Detailed Design Report

By Tolle Sudore

Mehmet Çağdaş AKAR – 1630524

Çağrı ASLANBAŞ – 1630607

Uğur BALTACI - 1559962

Cem EKİCİ - 1560119

CEng491

01.09.2011

Table of Content

Table of Figures

1. INTRODUCTION

This Detailed Design Description document is about the project named “Controlling Mobile Devices via Gesture Recognition” (CMDGR) and is written by using the IEEE Std 1016-1998 model. In this document, the detailed design of the project is going to be described.

1.1. Problem Definition

Human Computer Interaction (HCI) is getting more and more important with the improvement of the technology. In the history, different perspectives inspired people to develop some innovative systems. Static keys and buttons, track path devices, touch and multi-touch screens have developed respectively. The next innovation should let people use computers, game consoles or mobile devices without touching in order to control those devices easily and effortlessly. Thus, gesture recognition and motion tracking are the next improvement of interaction with computers and mobile devices.

Controlling mobile devices is not an easy task while driving car or mobile device is located away from user. Touching screen or pushing button enforces the user to focus on the device even the user is doing another job. However, The Gesture Recognition System will be the solution of getting out of an obligation of touching the screen or pushing the buttons of mobile phones in order to make a call, control the menu, take a picture etc. By detecting the hand movements of a user, there will be no necessity of touching on mobile devices.

Some applications related with gesture recognition are already available for computers. However, current mobile devices’ processors are not capable of handling this work since almost most of the mobile devices have very less powerful processors compare with even netbooks. The future mobile devices, namely smart phones, will have more powerful “new generation” processors (like Intel Atom) in a short time.

Due to all of these reasons, Controlling Mobile Devices via Gesture Recognition (CMDGR) will be a breakthrough innovation of mobile devices market.

1.2. Purpose

In this document “Controlling Mobile Devices via Gesture Recognition” structure will be explained. Detailed design concepts of CMDGR will be shown in this document. The design architecture and procedure, constraints, interface design and implementation strategies will be explained in detailed. The required UML diagrams will be added in order that the detailed design description document will be more understandable with the necessity visuals. In other words the design concepts mentioned in Initial Design Document will be revised and corrected and the additional required information are put in this document.

1.3. Scope

The detailed versions of the architectural design, data design, procedural design, design constraints and development schedule which are stated in the initial design description document will be covered in this document. The decomposed components are covered and identified more detailed. Besides this, dependencies are clarified and stated one by one. Description of each module including type, purpose, functions and subordinates, interfaces, processing and data are given clearly and more detailed with respect to the ones in the initial design description document. Also, the user interface will be covered.

1.4. Overview

In this document, general description of the software will be mentioned as an overview at the second chapter. At chapter three, the design considerations such as dependencies, constraints and the design goals will be covered. Description of data as an object oriented concept and their dictionary will be shown at chapter four. System architecture of the software in modular way is the topic of fifth chapter. The architectural design and components of the system will be described in detailed at that chapter. At chapter six user interface design will be explained. Chapter seven contains detailed design of the project. The libraries and tools which will be used and benefited in order to develop the software are going to be mentioned in chapter eight. Time planning of the project is shown as a Gantt Chart model and the conclusion of this document will be shown in chapter nine and chapter ten respectively.

1.5. Definitions, Acronyms and Abbreviations

Term	Definition
Visual Depth Information	Proper visual recognition of depth or the relative distances to different objects in space
Gesture Recognition	Mathematical interpretation of a human gesture by a computing device
Stereovision	Method that the human visual system uses to perceive depth

SRS	Software Requirements Specification
CMPGR	Controlling Mobile Phones via Gesture Recognition
GRE	Gesture Recognition Engine
MeeGo	A Linux-based open source mobile operating system, implemented by Intel and Nokia
IDD	Initial Design Description
DDD	Detailed Design Description
OS	Operating System
ER	Entity - Relationship
UI	User Interface

1.6. References

- Software Requirements Specification of CMPGR, Tolle Sudore
- Initial Design Description Report of CMPGR, Tolle Sudore
- "SDD-ieeestd1016-1998", IEEE Recommended Practice for Software Design Descriptions
- M. Stamp, "A Revealing Introduction to Hidden Markov Models", 2004
- G. Welch, G.Bishop " An Introduction to the Kalman Filter", 2006

2. SYSTEM OVERVIEW

The main goal of the product is to track and recognize basic human gestures and process them to control a mobile device using stereovision concept. The functionality of CMDGR is to provide an interaction between the mobile device and its user without need to touch the device. Thus, mobile device users can control the devices when they are doing their daily activities. There are some activities that one has to use his hands while doing it, like driving car, cooking meal. The CMDGR offers a solution to people to control their mobile devices even when they are doing that kind of activities. The system will also work properly in some specific conditions that make people use their mobile devices difficult such as using the device with wet hands, with gloves or even walking.

Goals & Objectives:

- Reliability:

  CMPGR will be designed as a whole simulation of a mobile device. Since mobile devices are one of the most indispensable objects for people in their daily life, our system have to be designed with minimum faults. It will contain a dataset of various hand features with different types and colors in order to decrease the error tolerance of the system. False errors, which are the most dangerous errors for a system to work correctly, will not be occurred in the system.

- Low-cost:

  Implementation of the stereovision technology by using two low cost cameras provides an inexpensive solution with respect to the current technologies which are already available on the mobile device market.

- Maintainability:

  Any error occurred while the system is in use will be tolerated and system recovers itself and continues properly.

- Portability:

  The system will be implemented into the mobile devices which have Unix based operating systems. Also, two cameras and a powerful processor are needed for the system. So the software will be low-portable.

- Performance:

  Since the system will work on real time, performance is one of the most important topics of the system. Performance of the system will be high enough that user can use the system without noticeable delays or performance problems.

- Usability:

  User interface of the system will be designed as user friendly. Menus will easily be accessible and required short cuts are also implemented. Interface will be noncomplex that user can understand the software easily and can get used to the software well.

- Time:

  Software will work on real time. Response of the system after the user makes an action will be fast enough that does not cause a problem.

- Security:

  Since the system is compatible for only Unix based OS, it will be more secure for the external attacks.

3. DESIGN CONSIDERATIONS

The design considerations are the most important part of the detailed design description document since the product is a breakthrough innovation. The innovative concept behind the product comes with many assumptions, dependencies and especially constraints.

3.1. Design Assumptions, Dependencies and Constraints

3.1.1. Design Assumptions and Dependencies

Firstly, this project comes up with an idea to be used in mobile devices which have two cameras located in front. Obviously, this kind of technology is not available today therefore we are going to implement this project in personal computers with two cameras.

It is needed that the performance and time limits of the software will be applied considering that the software would be working on a mobile device.

It is assumed that this project will operate well on new generation mobile devices without any problems if the mobile devices have powerful processors and two cameras located in front.

3.1.2. Design Constraints

3.1.2.1. Hardware Constraints

The system needs a powerful processor in order to make complicated pattern recognition and stereo matching algorithms, like processing 6-7 frames per second. A powerful processor like Intel Atom will be sufficient.

System memory consumptions never exceed 20% of the total memory, approximately 200 MB.

Stereo cameras of systems can capture frames with 640x480 resolutions; furthermore, focal length of these should be between 1.2mm and 2.1mm. Most of the mobile devices ensure this constraint.

