SE 3XA3: Software Requirements Specification
Snake 2.o

Team 30, VUA30
Andy Hameed — Hameea1
Usman Irfan — Irfanm7
Vaibhav Chadha — Chadhav

November 9, 2018

Contents

List of Tables

List of Figures

1 Introduction

1.1 Overview

Almost everyone nowadays relies on a computer as a multipurpose tool for research, video streaming, gaming and many other tasks. With the emergence of fast computing, gaming has become a popular pastime activity and a source of entertainment for many. However, not everyone has a device powerful enough to support extensive game applications. A simple, memory-effecient application of the Snake game allows it to be accessible for gamers without the need for extensive hardware or a high-performance computer. Our team, VUA30, will be creating a desktop application for the well-known Snake game with new enhancements and features. This competitive and addictive game can allow the user to play at their own pace and challenge their own high score.

1.2 Scope

The scope of the project will cover the reimplementation of the game as well as enhancements to the game, in particular, changes that pertain to gameplay themes, different modes/difficulties and so on. Due to the time constraints of the project, not all enhancements will be implemented but at least one enhancement will be implemented to differentiate Snake 2.o from previous remakes of the game.

1.3 Relevance of Module Guide

The Module Guide (MG) document will outline the structure of the reimplementation through modular programming. It highlights the components of the system such that each component can easily be identified by project members, whether they are maintaining or designing the software. Among the list of internal stakeholders that may make use of this document, any new members that are added to the team will also be able to use the MG as a convenient reference to specific modules that they are concerned with or working on (Parnas et al., 1984).

Table 1: **Revision History**

Date	Version	Notes
2018 -11-09	1.0	Andy added section 1, 7 and 5.3, Vaibhav added section 2,3 and 5.1, Usman added section 4,5.2 and 6
Date 2	1.1	Notes

1.4 SRS,MIS and MG

In the SRS, functional and non-functional requirements were specified along with testing methods. In this document, we will be covering the modules that will implement the functional requirements specified in the SRS. The MIS will outline the module attributes and methods that are needed to meet the functional requirements as well as descriptions of these attributes and methods and their purpose.

The document will have the following structure:

1. Introduction: Connecting the relevance of the MIS, SRS and the MG along with the scope and a reiteration of the project overview

2. Anticipated changes: This includes changes that will be made to this document including the addition and removal of modules and othe module-related changes that can come about in the future.

3. Module Heirarchy: Will list the modules that will be implemented, giving an overview of the module structure.

4. Traceability Matrix: Shows the connection between modules and how errors could be traced back to each module

5. Use Heirarchy: a visual of the modules and their working relationship between each other. an arrow from A to B explains that module A uses module B

2 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 2.1, and unlikely changes are listed in Section 2.2.

2.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The specific hardware on which the software is running.

AC2: The Operating System of which the software interfaces with.

AC3: The new High Score after any previous record is broken.

AC4: The speed of the snake when the user changes the difficulty level.

AC5: Storing the score to the text file after each game is played.

AC6: The theme of the playground is changed whenever the user decides changes the theme mode.

AC7: Default settings for inputs.

2.2   Unlikely Changes

UC1: Input/Output devices (The system assumes mouse, keyboard and screen are available).

UC2: The snake is responsive to the directions button under any circumstances.

UC3: The goal of the system: To provide user with entertainment and a fun game to play.

UC4: There will always be a source of input data external to the software.

3   Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module

M2: Interface Module

M3: Snake Module

M4: Food Module

M5: Themes Module

M6: Software Design Module

M7: init Module

M8: highscore Module

M9: Gameplay Module

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Format Module Snake Module Food Module Themes Module
Software Decision Module	HighScore Module

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

The Design of the software product was designed to meet the functional and non-functional requirements. The user-interference file displays the game interface that allows the user to either start the game or to select different modes, themes or even check the high score. The design was kept to keep the interface simple and easy to use. When the user runs the main file, it opens a title page which has multiple options from which a user can select options. To meet the functional requirement of outputting high score, the main interface file has a button which when pressed open a new window displaying the highest score of the game so far. A quit button and a Main Menu button has been added in the high score window so the user can either go back or end the game. In the Main Menu, there are different themes combined that the user can select letting us meet another requirement. The principal part of the design was to open a new window which begins the snake game. It was created by adding a new button "Game Time" in the Main Menu window. The design of the snake game is kept simple where a snake and food appears randomly on the window, upon pressing the Arrow direction keys the snake moves to the respective location proceeding the game smoothly and connecting our requirement to its design. The current score of the game displays on the top which keeps updating as the snake eats the food, and a quit button will be added on the bottom of the screen so the user can quit the game whenever they feel like.

To enhance our design, in the future the group has planned to add radio buttons, drop-down menus or use a slider to make the game interactive. The radio buttons would be installed where the user can select the difficulty modes of the game, the drop-down menus would be helpful in selecting the theme and the slider would work to alter the speed of the snake.

