UNIVERSITY OF APPLIED SCIENCES
ASCHAFFENBURG

Analysis Model

Project: Brain-Body-Computer Interface

Phase: Requirements Analysis

Author:
Ayberk Tuzcuoglu

Projekt Member:
Julian Emrich
Maximilian Spahn
Till Wolf

Documentname: AB-Brain-Body-Computer Interface-AM-2

Version: 1.0

Creation date: 2023-05-19

Modifications – Document Status

Version	Status	Creation Date	Editor	Modifications
1.0	Under construction	19.05.2023	Ayberk Tuzcuoglu	Initial Version

(Status ::= planned, under construction, presented, accepted)

TABLE OF CONTENTS

1 Introduction

The following Document is a component of the Analysis Model and explains the sequences within our program accompanied by relevant diagrams. With these diagrams you get gradually an exact overview of the program.

1.1 Purpose

The purpose is to finalize the Analysis model and to put all demands, requirements and information from the SRS into diagrams, such as class diagrams.

1.2 Scope

The game described in this document aims to showcase the capabilities of bio-sensors in an entertaining way. The goal of this project is to develop a game that uses bio-sensors to control its gameplay.

The game begins by selecting a person from the audience as a target. The task is to control the Moving Head with your mind and aim it towards the selected person in the audience.

The game evaluates the performance based on how accurate and how fast the player can aim the moving head at the selected person.

1.3 Definition, Acronyms and Abbreviations

API	Application Programming Interface
USB	Universal Serial Bus
DMX	(Digital Multiplex) A protocol for controlling stage lighting
EEG	Electroencephalogram

1.4 References

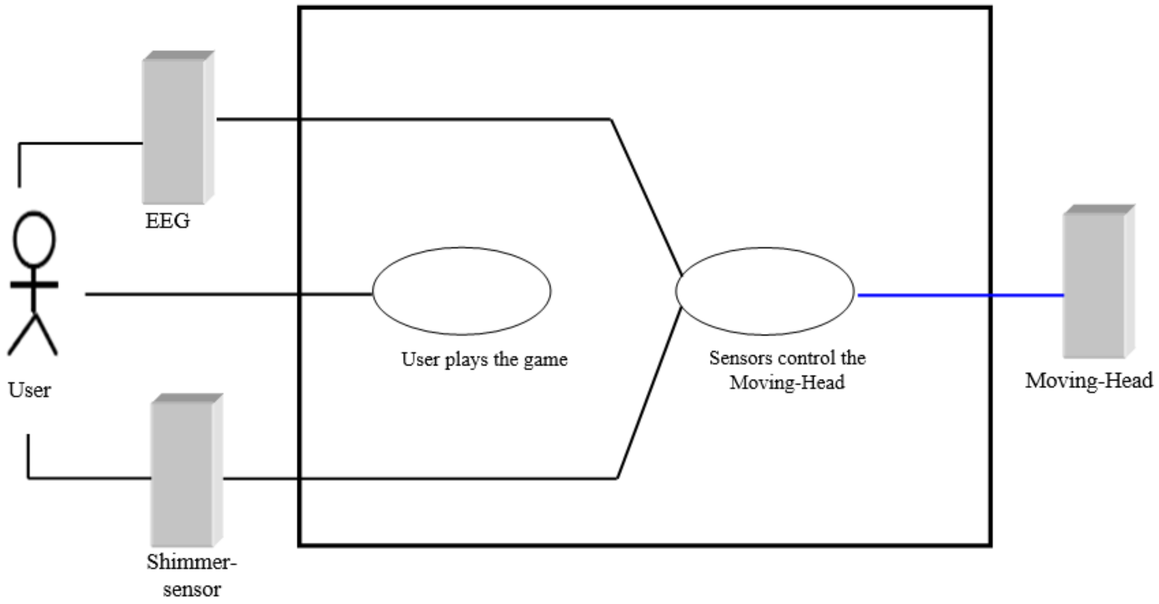
[AB-INF-PRC-1]	Programming guideline: Programmierrichtlinien für C/C++, Prof. Dr.-Ing. Konrad Doll, V 1.1
[SRS]	Maximilian Spahn: Software Requirements Specification, Hochschule Aschaffenburg, Version 1.0

1.5 Overview

In the second section, the sequence diagram provides an overview of the entire project, while the use case diagram and the class diagram specifically focus on detailing the code for executing individual peripheral modules.