3.1.2.2. Software Constraints

When the hardware part of the system is completed, software should work on Unix kernel (the most recent release) in order to be capable of running on different platforms like MeeGo, iOS, Android, Symbian. Software architecture will be based on Model-View-Controller (MVC) architecture.

3.1.2.3. Performance Constraints

When the system is activated, system will be in a state that interaction between end-users will be the most frequent event; thus, as it is stated before system should process 6-7 frames/sec for a proper and error-free interaction.

3.1.2.4. Time and Financial Constraints

Both the design and implementation phases should be completed within eight months. In order to achieve this, the schedule specified in the requirement analysis report should be followed strictly.

Project is financially supported by Kade Bilişim A.Ş in terms of cameras, software tools and low-level PC processors like new generation mobile processors.

3.2. Design Goals and Guidelines

The highest priority of the software is to design the Gesture Recognition module. After the module is designed, Face Recognition module will be inherited and Navigation and Phone Call modules will be manipulated. Thus, design of Gesture Recognition module is the first main step of the software. Module descriptions are explained in detailed at the fifth chapter.

One of the most important goals of the software is to keep the memory use in reasonable degree. Since the aim of the software is the software can be implemented into mobile devices, memory use is one of the most critical points that the software will faced. The upper limit of the memory use will be 20% of the memory.

The other important goal is the speed of the software. The process of the software, which is recognition of the gesture, processing it and providing the action, will be done in at most 0.2 second in order that the software will work in reasonable time.

4. DATA DESIGN

4.1. Data Description

CMDGR system consists of two different components that are working together. Engine and Interface are the mentioned components.

At first the Engine part tracks a hand gesture and sends it to the interface. Interface translates the gesture to the appropriate command and if it is valid, the corresponding menu or submenu transition or the corresponding action will be done. The ER diagram of the perfect cooperation of these two parts, Engine and Interface can be seen on Figure 1.

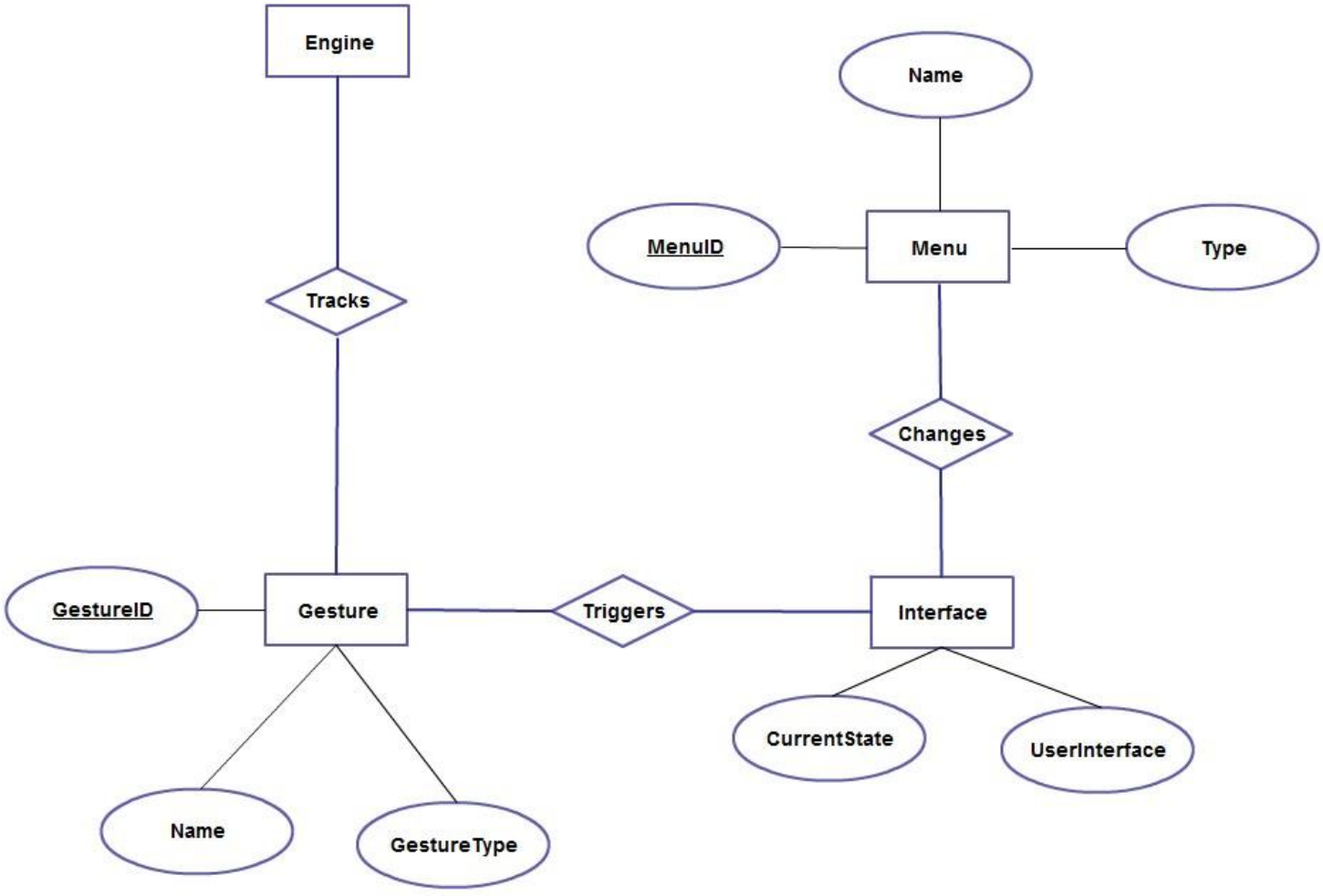

Figure 1 - Entity Relationship Diagram of System Entities

CMDGR system consists of two main data structures named Gesture and Menu. These data structures are created, modified or used by the two modules of the system, Engine and Interface.

Engine part of CMDGR system tracks a hand movement or gesture and creates a Gesture object belonging to it. Engine sends that gesture object to the interface and the current menu or submenu, reflected on the screen by the Interface module, will be changed according to that gesture.

Data structure of the menu and submenus will be tree structure. Main Menu will be the “root” and submenus of the Main Menu will be child nodes. Every non-leaf menu has a vector of child menus. On this vector the transitions are done if the current gesture object type is “Next” or “Previous”. Hence it is important in scroll-based menus.

CMDGR system will have two different types of menus: numerical menus and scroll-based menus. Numerical menus are type of menus which consists maximum 10 numbers of submenus. Thus, these types of menus can be controlled by users by making appropriate numbers with their fingers. When the hand movement, which refers to a number, is shown to the screen, Interface part of the system will go to the corresponding child of the current menu and directly select it. Scroll-based menus are type of menus that have more than 10 submenus. Contacts and Videos & Photos menus are some examples of scroll-based menus. Since it is impossible to control these menus by showing the appropriate number to the screen by using fingers, these menus will be controlled by special hand movements, which refer to up and down. After the recognition of the up or down movement of the hand by Engine part of the system, Interface part will decide on the action and will traverse on the menu items and stop when the user ends his/her hand movement. Data structures of CMDGR system and their attributes can be seen on Figure2.

