SocketTest

**Software Requirements Specification**

**Document**

**Version: 0.1** **Date: 08/23/2017**

**Table of Contents**

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations. 4

1.4 References 4

1.5 Overview 4

2. The Overall Description 5

2.1 Product Perspective 5

2.1.1 System Interfaces 5

2.1.2 Interfaces 5

2.1.3 Hardware Interfaces 6

2.1.4 Software Interfaces 6

2.1.5 Communications Interfaces 7

2.1.6 Memory Constraints 7

2.1.7 Operations 7

2.1.8 Site Adaptation Requirements 7

2.2 Product Functions 8

2.3 User Characteristics 8

2.4 Constraints 8

2.5 Assumptions and Dependencies 9

2.6 Apportioning of Requirements. 9

3. Specific Requirements 9

3.1 External Interfaces 11

3.2 Functions 11

3.3 Performance Requirements 12

3.4 Logical Database Requirements 12

3.5 Design Constraints 13

3.5.1 Standards Compliance 13

3.6 Software System Attributes 13

3.6.1 Reliability 13

3.6.2 Availability 13

3.6.3 Security 14

3.6.4 Maintainability 14

3.6.5 Portability 14

3.7 Organizing the Specific Requirements 15

3.7.1 System Mode 15

3.7.2 User Class 16

3.7.3 Objects 16

3.7.4 Feature 16

3.7.5 Stimulus 16

3. 7.6 Response 16

3.7.7 Functional Hierarchy 16

3.8 Additional Comments 16

4. Change Management Process 17

5. Document Approvals 17

6. Supporting Information 17

# 1. Introduction

## 1.1 Purpose

Nội dung của tài liệu này mô tả yêu cầu về việc phát triển phần mềm gửi nhận dữ liệu qua cổng mạng (Ethernet). Phần mềm có tên là SocketTest.

Nội dung của tài liệu mô tả một cách chi tiết các tính năng mà phần mềm cần phải có với các thông số cụ thể.

Nội dung trong tài liệu này cũng là cơ sở để nhiệm thu sản phẩm.

Sản phẩm được coi là đạt yêu cầu khi thỏa mãn tất cả nội dung yêu cầu trong tài liệu này.

## 1.2 Scope

Nội dung phát triển lần này bao gồm:

* Phần mềm SocketTest
* Architecture Design Documents
* Detailed Design Documents
* Unit Test Documents
* Integration Test Documents
* Deployment guideline documents bao gồm hướng dẫn biên dịch và hướng dẫn cách chạy chương trình.
* Use manual documents

## 1.3 Definitions, Acronyms, and Abbreviations.

|  |  |  |
| --- | --- | --- |
| **Item** | **Type** | **Description** |
| SocketTest | Definition | Tên phần mềm được yêu cầu phát triển lần này |
| SRS | Acronyms | Software requirement specification |
| SAD | Acronyms | Software architecture design |
| SDD | Acronyms | Software detailed design |
| UT | Acronyms | Unit test |
| IT | Acronyms | Integration test |
|  |  |  |

## 1.4 References

NA

## 1.5 Overview

Tài liệu được chia làm 3 phần:

1. Phần 1: Introduction – giới thiệu tổng quan về tài liệu này, bối cảnh của yêu cầu và các tham khảo, định nghĩa.
2. Phần 2: Oevrall Description – giới thiệu khái quát về môi trường hoạt động và chức năng của phần mềm.
3. Phần 3: Specific requirement – đây là nội dung cụ thể và chi tiết về các yêu cầu.
4. Phần 4: Change management process – phần này mô tả yêu cầu về quy trình và phương thức quản lý các thay đổi về yêu cầu.
5. Phần 5: Document Approval – Thể hiện trạng thái approval của tài liệu này.

# 2. The Overall Description

## 2.1 Product Perspective

SocketTest

(client)

SocketTest (server)

SocketTest

(client)

X socket server

SocketTest (server)

X socket client

### Bên trên là các mô hình hoạt động của SocketTest.

1. SocketTest có thể đóng vai trò là Server gửi/nhận dữ liệu với một SocketTest khác cói vai trò là một Client.
2. SocketTest đóng vai trò như một client gửi/nhận dữ liệu với một trình socket khác có vai trò là Server.
3. SocketTest đóng vai trò là một Server gửi/nhận dữ liệu với một trình socket khác có vai trò là một Client.