2 Object-Oriented Analysis

2.1 Use Case Diagram



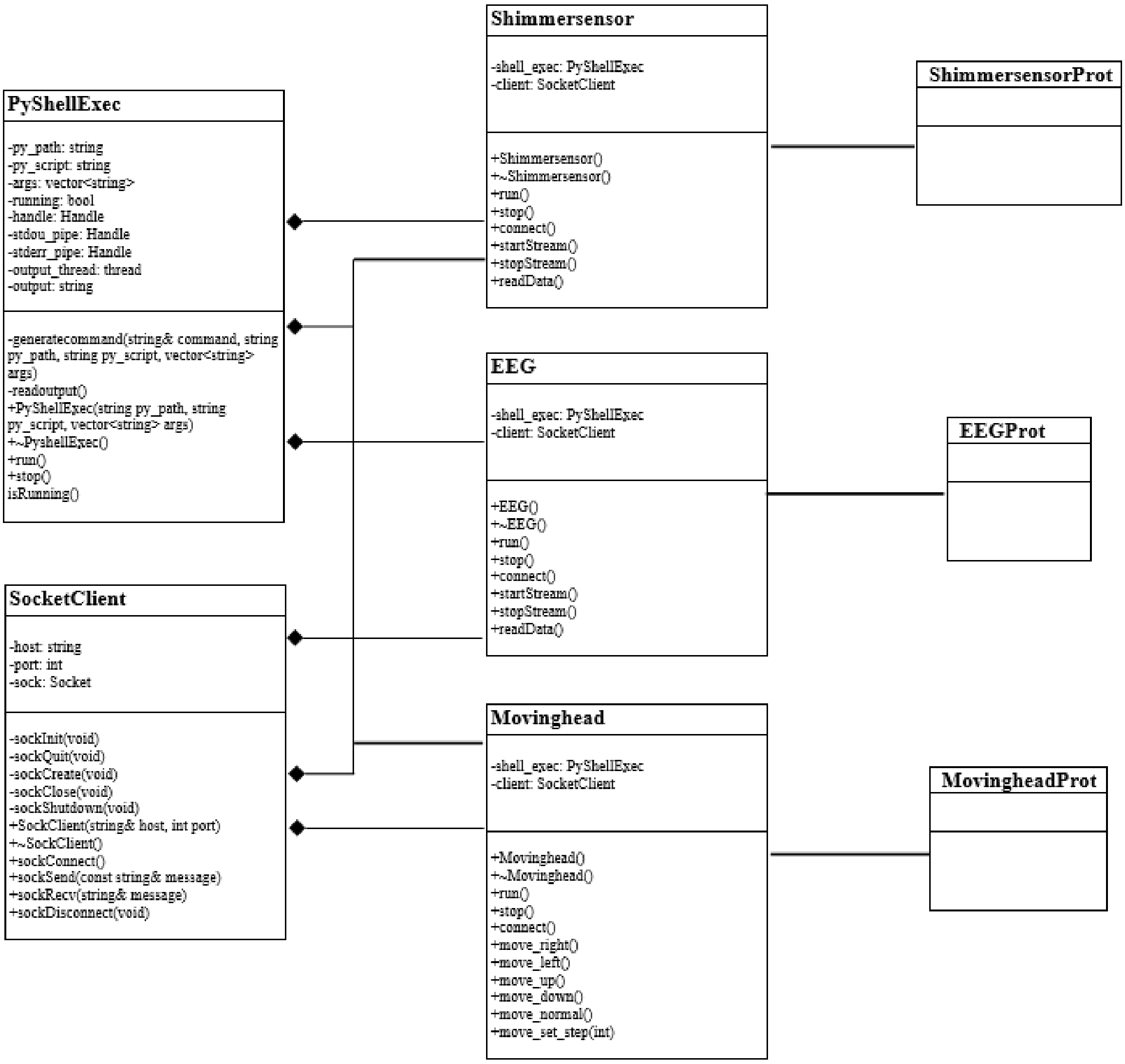
Actors: User, EEG, Shimmer-sensor, Moving-Head
Activator: Start of the Software
Results: Moving-Head points to the selected target

User options:

- Before starting the game interface appears, providing various options to the user.
- The user selects an option for the difficulty of the game.
- The user selects option “select target” to choose a person from the audience as a target.
- The user connects to the EEG and the Shimmer-sensor.
- Once the sensors are successfully connected the user can start the game by selecting the “start game” option.
- The user starts thinking about controlling the Moving-Head while the mental commands being detected by the EEG.
- The game uses the captured mental commands to control the Moving-Head
- The user’s score is displayed on the screen.
- The user can choose to exit the game by closing the software or selecting an option to quit.

