Module Guide for SFWRENG 4G06 Capstone Design Project

Team #18, InfiniView-AI
Anhao Jiao
Kehao Huang
Qianlin Chen
Qi Shu
Xunzhou Ye

January 17, 2024

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
UC	Unlikely Change
M	Module
SRS	Software Requirements Specification
HA	Hazard Analysis
VnVPlan	Verification and Validation Plan
MIS	Module Interface Specification
MVC	Model-view-controller, a well-known software design pattern
FR	Functional Requirement
LF	Look and Feel Requirement
PR	Performance Requirement
OE	Operational and Environmental Requirements
HS	Health and Safety Requirements
MS	Maintainability and Support Requirement
SR	Security Requirement
CR	Cultural Requirement
LR	Legal Requirement
HTTP	Hypertext Transfer Protocol
OS	Operating System
STUN	Session Traversal Utilities for NAT - a type of server needed for setting up peer-to-peer conn
RTC	Real-Time Communication
SFU	Selective Forwarding Unit - A software unit that can selectively forward video streams

Contents

List of Tables

List of Figures

3 Introduction

This Module Guide Document serves as a design blueprint for MotionMingle - a WebRTC-based video conferencing application designed to enrich the experience of Tai Chi instruction. It presents a modular perspective of the project, guiding the development team in creating a platform that emphasizes real-time interaction, minimal hardware requirements, and ease of use.

3.1 Overview

This Module Guide Document serves as a design blueprint for MotionMingle - a WebRTC-based video conferencing application designed to enrich the experience of Tai Chi instruction. It presents a modular perspective of the project, guiding the development team in creating a platform that emphasizes real-time interaction, minimal hardware requirements, and ease of use.

The document aligns with standards outlined in various project documents, including the SRS, HA, VnVPlan, SystDes, and MIS.

SRS - /docs/SRS/SRS.pdf

HA - /docs/HazardAnalysis/HazardAnalysis.pdf

VnVPlan - /docs/VnVPlan/VnVPlan.pdf

MIS - /docs/Design/SoftDetailedDes/MIS.pdf

3.2 Purpose

The aim of this Module Guide Document is to detail the architecture of modules, based on selected design principles and patterns, to clarify the project's functionalities and the specific roles of each module.

3.3 Design Principles

The Module Guide Document employs principles like information hiding, high cohesion, low coupling, high fan-in, and low fan-out for breaking down modules. These principles involve identifying and securing anticipated changes.

3.4 Design Pattern

The project utilizes the MVC design pattern, facilitating the division of complex problems into manageable sub tasks across different modules, adhering to the separation of concerns

principle.

4 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 4.1, and unlikely changes are listed in Section 4.2.

4.1 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 adopted here is called design for change.

AC1: Video stream sharing by practitioners might be allowed for better user communications.

AC2: Audio stream sharing by practitioners might be allowed for better user communications.

AC3: The number of annotation types available for users to choose from might change as discovered through the development process.

AC4: The initial release of the application has a limited load capacity, which will need to be expanded as the application's popularity grows to accommodate a growing number of users.

AC5: The exactness of annotations is subject to change, given the lack of definitive measurement standards.

AC6: The application may be extended to offer native client implementation.

AC7: The application may be extended to offer mobile client implementation.

AC8: The application could undergo enhancements with new features based on real-world usage insights.

AC9: The application might incorporate an account login system to verify the identity of instructors.

4.2 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: The need for specific input/output devices is expected to remain stable, given the application's primary function is to augment the instructor's video stream with annotations.

UC2: The necessity for a stable internet connection should persist, as the application depends on HTTP requests for communication between server and client.

UC3: The application's ease of use is anticipated to remain a constant requirement, catering to its primary user base of elderly individuals.

UC4: The requirement for the application to be easy to learn is expected to stay unchanged, considering its elderly user demographic.

UC5: The demand for server reliability and availability is projected to be consistent, due to its critical role in the application's functionality.

UC6: The system is intended to be compatible with the latest versions of Windows, Linux, and macOS.

UC7: Browser compatibility is a designed feature of the system.

UC8: Unauthorized modifications to the application are to be prevented.

UC9: The fundamental feature of adding annotations to the video stream is not expected to change.

UC10: The application's design, ensuring compliance with relevant laws, is foreseen to remain unaltered.

UC11: The client for this project is not expected to change, as it is designed as a student capstone project with limited market potential.

UC12: The stakeholder for this project is unlikely to change, as it is a student capstone project adhering to specific course guidelines.

UC13: The project's schedule and budget constraints are anticipated to stay fixed, in line with the requirements of the student capstone course.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Media Control Module

M2: User Authentication Module

M3: Instructor View Module

M4: Practitioner View Module

M5: Annotation Configuration Module

M6: STUN Server Module

M7: RTC Control Module

M8: App Module

M9: Video Transform Module

M10: Human Pose Estimation Annotation Module

M11: Center of Mass Annotation Module

M12: SFU Server Module

Level 1	Level 2
Hardware-Hiding Module	M13
Behaviour-Hiding Module	M2
	M3
	M4
	M5
	M7
Software Decision Module	M6
	M8
	M9
	M10
	M11
	M12

Table 1: Module Hierarchy

6 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.

7 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. *SFWRENG 4G06 Capstone Design Project* means the module will be implemented by the SFWRENG 4G06 Capstone Design Project 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.

7.1 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

7.1.1 Media Control Module (M1)

Secrets: The implementation for capturing camera and microphone input from clients.