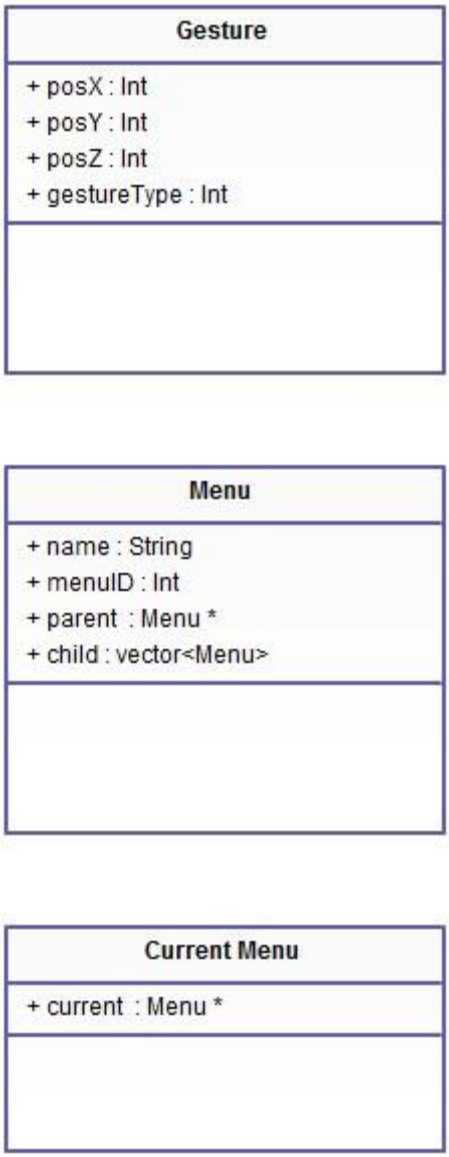

Figure 2 - Class Diagrams and Attributes of Main Data Structures of CMPGR

Interface module is the module that gets the gesture object from Engine module and decides the action corresponding to the gesture. Only one main data structure will work on the interface module: Current Menu Object. Current menu object only holds a pointer on the current menu shown on the screen.

In order to be more understandable, data flow diagram of an action that invokes Call function is shown in Figure 3.

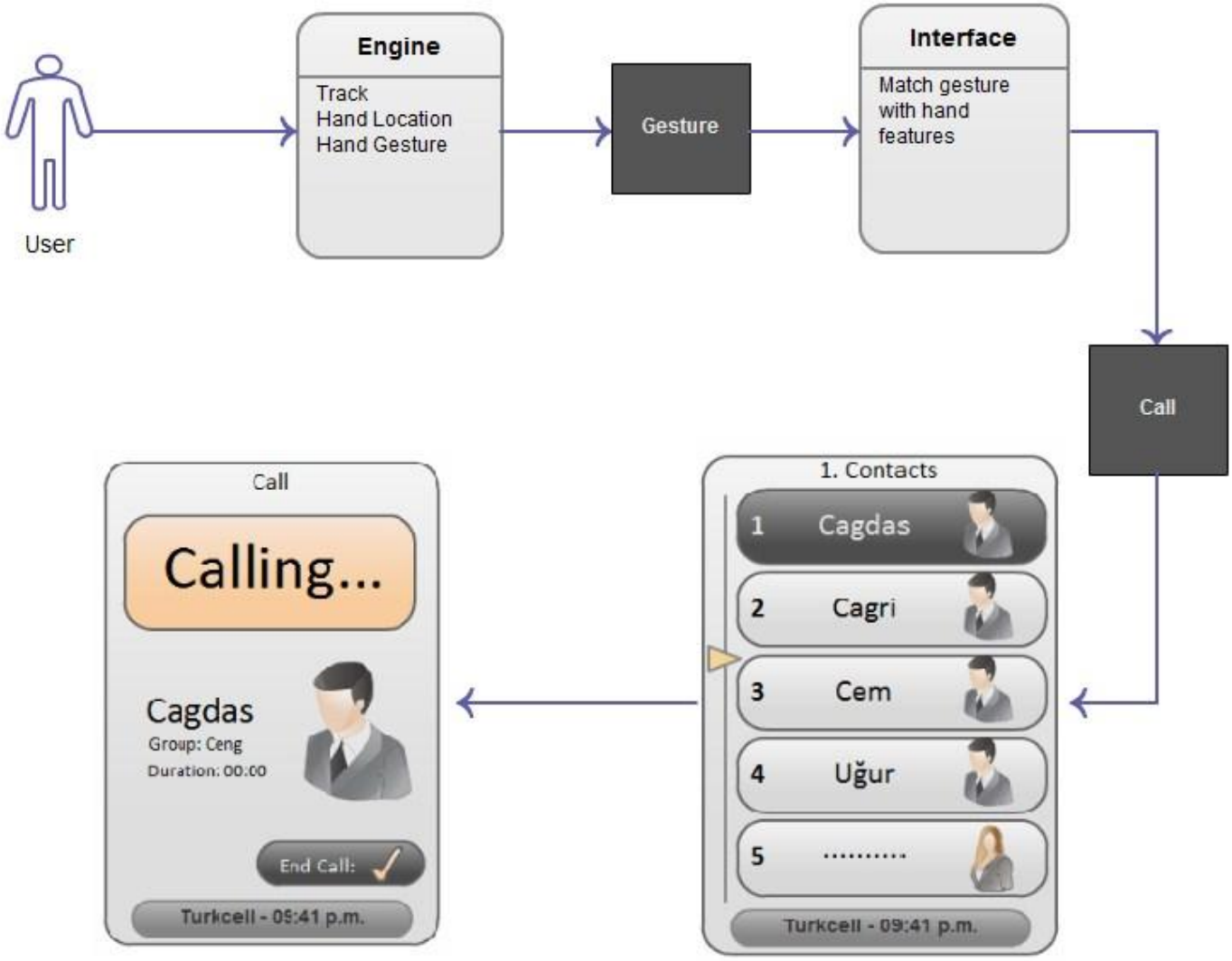

Figure 3 - Data Flow Diagram of Call Function

4.2. Data Dictionary

- **Face Recognition**

Attributes:

vector< int :feature> ;

map <int:face_id, vector:feature_vector>;

Methods and Parameters:

IplImage getFrames(void);

boolean faceRecognized(void);

void sendAuthentication(void);

- **Gesture Recognition**

Attributes:

vector< int :feature> ;

map <int:gesture_id, vector:feature_vector>;

int:initial_pos_x;

int:initial_pos_y;

int:initial_pos_z;

int:end_pos_x;

int:end_pos_y;

int:end_pos_z;

Methods and Parameters:

IplImage getFrames();

bool isValid(Gesture: gesture) ;

Gesture* createGesture(int: gesture_id),

void sendGesture(Gesture: gesture);

- **Navigation**

Attributes:

int*: state;

bool isScrollBased;

Methods and Parameter:s

bool unlockScreen(void);

void getGesture(Gesture: gesture);

int checkType(Gesture:gesture);

int* changeState(void);

void lockScreen(void);

void startCall(string:phone_number);

void endCall(Gesture:gesture);

- **PhoneCall**

Attributes:

string: phone_number;

Methods and Parameters:

string getNumber();

bool isEnd();

5. SYSTEM ARCHITECTURE

Description of system architecture is given below.

