**91.411.201: Software Engineering I**

**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, you might keep the header, but insert a comment saying why you omit the data.

**91.411.201: Software Engineering I**

**Bubble Warrior Adventures**

**Software Requirements Specification**

**Document**

**Version: (1)** **Date: (04/04/2017)**

**Table of Contents**

1. Introduction 51.1 Purpose 5

1.2 Scope 5

1.3 Definitions, Acronyms, and Abbreviations 5

1.4 References 5

1.5 Overview

5

2. The Overall Description 6

2.1 Product Perspective 6

2.1.1 System Interfaces 6

2.1.2 Interfaces 6

2.1.3 Hardware Interfaces 6

2.1.4 Software Interfaces 7

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 10

3.2 Functions 10

3.3 Performance Requirements 11

3.4 Logical Database Requirements 11

3.5 Design Constraints 12

3.5.1 Standards Compliance 12

3.6 Software System Attributes 12

3.6.1 Reliability 12

3.6.2 Availability 12

3.6.3 Security 12

3.6.4 Maintainability 13

3.6.5 Portability 13

3.7 Organizing the Specific Requirements 14

3.7.1 System Mode 14

3.7.2 User Class 14

3.7.3 Objects 14

3.7.4 Feature 14

3.7.5 Stimulus 15

3.7.6 Response 15

3.7.7 Functional Hierarchy 15

3.8 Additional Comments

15

4. Change Management Process

5. Document Approvals

6. Supporting Information 15

# 1. Introduction

## 1.1 Purpose

The purpose is to build a 2D RPG styled video game for Windows and UNIX environments.

## 1.2 Scope

The software being produced is called “Bubble Warriors Adventure”. Will have a storyline, controllable character, player vs enemy A.I. battling, obtainable items, and conversation with non-player characters, leveling up, and upgrade-able statistics for the player. Will not feature multiplayer option.

## 1.3 Definitions, Acronyms, and Abbreviations.

1. Bubble Warrior Adventures – BWA
2. Bubble Warrior Adventures – “this product”

## 1.4 References

1. SFML 2.4.2, Source: https://www.sfml-dev.org/

2. TGUI 0.7.3, Source: https://tgui.eu/

3. Python 3.6.1, Source: https://www.python.org/

## 1.5 Overview

The rest of the SRS details the overview and explanation about what the product aims to do in order to entertain the user. A lot of section 2 is fairly non-explanatory, as a lot of the section deals with topics unrelated to the product, as this is a standalone desktop program. Section 3 has almost all the important and interesting aspects to the product.

# 2. The Overall Description

## 2.1 Product Perspective

This product is a self-contained product. So long as the user has the product on their system, it will be able to be run.

### 2.1.1 System Interfaces

### This product has no system interfaces.

### 2.1.2 Interfaces

This product will have a standalone window, with a GUI for the menus, and an overlay GUI for the main gameplay. This is meant for any user who can use two hands effectively, and has an interest in this style of video game. A keyboard and mouse is required, or equivalent.

### 2.1.3 Hardware Interfaces

### This system has no hardware interface requirements. A basic, single Windows or Unix environment computer will be enough to run this program.

### 2.1.4 Software Interfaces

This product does not connect with other software interfaces. It only uses C++ libraries. The libraries are precompiled with the executable and does not have to be installed manually by the user.

### 2.1.5 Communications Interfaces

There are no communication interfaces or protocols in this system. Will be dependent on a single local system.

### 2.1.6 Memory Constraints

There are no memory constraints for this system, but will optimize memory usage as much as we can.

### 2.1.7 Operations

BWA will be played using both the keyboard and mouse, where the keyboard is for movement and interaction, while the mouse is mainly for GUI elements. The user would remain in the game for the entire operation, and would have the option to pause gameplay in order to leave the operation. The user can also save the current state of the system and be able to come back to it at a later date.

### 2.1.8 Site Adaptation Requirements

There is no need for any site adaptation, as the nature of the system is not of any critical importance.

## 2.2 Product Functions

The function of BWA is to provide enjoyment to the player by: allowing them to experience an immersive story, challenging the player's skills in combat, providing a game world for them to explore, and letting them customize their character's abilities by using special items (“Bubbles”).