5 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. (1984). The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the

module is provided by the operating system or by standard programming language libraries. Also indicate if the module will be implemented specifically for the software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented. Whether or not this module is implemented depends on the programming language selected.

5.1 Hardware Hiding Modules

Secrets: user controls

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can take inputs from the keyboard and mouse, and then further output it on the screen.

Implemented By: Pygame library and OS

5.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

5.2.1 Interface Module

Secrets: Input Data

Services: Collects data on customized fields in the game such as speed, theme, difficulty and other important variables.

Implemented By: Pygame library

5.2.2 Snake Module

Secrets: Snake

Services: Defines the snake class and its attributes and behaviours. This includes the movement of the snake depending on its attributes and its interaction with user events.

Implemented By: Pygame library

5.2.3   Food Module

Secrets: Food

Services: Defines the food item class for spawning the food item during gameplay.

Implemented By: Pygame library

5.2.4   Themes Module

Secrets: Themes

Services: Allows the user to choose different themes of the snake game.

Implemented By: Pygame library

5.2.5   Gameplay Module

Secrets: gameplay

Services: connects different modules to the Pygame event functionality and creates movement of snake around window.

Implemented By: Pygame library

5.2.6   highscore Module

Secrets: highscore

Services: provides highscore of player on a separate menu after initial start menu.

Implemented By: Pygame library

5.3   Software Decision Module

Secrets: Text files

Services: Creates a text file which stores the game score

Implemented By: N/A

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
FR1	M2, M3
FR2	M2, M3
FR3	M2, M3
FR4	M2, M3
FR5	M2, M3
FR6	M2, M3
FR7	M6
FR8	M3
FR9	M2, M5
FR10	M6
FR11	M5
FR12	M3
FR13	M3
FR14	M3
FR15	M4
FR16	M3
FR17	M3
FR18	M4
FR19	M6

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M2, M6
AC3	M6
AC4	M5,M6
AC5	M6
AC6	M5
AC7	M2

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

Figure 1: Use hierarchy among modules
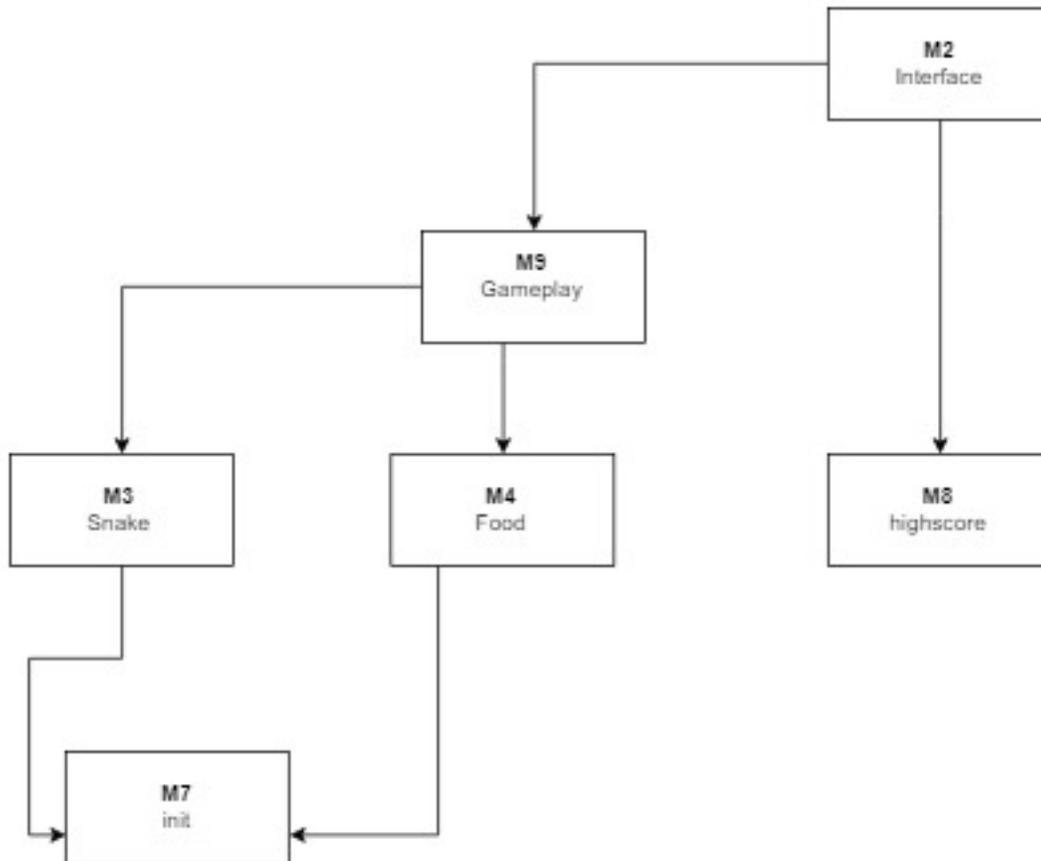
References

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.