5.1. Architectural Design

Our system is consists of two main modules, namely, interface and engine. As its name suggests, interface is the user interface controller for user interaction such as quit, select, call etc. Engine is responsible for recognizing of hand movements and recognizing of faces. Once, valid gesture comes, engine interprets related action and triggers interface objects to do the action of the suggested gesture. In basic, interface is the subsystem which enables the user of the phone to give the freedom of control without touching phone and engine is the subsystem that is responsible for capturing, analyzing and detecting the desired action for the incoming hand movement.

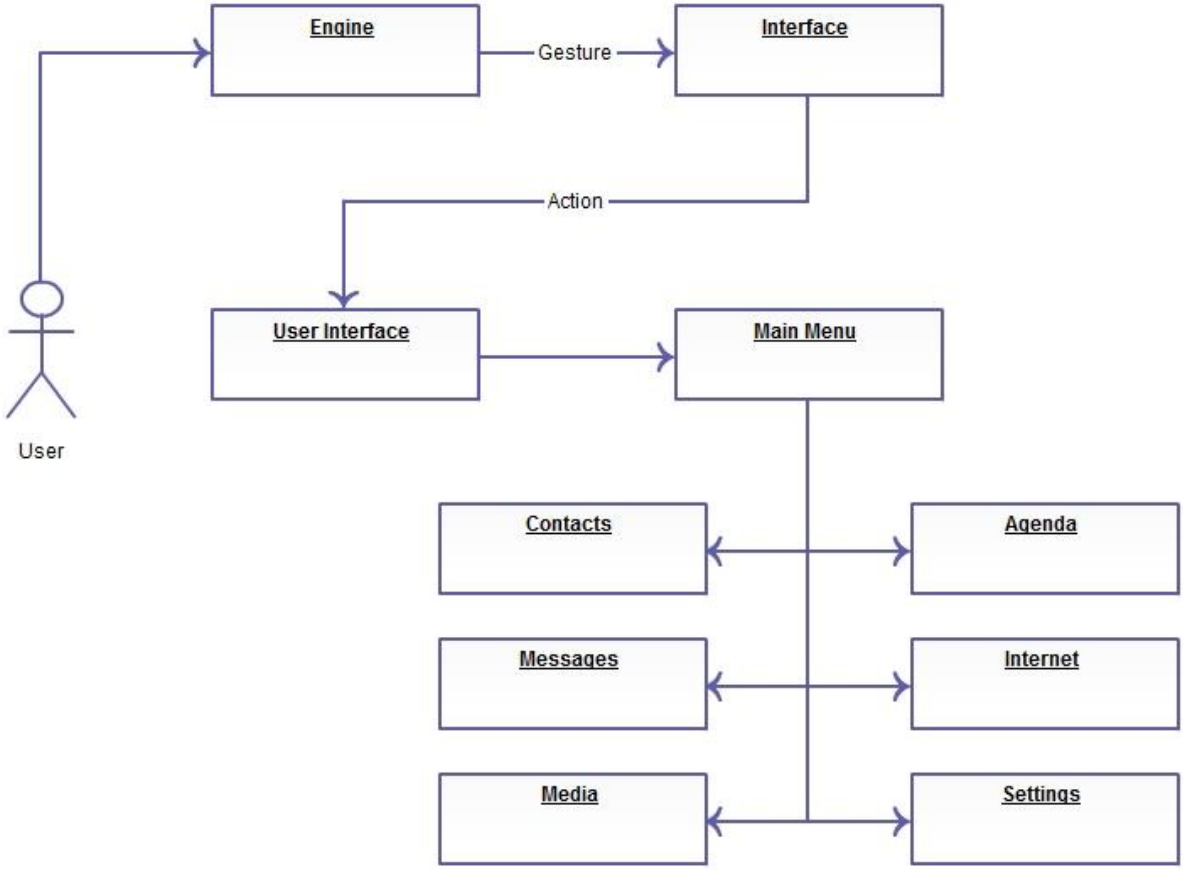

Figure 4 - Use Case Diagram of the System

5.2. Description of Components

In the upper section it's briefly said that CMPGR has two subsystems, namely Interface module and Engine module. Interface has been decomposed into two components. First component is Navigation and the second one is PhoneCall component. Moreover, Engine is the combination of two components, Face Recognition and Gesture Recognition. Basically, Navigation is the component that is responsible for changing the state of the menu and painting GUI, navigating through menus in the phone with the coming gesture input from Gesture Recognition. It also locks the screen until Face Recognition component reports back. The responsibility of Face Recognition is to capture consecutive frames and try to recognize the face of the holder of the phone. Also it is reported back to Navigation that the screen of the phone will be unlocked. Gesture Recognition gets the hand movements from camera and tries to interpret the gesture. If a gesture is found in pre-defined gesture data set, Navigation component is notified. Navigation looks for the gesture and takes in action the desired action. If 'call' gesture comes to Navigation, PhoneCall component is triggered by Navigation. PhoneCall is the component that is responsible for making phone calls.

5.2.1. Gesture Recognition

Gesture Recognition is the component that basically recognizes incoming hand movement from camera.

5.2.1.1. Processing Narrative

Two responsibilities of the Gesture Recognition component is to recognize gestures and create gesture objects and send it to Navigation component of the Interface module for further processing.

5.2.1.2. Gesture Recognition Interface Description

Frames that come from the camera and trigger from face recognition are the input to this component and output is the gesture object that is created after the recognition of the hand movement. Output of Gesture Recognition component is an input to the Navigation component.

5.2.1.3. Processing Detail

Firstly, features vector are extracted in the training phase - of course -, training phase are not related to software product. It is done once before product is released. Then, according to

feature vector, Bayes Decision Theory tries to recognize valid gestures from coming frames. Once gesture is recognized, then it creates a gesture object and passes it to interface package.

According to success rate and execution speed of Bayes Decision Theory, we may alter patter recognition algorithm to Hidden Markov Model. Also to speed up, we may use Kalman filter in the pre-process part.

5.2.1.4. Dynamic Behaviour

As seen in the figure below, Gesture Recognition gets the frames from camera and creates proper gestures if the user makes valid hand movements. In other case, that is user makes meaningless hand movements which does not exist in hand gesture data set, no gesture is created.

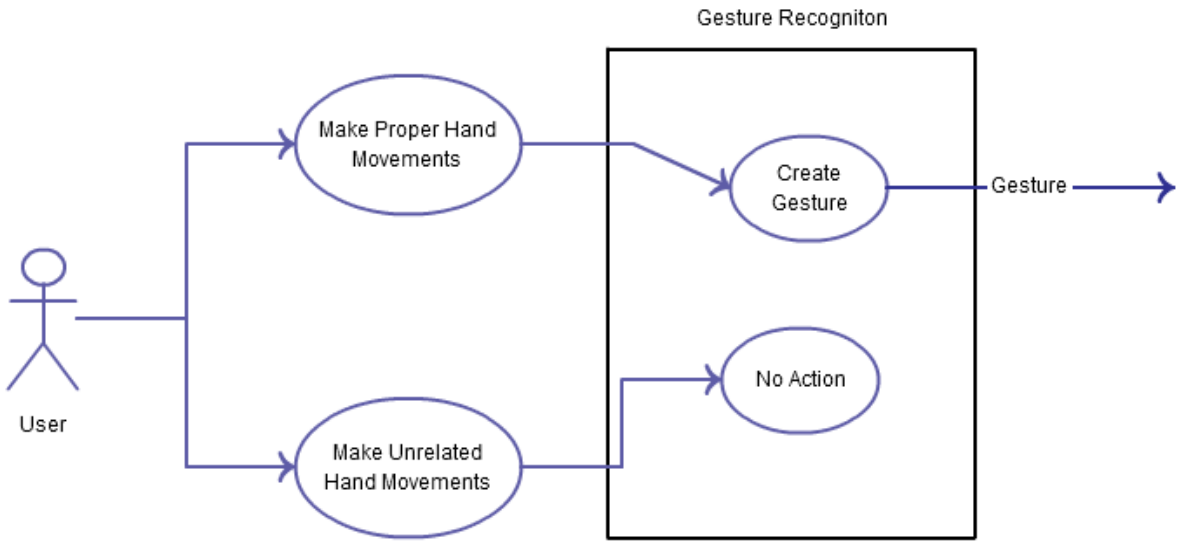

Figure 5 - Use Case Diagram of Gesture Recognition

After face is recognized by Face Recognition object, a new Gesture Recognition object is created and starts to get frames by camera using openCV library. Once it matches a valid gesture it creates a Gesture object and sends it to Navigation object. Sequence diagram for Gesture Recognition is given as below.