### 2.1.1 System Interfaces

Sử dụng socket để gửi/ nhận dữ liệu.

### 2.1.2 Interfaces

Cho phép người dung thao tác qua giao diện đồ họa.

### 2.1.4 Software Interfaces

Chương trình hoạt động trên nền tảng Windows Desktop các phiên bản:

* + Windows 7 Professional 32 và 64 bit
  + Windows 8.1 Professional 32 và 64 bit
  + Windows 10 Professional 32 và 64 bit

Source code phát triển dựa trên thư viện MFC/Visual Studio 2012

### 2.1.5 Communications Interfaces

Chương trình có thể gửi/nhận với các trình ứng dụng socket khác thông qua giao thức UDP và TCP

### 2.1.6 Memory Constraints

NA

### 2.1.7 Operations

Chương trình cung cấp giao diện đồ họa cho phép người dung thực hiện:

1. Chọn chế độ Server/ Client
2. Chọn giao thức gửi/nhận dữ liệu là UDP/TCP
3. Chỉ định địa chỉ nguồn nếu chế độ là Server và chỉ định địa chỉ đích nếu là Client
4. Cho phép người dung tạo các cấu trúc dữ liệu và điền nội dung để gửi đi.
5. Các cấu trúc dữ liệu và nội dung có thể được lưu lại trên đĩa với định dạng file để có thể sử dụng lại.
6. Các cấu trúc dữ liệu có thể được nhóm theo danh mục mà người dùng đặt tên
7. Có thể cho người dùng tạo sẵn danh sách dữ liệu cần gửi và chọn một hoặc nhiều cấu trúc dữ liệu cần gửi.
8. Cho phép người dùng gửi dữ liệu đã thiết lập sẵn một cách lặp đi lặp lại theo số lần và tần suất chỉ định.

## 2.3 User Characteristics

Giao diện người dùng sử dụng định dạng cửa sổ, có thể phóng to, thu nhỏ.

Chương trình có thể chạy nhiều thể hiện đồng thời trên cùng một máy tính một cách độc lập.

## 2.4 Constraints

NA

## 2.5 Assumptions and Dependencies

NA

# 3. Specific Requirements

## 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:*

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

## 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:*

1. *Validity checks on the inputs*
2. *Exact sequence of operations*
3. *Responses to abnormal situation, including*
4. *Overflow*
5. *Communication facilities*
6. *Error handling and recovery*
7. *Effect of parameters*
8. *Relationship of outputs to inputs, including*
9. *Input/Output sequences*
10. *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:*

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

*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

*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:*

1. *Utilize certain cryptographic techniques*
2. *Keep specific log or history data sets*
3. *Assign certain functions to different modules*
4. *Restrict communications between some areas of the program*
5. *Check data integrity for critical variables*

### 3.6.4 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. If someone else will maintain the system*

### 3.6.5 Portability

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

1. *Percentage of components with host-dependent code*
2. *Percentage of code that is host dependent*
3. *Use of a proven portable language*
4. *Use of a particular compiler or language subset*
5. *Use of a particular operating system*

*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.*

# 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?*

# Document Approvals

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

# Supporting Information

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

1. *Table of Contents*
2. *Index*
3. *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

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Mode 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

1. 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

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*

1. Design constraints
2. Software system attributes
3. 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

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 User class 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

1. 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

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Classes/Objects

3.2.1 Class/Object 1

3.2.1.1 Attributes (direct or inherited)

1. Attribute 1

.....

3.2.1.1.*n* Attribute *n*

1. 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

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. 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

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. 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

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. 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

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Information flows

3.2.1.1 Data flow diagram 1

1. Data entities
2. Pertinent processes
3. Topology

3.2.1.2 Data flow diagram 2

1. Data entities
2. Pertinent processes
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

1. Process 1
2. Input data entities
3. Algorithm or formula of process
4. 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

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. 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