2.2 Static Model

2.2.1 Class Model



2.2.2 Class Descriptions

PyShellExec

Purpose:

The class PyShellExec contains methods for executing Python files.

Collaborations:

No collaborations

Attributes:

- py_path
  description the path where the Python executable is located.
  type string

- py_script
  description the path where the Python script is located.
  type string

- args
  description vector that stores the arguments passed to the script as multiple strings.
  type vector<string>

- running
  description provides status indicating whether the respective script is being executed or not.
  type bool

- handle
  description reference to the process in which it is currently running.
  type Handle

- stdou_pipe
  description standard output pipe, reference to the pipes in which the console outputs are transferred.
  type Handle

- stderr_pipe
  description standard error pipe, reference to the pipes in which the console outputs are transferred.
  type Handle

- output_thread
  description reference to a process that reads the output pipes.
  type thread

- output
  description string in which the console output of the running script is buffered as a text file.
  type string

Operations:

- generateCommand
  description generates the console command that is executed.
  arguments reference to a string in which something is written, the path to python, the path to the script, arguments for the script (string &command, string py_path, string py_script, vector<string> args)
  return value void

- readoutput
  description function that is called in a separate thread and reads console output stream from the pipes.
  arguments no arguments
  return value void

- PyShellExec
  description Constructor of PyShellExec
  arguments the path to python, the path to the script, arguments for the script (string py_path, string py_script, vector<string> args)
  return value no return value.

- ~PhyShellExec
  description destructor of PyShellExec
  arguments no arguments
  return value no return value.

- run
  description launching the python script.
  arguments no arguments
  return value bool

- stop
  description stopping the python script.
  arguments no arguments
  return value bool

- isRunning
  description checks if the script is currently running.
  arguments no arguments
  return value bool

SocketClient

Purpose:

The class contains methods to connect with the socket server in Python scripts and represents a socket client.

Collaborations:

No collaborations

Attributes:

- host
  description the IP address of the host.
  type string

- port
  description port on which the server listens.
  type int

- sock
  description reference to the currents socket.
  type Socket

Operations:

- sockInit
  description initializing the socket.
  arguments (void)
  return value int

- sockQuit
  description closes the entire socket object.
  arguments (void)
  return value int

- sockCreate
  description creates the socket.
  arguments (void)
  return value int

- sockClose
  description closes the socket (Linux, not required for Windows).
  arguments (void)
  return value int

- sockShutdown
  description closes the socket connection.
  arguments (void)
  return value int

- SocketClient

  description Constructor of SocketClient. Executes the methods sockInit and sockCreate.

  arguments reference to the host IP address, integer for the port (const string& host, int port)

  return value no return value.

- ~SocketClient

  description destructor of SocketClient. Executes the methods sockShutdown then sockQuit.

  arguments no arguments

  return value no return value.

- sockConnect

  description connects to the socket.

  arguments no arguments

  return value int

- sockSend

  description sends the string message.

  arguments reference to the string message (const string& message)

  return value int

- sockRecv

  description checks if data is available and tries to receive it. Returns the status indicating whether it was successful or not.

  arguments reference to a string where the output is written (string& message)

  return value int

- sockDisconnect

  description disconnects from the socket. Calls the method sockClose.

  arguments (void)

  return value int

Shimmersensor

Purpose:

The class Shimmersensor contains methods for interacting with the Shimmer-sensor.

Collaborations:

As you can see from the static class model, Shimmersensor has one member of PyShellExec and one member of SocketClient.