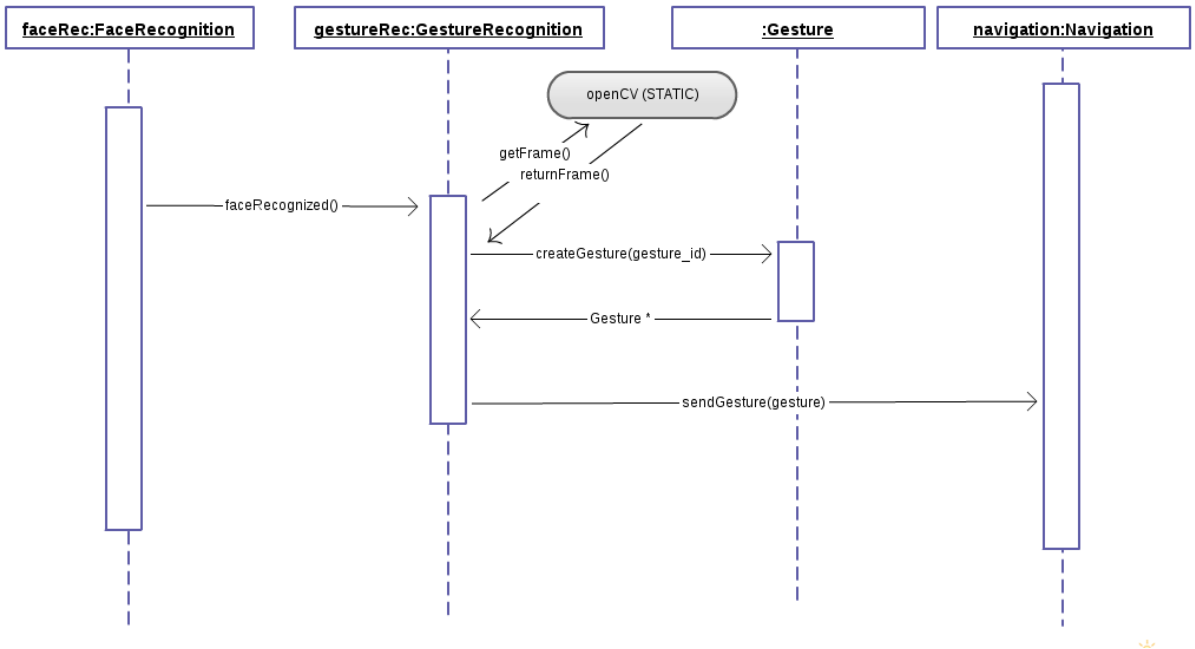

Figure 6 - Sequence Diagram of Gesture Recognition

5.2.2. Face Recognition

Face Recognition is the component that basically recognizes the face of the user of the system.

5.2.2.1. Processing Narrative

Two responsibilities of the Face Recognition component is to recognize the face of the user and send an authentication variable to Navigation component of the Interface module for unlocking the screen.

5.2.2.2. Interface Description

Frames that come from the camera are the input to this component and output is a signal that is sent to the Navigation component. Also it triggers gesture recognition component of engine module.

5.2.2.3. Processing Detail

Firstly, face detection algorithm of openCV are applied to retrieved frames. If face is detected then our specified face detection algorithm compares to features of detected face with features of user's face. If comparison inequality is lower than threshold, it recognizes as he/she is a user and authenticate to device.

5.2.2.4. Dynamic Behaviour

Actual user shows his or her face to the camera and Face Recognition component identifies the face and sends authentication signal to Navigation. If face of the user is not defined or stored as authenticated face in the system, face won't be recognized, and UI remains locked.

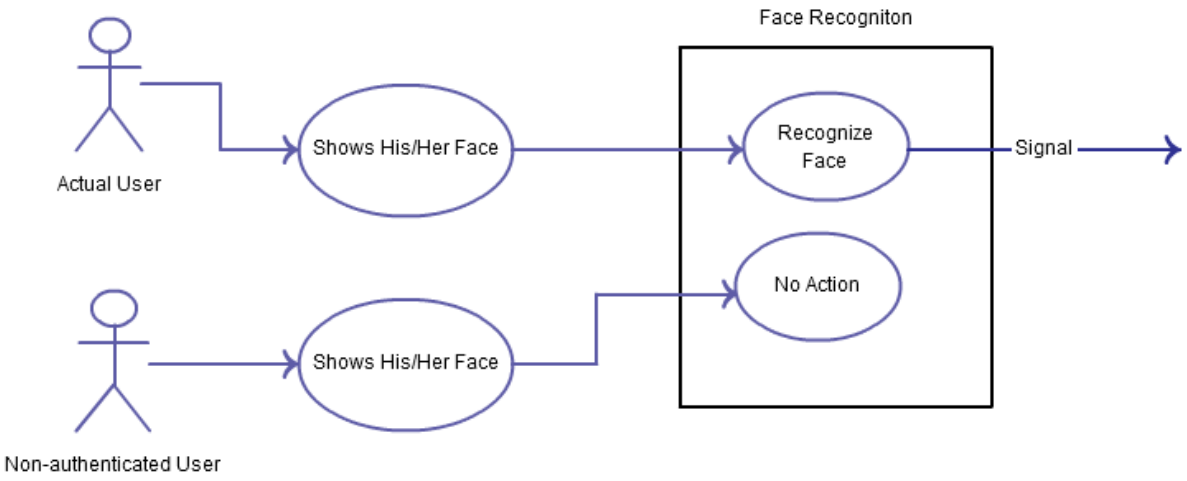

Figure 7 - Use Case Diagram of Face Recognition

Face Recognition object gets frames from camera using openCV library. As soon as it recognizes an authenticated face of a user, it sends authentication to Navigation. When lockScreen command comes from Navigation, it destroys the Gesture Recognition object that is created after successful face recognition.