The story should be interesting and relatable, and last approximately 10-20 hours assuming the standard amount of time spent exploring. Although the player should be given some choices in where to explore next, the story itself will not be open-ended and will have a definite plot that the player is meant to follow.

Combat will take place on the same map as exploration, and will be action oriented, following the standard set by games such as “Zelda: A Link to the Past.” Players will be able to control weapon swings using a button or mouse press, and damage will be based on the player or enemy being struck.

The game world should consist of several large maps, with enemies scattered across them, visible to the player as they explore. Apart from the main quest, there should be a few optional quests and areas to explore for the player to discover as the game progresses.

The player will be able to customize their character’s abilities by using special items known as “Bubbles,” which will confer powers on the player character when equipped. Bubbles will be able to become stronger over time, and the player will be able to customize them to suit their playstyle.

## 2.3 User Characteristics

People who enjoy video games would be the intended users of this product. Our UI design will try to make the gameplay easy for the user, but yet interesting and unique enough it is enjoyable. We are also keeping in mind that the graphics need to be pleasant enough for the user to look at for a long period of time, as this game will not be a fast one to complete, or at least should not be.

## 2.4 Constraints

Our only limitations would be how far our libraries can go in regards to appearance and usability.

## 2.5 Assumptions and Dependencies

Any dependencies that the system requires would not cause any strain on the SRS to change. This program runs off of only C++ and the libraries we have chosen. Almost no factors of the SRS could affect the system as a whole if it were to be changed.

## 2.6 Apportioning of Requirements.

Requirements that the group has thought about include a web based version of the same game. That would most likely be postponed until the next semester, as it would require a major overhaul to how the system runs.

# 3. Specific Requirements

The game shall load in .tmx files created via the program Tiled.

The game shall render the map in layers.

The game shall handle keyboard/mouse input to interact with the player sprite.

The game should contain sound effects.

The game shall contain entities (enemies, NPCs).

The game shall contain a storyline.

## 3.1 External Interfaces

JSON is used to store settings and savegames.

## 3.2 Functions

The game shall load in .tmx files created via the program Tiled.

The game shall render the map in layers.

The game shall handle keyboard/mouse input to interact with the player sprite.

The game should contain sound effects.

The game shall contain entities (enemies, NPCs).

The game shall contain a storyline.

## 3.3 Performance Requirements

To be runnable on Windows, Linux and Mac with similar performance.

The game should run at 60 FPS on modern hardware and smoothly with no graphical disformation.

## 3.4 Logical Database Requirements

This product has no database, and all data saved using various extensions: .tmx, .conf, .txt

## 3.5 Design Constraints

This product has no design constraints imposed by other standards or hardware limitations.

### 3.5.1 Standards Compliance

Uniform coding style is written on the github CONTRIBUTING.md. No other standards or regulations apply. All data is not sensitive, and this product’s purpose is for entertainment and for our software engineering course.

## 3.6 Software System Attributes

### 3.6.1 Reliability

The game should run whenever the user would like to play it. It should never throw exceptions unless for missing files or data corruption. The user will also be able to save the game.

### 3.6.2 Availability

The game works by saving your progress as you progress, in case anything happens you can choose to reopen the game with the current progress you last had or forget about it. So it will never conflict with your previous savegames.

### 3.6.3 Security

The game does not connect to any server or ask the user for any information except a name for the player, so security is not critical. For the savegames, however, they are encrypted to prevent cheating by modifying values.

### 3.6.4 Maintainability

All coding styles and descriptions of all code are maintained on the GitHub repository, and anyone can modify the code based on the coding style found in the CONTRIBUTING.md file.

### 3.6.5 Portability

The game runs on Windows, Linux and Mac. The game should run wherever SFML can run.

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

# Change Management Process

We communicate via Gitter and Skype whenever chatting face to face is not fesible. We have to reach an agreement and find a solution for a particular change. We then divide it up and we work on it as a mini sprint and check in everyday until three days pass where we meet up and test each other’s code and finally begin to integrate them together into one to then add to the game.

# Document Approvals

We the developer behind Bubble Warrior Adventures approve this document.

# Supporting Information

The table of contents, index and appendix can be used as such.

**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