.
(2) Purging of old data from O-MAF SQL Server.
(3) Uploading the new data .csv file to O-MAF SQL Server that replenish the data with new and most updated data.

2.1.3 Hardware Interfaces

O-MAF will work in any device platform that uses a web browser.

(1) Mobile phones
(2) Desktop computers
(3) Tablet
(4) SQL Server

2.1.4 Software Interfaces

O-MAf will work with any web browser and it will adapt to the user's screen orientation.

(1) Microsoft Edge
(2) Google Chrome
(3) Mozilla Firefox

All of the used codes and software will be stored online in a secured webhost wun by the unit's system administrator.

(1) MSSM
(2) Visual Studion
(3) SharePoint

2.1.5 Communications Interfaces

Users must have access to the following:

(1) Users must be able to login to SharePoint using their issued CAC card.
(2) Users must have access to the portal.
    a. Users must submit request using a SAR form. to gain access.
(3) Users must have proper credential for different levels of qualification.

(4) Users must have an account created with the following:
   a. OOMA
   b. O-MAF
   c. SharePoint site
   d. Shareddrive

<u>**2.1.6 Memory Constraints**</u>

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 design has got to come in 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, so state.

<u>**2.1.7 Operations**</u>

Specify the normal and special operations required by the user such as:
(1) The various modes of operations in the user organization
(2) Periods of interactive operations and periods of unattended operations
(3) Data processing support functions
(4) Backup and recovery operations

(Note: This is sometimes specified as part of the User Interfaces section.) If you separate this from the UI stuff earlier, then cover business process type stuff that would impact the design. For instance, if the company brings all their systems down at midnight for data backup that might impact the design. These are all the work tasks that impact the design of an application, but which might not be located in software.

<u>**2.1.8 Site Adaptation Requirements**</u>

In this section:
(1) Define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode
(2) Specify the site or mission-related features that should be modified to adapt the software to a particular installation

If any modifications to the customer's work area would be required by your system, then document that here. For instance, "A 100Kw backup generator and 10000 BTU air conditioning system must be installed at the user site prior to software installation". This could also be software-specific like, "New data tables created for this system must be installed on the company's existing DB server and populated prior to system activation." Any equipment the customer would need to buy or any software setup that *needs to be done so that your system will install and operate correctly should be documented here.*

2.2 Product Functions

Provide a summary of the major functions that the software will perform. Sometimes the function summary that is necessary for this part can be taken directly from the section of the higher-level specification (if one exists) that allocates particular functions to the software product.

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.

AH, Finally the real meat of section 2. This describes the functionality of the system in the language of the customer. What specifically does the system that will be designed have to do? Drawings are good, but remember this is a description of what the system needs to do, not how you are going to build it. (That comes in the design document).

2.3 User Characteristics

Describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise. Do not state specific requirements but rather provide the reasons why certain specific requirements are later specified in section 3.

What is it about your potential user base that will impact the design? Their experience and comfort with technology will 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. 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)*
(9) *Reliability requirements*
(10) Criticality of the application
(11) Safety and security considerations

This section captures non-functional requirements in the customers language. A more formal presentation of these will occur in section 3.

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

Users must have access to the requirements stated in paragraph 2.1.5.

The following are requirements per level of qualifications.

(1) Initiator / Pilot
    a. No qualification required to initiate and work on a job.
(2) CDI
    a. Must have his/her CDI qualification to inspect and enter In-Processes. Must be verifiable using the following:
        i. OOMA
        ii. ASM
        iii. QA

(3) QA
   a. Must have his/her CDI qualification to inspect and enter In-Processes. Must be verifiable using the following:
      i. OOMA
      ii. ASM
      iii. QA

(4) MC SFF
   a. Must have his/her MC qualification to approve MAF that are initiated, inspected, and completed for completion. SFF qualification is required for screening an aircraft safe for flight. Must be verifiable using the following:
      i. OOMA
      ii. ASM
      iii. QA

This section contains 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*
- *unambiguous*
- *complete*
- *consistent*
- *ranked for importance and/or stability*
- *verifiable*
- *modifiable*
- *traceable*

(2) Specific requirements should be cross-referenced to earlier documents that relate
(3) All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)
(4) Careful attention should be given to organizing the requirements to maximize readability (Several alternative organizations are given at end of 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.

Remember this is not design. Do not require specific software packages, etc unless the customer specifically requires them. Avoid over-constraining your design. 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.

Each requirement should be uniquely identified for traceability. Usually, they are numbered 3.1, 3.1.1, 3.1.2.1 etc. Each requirement should also be testable. Avoid imprecise statements like, "The system shall be easy to use" Well no kidding, what does that mean? Avoid "motherhood and apple pie" type statements, "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 UI and data tables.

3.1 External Interfaces

This contains a detailed description of all inputs into and outputs from the software system. It complements the interface descriptions in section 2 but does not repeat information there. Remember section 2 presents information oriented to the customer/user while section 3 is oriented to the developer.