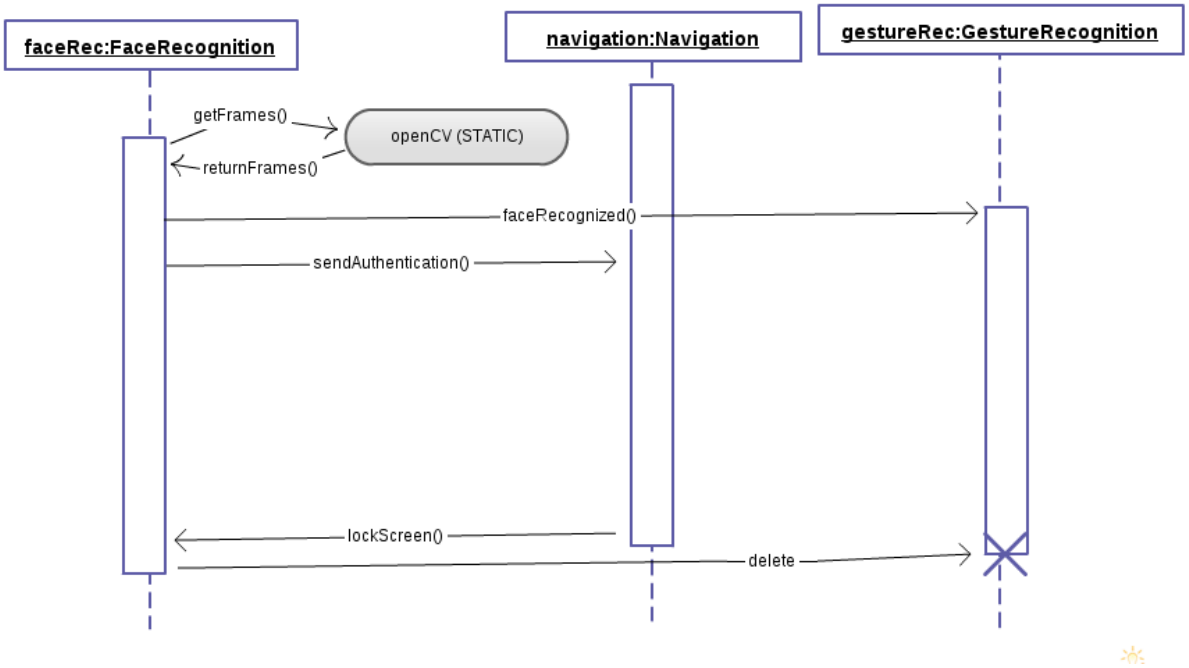

Figure 8 - Sequence Diagram of Face Recognition

5.2.3. Navigation

Navigation is the component that enables the user to navigate through the menus of the phone.

5.2.3.1. Processing Narrative

Navigation component is responsible for keeping the state of the user interface, namely the current menu to paint in the GUI. If the gesture comes from Gesture Recognition component is indicating a number and the current menu in the interface is a numbered menu, then it selects the corresponding menu and repaints the interface for selected menu. Furthermore, if the gesture is 'approve', then if UI is in an action that requires approval, Navigation component makes the action happen. It is also responsible for triggering the other component of Interface package, PhoneCall.

5.2.3.2. Interface Description

Inputs to this component comes from Gesture Recognition or Face Recognition component, output can be seen as the changes on UI. One more output is the phone number that is given to PhoneCall component.

5.2.3.3. Processing Detail

It is a simple process that tries to change UI according to given action of gesture id from engine component. If current state of UI allowed to the actions that are coming, it applies given action over UI.

5.2.3.4. Dynamic Behaviour

User makes hand movements and Navigation handles the proper action on the UI. Use case for Navigation is shown below.

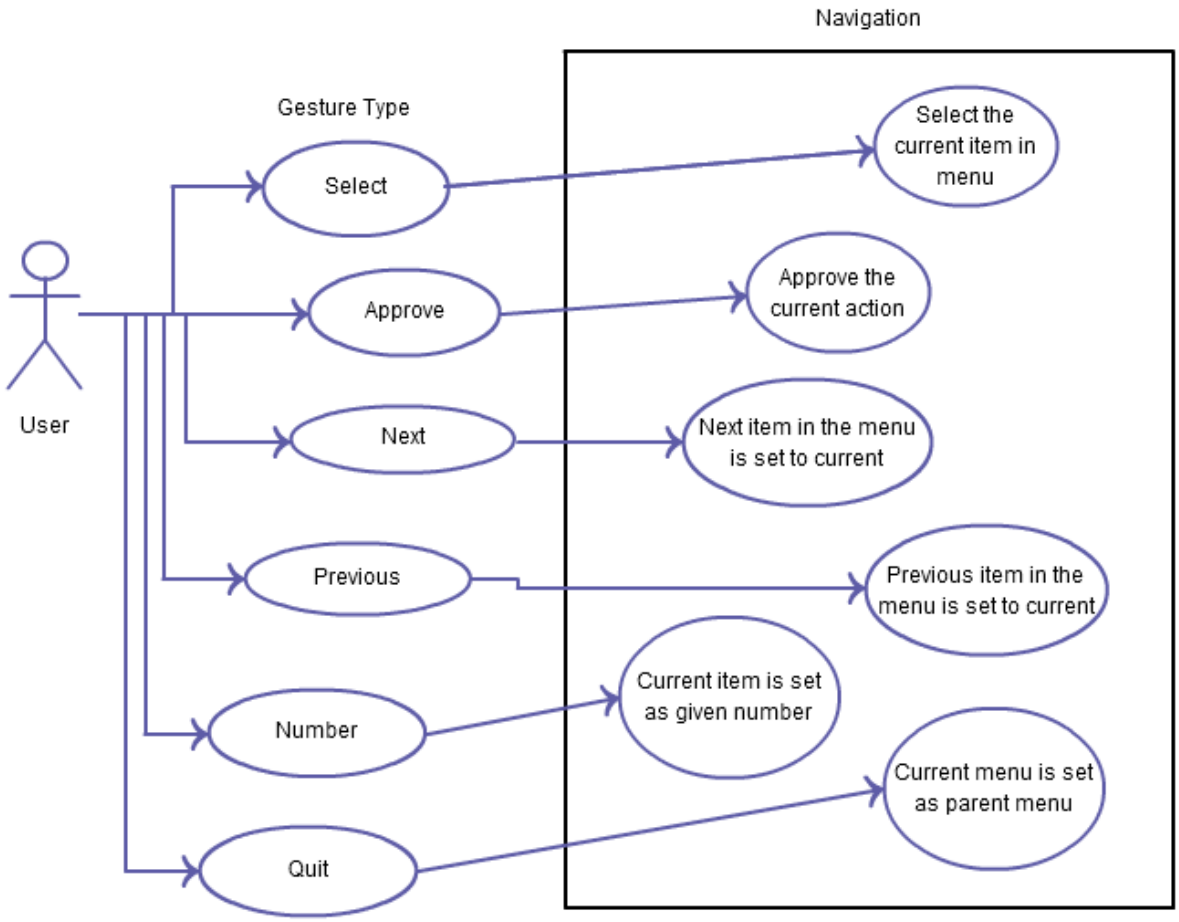

Figure 9 - Use Case Diagram of Navigation

When Navigation object gets the authentication signal, it unlocks the screen and repaints the UI.

