SE 3XA3: Module Guide
Ratava

Team 9, Makiam Group
Aidan McPhelim - mcpheima
Alexie McDonald - mcdona16
Illya Pilipenko - pilipeni

December 6, 2018

Contents

List of Tables

List of Figures

1 Introduction

Having separate modules within a software system makes things so much easier for said software's development, not only while making it, but long after the software has been deployed and people are going back to maintain it. Having this breakdown will also serve as a reference for new members beginning work on the software, so that they can gain an understanding of what the software is currently doing and save time.

In this document you will find a breakdown of each module in the software, what it's purpose is and how it relates to the Requirement Specification Document(SRS). You will gain an understanding of why each module was made as it is, depending on things like fields that we want to be hidden because it is likely to change; a list of anticipated changes can be seen in section 2. The full module decomposition for this idea can be found in secion 5.

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 templates used to generate avatars.

AC2: The input criteria.

AC3: The output file's format.

AC4: Available hash functions.

Table 1: **Revision History**

Date	Version	Notes
2018-09-18	0.0	File structure added
2018-10-09	1.0	Added sections 1, 2 and 6
2018-10-09	1.1	Module guide and Breakdown added
2018-10-09	1.2	Uses hierarchy between modules added
2018-12-03	1.3	Updated team name
2018-12-05	1.4	Updated module decompositions

AC5: The range of colour pallets available to the user.

AC6: The range of unique output shapes.

AC7: The graphics library used for generation.

2.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

UC2: The format in which the user enters the input String.

UC3: The type of data the user enters as input.

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: Hash Generating Module

M2: GUI Module

M3: Graphics Drawing Module

M4: Template Drawing Module

M5: Hash Using Module

4 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

As can be seen in the SRS, the functional requirements in particular focus around a few

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	GUI Module Graphics Drawing Module Hash Using Module Hash Generating Module
Software Decision Module	Template Drawing Module

Table 2: Module Hierarchy

specific concepts, like the hash functionality. In order to best satisfy these requirements, we decided it would be best to have a GenerateHash module, which handles all the types of Hash's available and does all related things within it. We decided to keep functionality which made use of the hash output separate since what we are generating and how we use that information is subject to change as can be seen in section 2.

There are lots of requirements, non-functional and functional, which are about the users customization and usability, so we thought it would be best to keep all of that stuff in one neat module about the GUI, which can handle all user input and output.

As for requirements regarding the generated avatar, we decided to split it into two modules, one for storing the templates so that it can be easily expanded, and one which handles the actual drawing and combining of these templates, which means if we want to change the way this is done, it is separate from the templates themselves.

For a detailed view on how each individual requirement is represented by the modules, see section 6 for the Traceability Matrix.

5 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by **?**. 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: The data structure and algorithm used to implement the virtual hardware.

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 use it to display outputs or to accept inputs.

Implemented By: 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 Hash Generating Module (M1)

Secrets: Generates a hash value.

Services: Generates a hash value from a given string using a chosen hashing function.

Implemented By: Using hashlib to generate a hash value from a string.

5.2.2 GUI Module (M2)

Secrets: Allows user to interface with system.

Services: Lets the user pick the avatar template, The colours used, the file information, and what string to use for hashing with what kind of hashing function.

Implemented By: Using tkinter to create a visual interface for the user.

5.2.3 Graphics Drawing Module (M3)

Secrets: Creates avatar image file.

Services: Creates the .jpg file with the generated avatar at the user-specified location, and opens it.

Implemented By: Using M5, M4, and M2 to get information on what to make the avatar look like, and Pillow to create the image file.

5.2.4 Hash Using Module (M5)

Secrets: Uses a hash value to randomize avatar generation.

Services: Takes a hash value and uses it to generate sets of colours and randomize certain avatar parts.

Implemented By: Converting parts of a hash value to integers within the part tuple size, and converting parts of a hash value to integers between 0-255 for RGB values.

5.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

5.3.1 Template Drawing Module (M4)

Secrets: Provides avatar templates.

Services: Provides information on what avatar templates there are and how to draw each template.

Implemented By: List of available templates, and their corresponding pixel arrays.

6 Traceability Matrix

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

Req.	Module 1	Module 2	Module 3	Module 4	Module 5
R1		X			
R2					X
R3		X			
R4		X			
R5		X			
R6		X			X
R7		X			
R8		X			
R9		X			
R10		X			
R11		X			
R12		X			
R13		X			
R14		X			
R15		X			
R16		X			
R17		X			
R18	X		X	X	X
R19	X		X	X	X
R20			X	X	
R21					
R22					
R23					
R24					
R25					
R26				X	
R27	X				
R28			X		
R29	X	X	X	X	X
R30					
R31	X				X
R32	X				
R33					
R34			X	X	
R35					
R36					

Table 3: Trace Between Requirements and Modules

AC	Module 1	Module 2	Module 3	Module 4	Module 5
AC1				X	
AC2		X			
AC3		X			
AC4	X				
AC5		X			
AC6			X		
AC7			X		

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. **?** said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

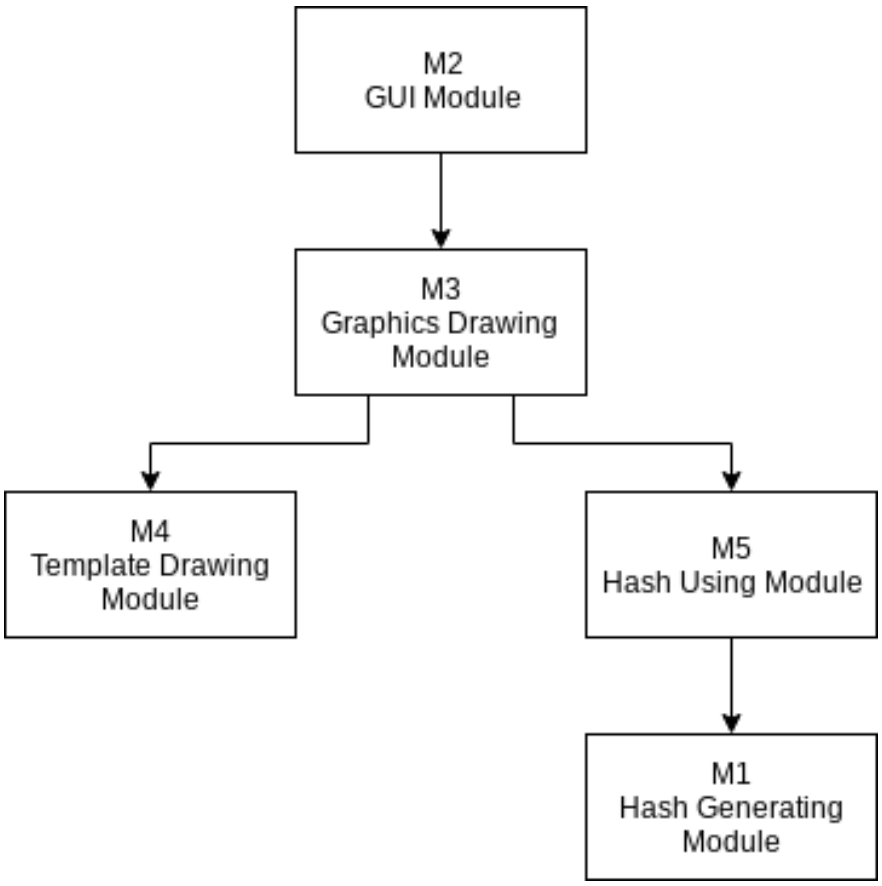

Figure 1: Use hierarchy among modules