It contains 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 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

Static numerical requirements are sometimes identified under a separate section entitled capacity.

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 Database Requirements

This section specifies the logical requirements for any information that is to be placed into a database. 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 a diagram is worth a thousand words of confusing text.

3.5 Design Constraints

Specify design constraints that can be imposed by other standards, hardware limitations, etc.

3.5.1 Standards Compliance

Specify the requirements derived from existing standards or regulations. They might include:
(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.

3.6 Software System Attributes

There are a number of attributes of software that can serve as requirements. It is important that required attributes by specified so that their achievement can be objectively verified. The following items provide a partial list of examples. These are also known as non-functional requirements or quality attributes.

These are characteristics the system must possess, but that pervade (or cross-cut) the design. These requirements have to be testable just like the functional requirements. Its easy to start philosophizing here, but keep it specific.

3.6.1 Reliability

Specify the factors required to establish the required reliability of the software system at time of delivery. If you have MTBF requirements, express them here. This doesn't refer to just having a program that does not crash. This has a specific engineering meaning.

3.6.2 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 on-demand (like MS Word). Some systems have to run 24/7 (like an e-commerce web site). The required availability will greatly impact the design. 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".

3.6.3 Security

Access to the data are given to limited system admin only. One or two MC will have access to the data as well. Care must be given n accessing the back-end administration to limit accidental deletions of data.The system will utilize the following servers for secured access.

(1) Sharepoint is only accessible to an approved users.
(2) Shareddrive is only accessible to an approved users.
(3) O-MAF also requires users to create an account for access.

3.6.4 Maintainability

The MC and MDS system admin will maintain the site for purging and populating data to the system.

3.6.5 Portability

O-MAF will be dependent to SharePoint at the current version. Future improvements will allow users to access O-MAF using their mobile device. Future improvements can be purchased by the custodian unit maintaining aircraft.

- 100% reliability in SharePoint
- 80% reliability in Shareddrive

Once the relevant characteristics are selected, a subsection should be written for each, explaining the rationale for including this characteristic and how it will be tested and measured. A chart like this might be used to identify the key characteristics (rating them High or Medium), then identifying which are preferred when trading off design or implementation decisions (with the ID of the preferred one indicated in the chart to the right). The chart below is optional (it can be confusing) and is for demonstrating tradeoff analysis between different non-functional requirements. H/M/L is the relative priority of that non-functional requirement.

ID	**Characteristic**	**H/M/L**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	Correctness													
2	Efficiency													
3	Flexibility													
4	Integrity/Security													
5	Interoperability													
6	Maintainability													
7	Portability													
8	Reliability													
9	Reusability													
10	Testability													
11	Usability													
12	Availability													

Definitions of the quality characteristics not defined in the paragraphs above follow.

- *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*
- *Reliability - extent to which program performs with required precision*
- *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*

THE FOLLOWING (3.7) is not really a section, it is talking about how to organize requirements you write in section 3.2. At the end of this template there are a bunch of *alternative organizations for section 3.2. Choose the ONE best for the system you are writing the requirements for.*

3.7 Organizing the Specific Requirements

For anything but trivial systems the detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a manner optimal for understanding. There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements in section 3. Some of these organizations are described in the following subclasses.

3.7.1 System Mode

Some systems behave quite differently depending on the mode of operation. When organizing by mode there are two possible outlines. The choice depends on whether interfaces and performance are dependent on mode.

3.7.2 User Class

Some systems provide different sets of functions to different classes of users.

3.7.3 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.

3.7.4 Feature

A feature is an externally desired service by the system that may require a sequence of inputs to effect the desired result. Each feature is generally described in as sequence eof stimulus-response pairs.

3.7.5 Stimulus

Some systems can be best organized by describing their functions in terms of stimuli.

3. 7.6 Response

Some systems can be best organized by describing their functions in support of the generation of a response.

3.7.7 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 use dot show the relationships between and among the functions and data.

3.8 Additional Comments

Whenever a new SRS is contemplated, more than one of the organizational techniques given in 3.7 may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification.

Three 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; when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.

In any of the outlines below, 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.

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. Document Approvals

Identify the approvers of the SRS document. Approver name, signature, and date should be used.

6. Supporting Information

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

- *Table of Contents*
- *Index*
- *Appendices*

The Appendices are not always considered part of the actual requirements specification and are not always necessary. They may include:

(a) Sample I/O formats, descriptions of cost analysis studies, results of user surveys
(b) Supporting or background information that can help the readers of the SRS
(c) A description of the problems to be solved by the software
(d) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements

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

Tables on the following pages provide alternate ways to structure section 3 on the specific requirements. You should pick the best one of these to organize section 3 requirements.

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
   3.3 Software system attributes
   3.4 Other requirements

Outline for SRS Section 3
Organized by user class (i.e. different types of users ->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? Then 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