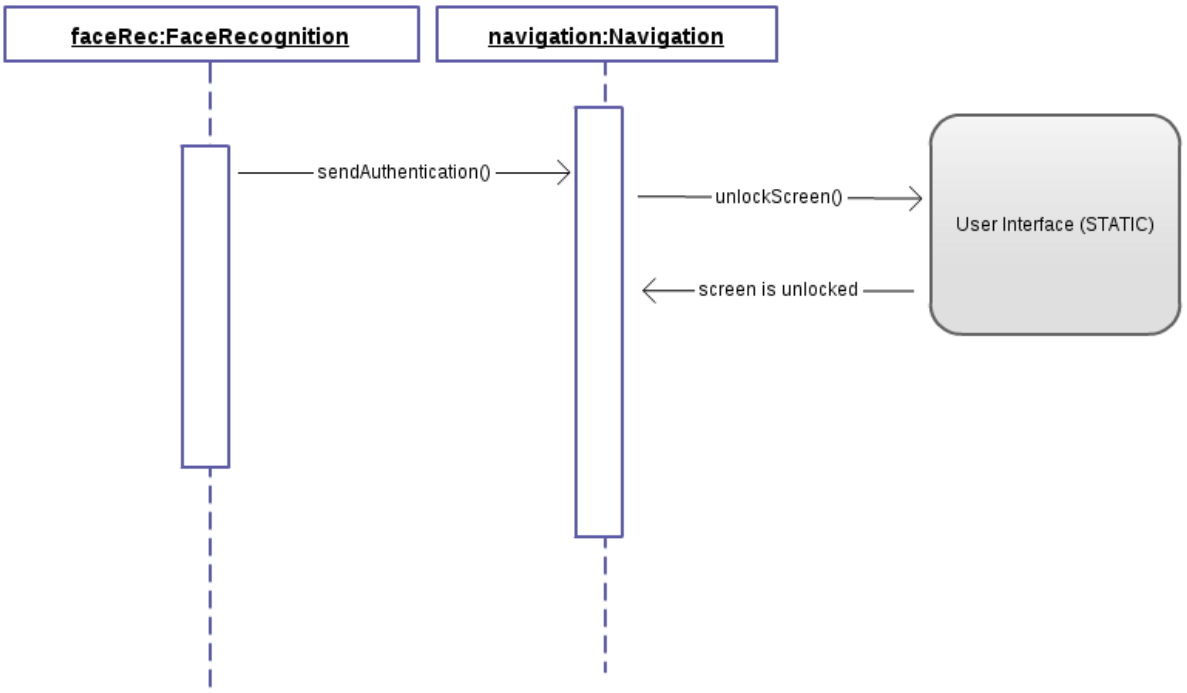

Figure 10 - Sequence Diagram 1 of Navigation

As seen in the figure below, gesture comes from Gesture Recognition object and its corresponding action is reflected on screen.

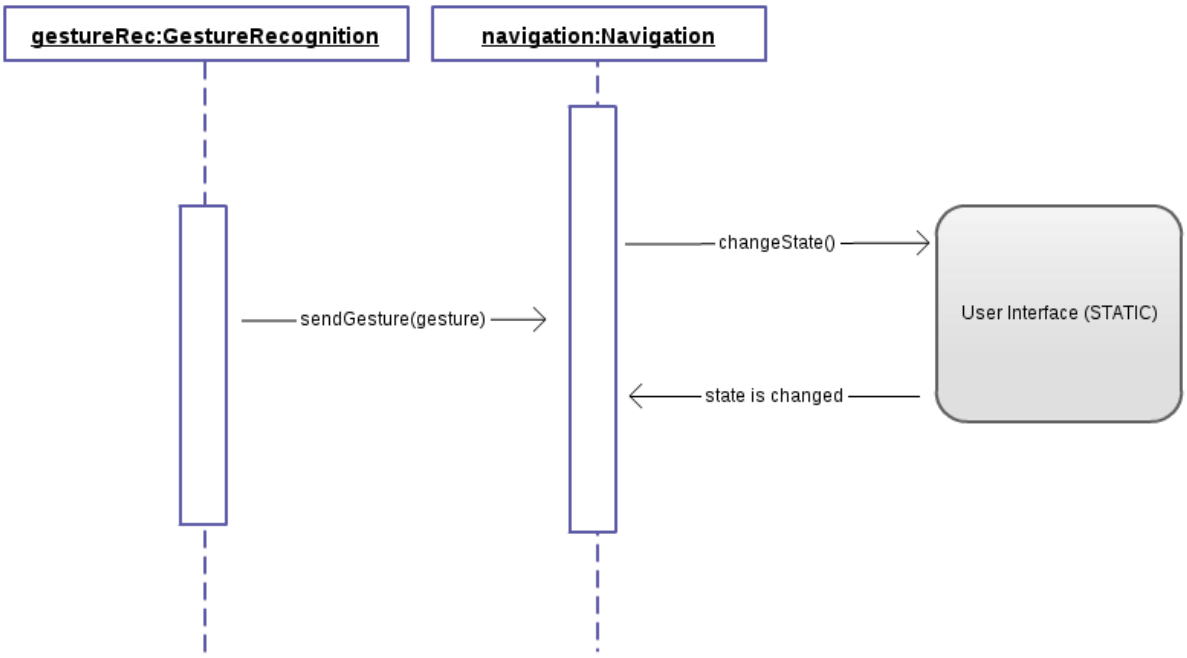

Figure 11 - Sequence Diagram 2 of Navigation

Use case for the “phone call” and “end call” gestures for Navigation component is shown below.

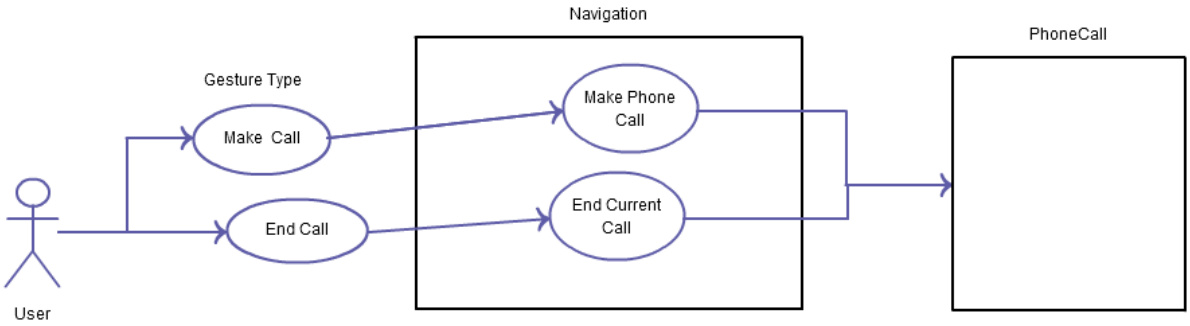

Figure 12 - Use Case Diagram of Navigation and PhoneCall

When phone call gesture comes from Gesture Recognition object, Navigation creates a PhoneCall object and changes UI correspondingly. If “end call” gesture arrives, phoneCall object is destroyed and call ends.