Attributes:

- shell_exec
  description saves the shellexec object.
  type PyShellExec

- client
  description saves the socketclient object.
  type SocketClient

Operations:

- Shimmersensor
  description Constructor of Shimmersensor
  arguments no arguments
  return value no return value.

- ~Shimmersensor
  description destructor of Shimmersensor
  arguments no arguments
  return value no return value.

- run
  description executes the Python script.
  arguments no arguments
  return value void

- stop
  description stops the Python script.
  arguments no arguments
  return value void

- connect
  description connects to the python script.
  arguments no arguments
  return value void

- startStream
  description starts the stream.
  arguments no arguments
  return value void

- stopStream

description	stops the stream.
arguments	no arguments
return value	void

- readData

description	reads data coming from the sensor.
arguments	no arguments
return value	void

EEG

Purpose:

The class EEG contains methods for interacting with the EEG.

Collaborations:

As you can see from the static class model, EEG has one member of PyShellExec and one member of SocketClient.

Attributes:

- shell_exec
  description saves the shellexec object.
  type PyShellExec

- client
  description saves the socketclient object.
  type SocketClient

Operations:

- EEG
  description Constructor of EEG
  arguments no arguments
  return value no return value.

- ~EEG
  description destructor of EEG
  arguments no arguments
  return value no return value.

- run
  description executes the Python script.
  arguments no arguments
  return value void

- stop
  description stops the Python script.
  arguments no arguments
  return value void

- connect
  description connects to the python script.
  arguments no arguments
  return value void

- startStream
  description starts the stream.
  arguments no arguments
  return value void

- stopStream

description	stops the stream.
arguments	no arguments
return value	void

- readData

description	reads data coming from the sensor.
arguments	no arguments
return value	void

Movinghead

Purpose:

The class Movinghead contains methods for controlling the Movinghead.

Collaborations:

As you can see from the static class model, Movinghead has one member of PyShellExec and one member of SocketClient.

Attributes:

- shell_exec
  description saves the shellexec object.
  type PyShellExec

- client
  description saves the socketclient object.
  type SocketClient

Operations:

- Movinghead
  description Constructor of Movinghead
  arguments no arguments
  return value no return value.

- ~Movinghead
  description destructor of Movinghead
  arguments no arguments
  return value no return value.

- run
  description executes the Python script.
  arguments no arguments
  return value void

- stop
  description stops the Python script.
  arguments no arguments
  return value void

- connect
  description connects to the python script.
  arguments no arguments
  return value void

- move_right
  description moves the Movinghead to the right.
  arguments no arguments
  return value void

- move_left
  description moves the Movinghead to the left.
  arguments no arguments
  return value void

- move_up
  description moves the Movinghead up.
  arguments no arguments
  return value void

- move_down
  description moves the Movinghead down.
  arguments no arguments
  return value void

- move_normal
  description the Movinghead goes to the starting position.
  arguments no arguments
  return value void

- move_set_step
  description the distance that the Movinghead moves in one step.
  arguments distance (int)
  return value void

ShimmersensorProt

Purpose:

The class ShimmersensorProt is an automatically generated class and defines the structure of the data to be transmitted and enables efficient serialization and deserialization of the sensor values in a cross-platform format.

Collaborations:

Attributes:

Operations:

EEGProt

Purpose:

The class EEGProt is an automatically generated class and defines the structure of the data to be transmitted and enables efficient serialization and deserialization of the EEG signals and associated metadata in a cross-platform format.

Collaborations:

Attributes:

Operations:

MovingheadProt

Purpose:

The class MovingheadProt is an automatically generated class and defines the structure of the data to be transmitted and enables efficient serialization and deserialization of the control commands and parametersin a cross-platform format.