Services: Enables users to turn on/off camera and microphone to upload audio and video.

Implemented By: JavaScript, WebAPI

Type of Module: Abstract Data Type

7.2 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: –

7.2.1 User Authentication Module (M2)

Secrets: The authentication mechanism and user credentials storage format.

Services: Authenticates users to grant access to the system based on credentials.

Implemented By: JavaScript, React

Type of Module: Abstract Object

7.2.2 Instructor View Module (M3)

Secrets: Instruction delivery and progress tracking mechanisms for instructors.

Services: Provides instructors with tools to manage course streaming, and interact with participants.

Implemented By: JavaScript, React

Type of Module: Abstract Object

7.2.3 Practitioner View Module (M4)

Secrets: User interface customization based on practitioner's preferences.

Services: Allows practitioners to view available annotations, and select preferred annotations.

Implemented By: JavaScript, React

Type of Module: Abstract Object

7.2.4 Annotation Configuration Module (M5)

Secrets: The configuration options for annotation types and parameters.

Services: Enables users to select and configure various annotation types for personalized course delivery.

Implemented By: JavaScript, React

Type of Module: Abstract Data Type

7.2.5 RTC Control Module (M7)

Secrets: The peer-to-peer connection mechanism and connection between the client and the server.

Services: Provides other modules the ability to manage WebRTC connections.

Implemented By: JavaScript, WebAPI

Type of Module: Abstract Data Type

7.3 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: –

7.3.1 APP Module (M8)

Secrets: The business logic that coordinates the flow of data between modules.

Services: Manages application state and serves as the central communication hub for other modules.

Implemented By: JavaScript, React

Type of Module: Abstract Object

7.3.2 Video Transform Module (M9)

Secrets: Algorithms for video encoding, decoding, and transformation.

Services: Process video streams to apply transformations such as scaling, rotating, and overlaying annotations.

Implemented By: Python

Type of Module: Library

7.3.3 Human Pose Estimation Annotation Module (M10)

Secrets: The algorithms and data structures used to calculate and represent the skeletal structure of a human figure in video streams.

Services: Processes video data to overlay a representation of the human skeleton, aiding in the correction and learning of Tai Chi movements.

Implemented By: Python, opencv-python

Type of Module: Abstract Object

7.3.4 Center of Mass Module (M11)

Secrets: The methods used to determine the center of mass in a given pose or series of movements.

Services: Calculates and displays the center of mass in the video feed to help practitioners understand and improve their balance in various Tai Chi poses.

Implemented By: Python, opencv-python

Type of Module: Abstract Object

7.3.5 SFU Server Module (M12)

Secrets: The implementation of the Selective Forwarding Unit (SFU) for handling media streams.

Services: Facilitates the routing of media streams between users to support multi-party video conferencing.

Implemented By: Python, aiortc

Type of Module: Abstract Object

7.3.6 STUN Server Module (M6)

Secrets: Network traversal techniques and session negotiation mechanisms.

Services: Assists in NAT traversal by resolving public IP addresses and enabling peer-to-peer connections.

Implemented By: Implemented externally

Type of Module: Abstract Object

8 Traceability Matrix

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

Functional Requirements	Modules
FR1	M1, M2, M3
FR2	M1, M2, M4
FR3	M1, M2, M3, M4
FR4	M1, M2, M6, M8,
FR5	M1, M2, M6, M8,
FR6	M1, M2, M5
FR7	M1, M9
FR8	M1, M10, M12
FR9	M1, M12
FR10	M1, M12
FR11	M1, M9, M10, M11, M12
FR12	M1, M6, M8, M9, M12
FR13	M1, M6, M8

Table 2: Trace Between Requirements and Modules

Functional Requirements	Modules
LF1	M1, M2, M3
UH1	M1, M2, M4
UH2	M1, M2, M3, M4
PR1	M1, M2, M6, M8,
PR2	M1, M2, M6, M8,
PR3	M1, M2, M5
PR4	M1, M9
PR5	M1, M10, M12
PR6	M1, M12
PR7	M1, M12
PR8	M1, M9, M10, M11, M12
PR9	M1, M6, M8, M9, M12
PR10	M1, M6, M8

Table 3: Trace Between Requirements and Modules

Non-functional Requirements	Modules
LF1	M3, M4
UH1	M3, M4
UH2	M3, M4
PR1	M3, M4, M8
PR2	M7, M12, M6
PR3	M7, M12, M6
PR4	Everything
PR5	M7
PR6	M8
PR7	M3, M4
PR8	M6
PR9	M1
PR10	M3, M4
PR11	M3, M4
PR12	M5, M9, M10, M11
PR13	M1, M3, M4
PR14	M11, M12
PR15	M3, M4
OE1	M1
OE2	M1
OE3	M1
HS1	M1, M8
HS2	M3, M4
MS1	M1
MS2	M1
SR1	M2
SR2	M2, M3, M4
SR3	M2
SR4	M3, M4
SR5	M3, M4
SR6	M3, M4
CR1	Everything
LR1	Everything
AC1	M3, M4, M9
AC2	M3, M4, M9
AC3	M5, M8
AC4	M8
AC5	M7, M8
AC6	M3, M4

Anticipated Change	Module
AC1	M3, M4, M9
AC2	M3, M4, M9
AC3	M5, M8
AC4	M8
AC5	M7, M8
AC6	M3, M4
AC7	M3, M4
AC8	M7M
AC9	M2

Table 5: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas (1978) 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

References

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.

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.