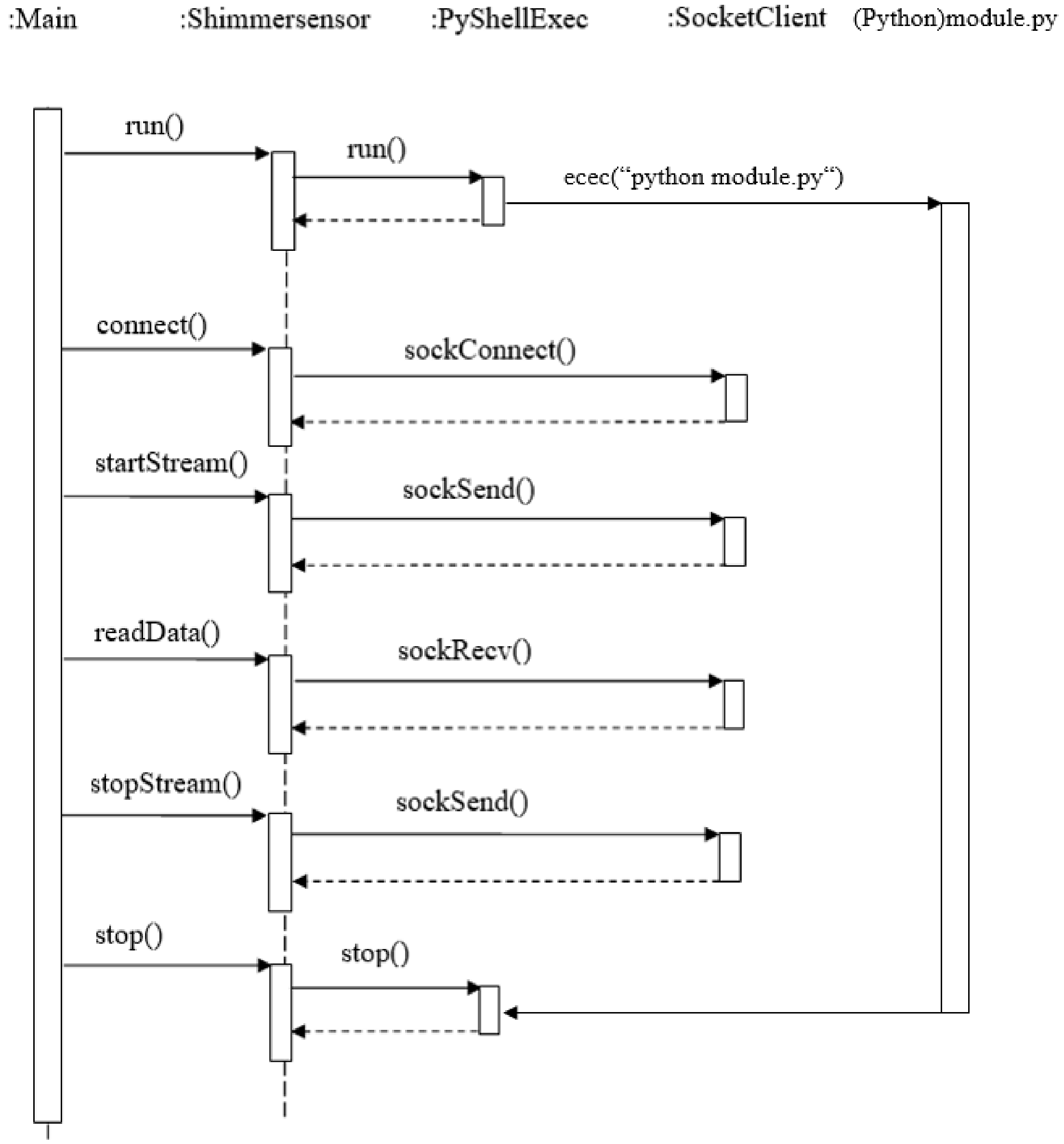
Collaborations:

Attributes:

Operations:

2.3 Dynamic Model

The sequence diagram below shows the order of method calls.



The sequence diagram shows that the main class calls the run() method. Then the Shimmersensor class calls the run() method of the PyShellExec class in order to launch the python script as an independent process using the Windows API. The process ends when the stop method terminates it. Now the Main class calls the connect() method. The Shimmersensor class calls then the sockConnect() method from the SocketClient class to connect to the socket. After that the Main class calls the startStream() method for starting the stream. Then the Shimmersensor class calls the sockSend() method to send the messages. Now the Main class calls the readData() method in order to read the data of the Shimmersensor. The Shimmersensor class calls the sockRecv() method to receive the data. After reading the data the Main class calls the stopStream() method to stop the stream and the Shimmersensor class calls the sockSend() method to send the messages. At the end the Main class calls the stop() method. Now the Shimmersensor class calls the stop() method from the PyShellExec class to stop the python script.