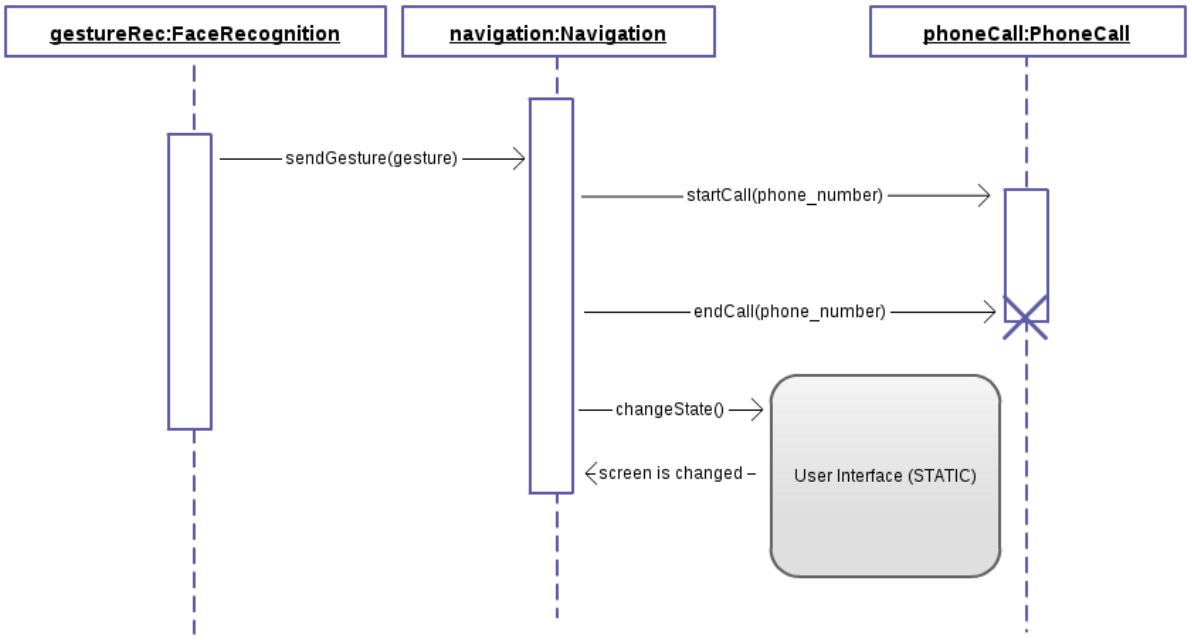

Figure 13 - Sequence Diagram 3 of Navigation

5.2.4. PhoneCall

PhoneCall is the component that handles the call operation to the desired contact number.

5.2.4.1. Processing Narrative

PhoneCall is responsible for making phone call and ensure that the phone call continues until signal, indicating the end of the call, from Navigation component comes.

5.2.4.2. Interface Description

The phone number comes from Navigation component of Interface package is the input to this component. Other input to end call is another input to this component that is also coming from Navigation. This component has no output.

5.2.4.3. Processing Detail

5.2.4.4. Dynamic Behaviour

Phone number to be called is given by navigation object. Then, a new PhoneCall object is created. This object handles the phone call operation until the endCall command comes from navigation. After that, PhoneCall object will be destroyed.

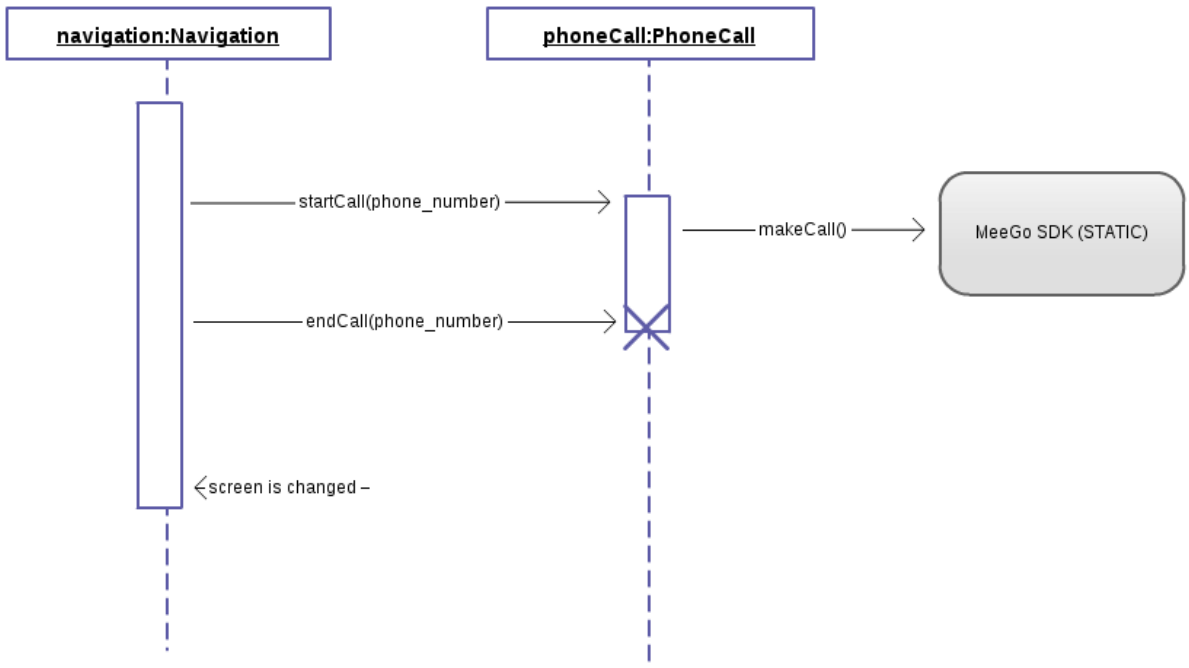

Figure 14 - Sequence Diagram of PhoneCall

5.3. Design Rationale

We have initially divided the whole system into two packages. The packages available in the system are Interface and Engine packages that they are able to work independently from each other. Since packages have the capability to work separately, the improvements on the Engine package can be incorporated in the system such that Interface package would not have to be changed. Furthermore, if a new User Interface is designed and decided to be included in the system, just changing the Interface package will be sufficient to keep the system functional without any change in Engine package is required.

Since Engine package has two vital responsibilities that needs different kinds of algorithms to overcome the recognition problem, recognition of the face and gesture is partitioned into two components. Thus, any changes in the algorithm of one component don't change the functionality of the other component.

Firstly, Interface package is considered not to be decomposed into components. But later on, Interface package is divided into two components, namely Navigation and PhoneCall, since the main purpose of the Interface package is to manipulate the UI and UI has two main operations that work independently.

6. USER INTERFACE DESIGN

6.1. Overview of User Interface

The aim of CMDGR system is to ease the use of mobile devices for their users. To do that, one of the most important issues is the simplicity of the user interface of the system. Transitions between menus have to be easy in order to increase functionality in specific situations like while driving the car, doing sports or activities like that, for the user.

Number of types of hand movements should be minimum to prevent confusion for the mobile device user. As it was mentioned in Software Requirements Specification of CMDGR, there are eight functions belonging to user interface. Except UnlockScreen function, the remaining seven functions have their own hand movement In order to take action. UnlockScreen goes into action by recognizing the face of the mobile device user in order to unlock the screen. Then the user finds herself or himself in the Main Menu. Except the Contacts and the Videos & Photos menu, the user selects the menu that he/she wants by showing the number of the corresponding menu by his/her hand. Select function takes action after doing this selection. In Contacts and Videos & Photos menu the user traverses the whole menu by rotating his/her fingers up or down. Next and Previous functions take action by making this hand movements. In all the menus and submenus, after selecting the right menu or submenu, the user makes a special hand movement in order to approve the selection. Here, Approve action goes into action. Quit function is used for returning to parent menu of the current submenu. Call and EndCall, by their special hand movements, is used for calling a contact, which is selected from Contacts menu, and ending the current call with a contact.

6.2. Screen Images

After unlocking screen by face recognition, the mobile device goes into Main Menu, where the user can select the appropriate menu by showing the number of the corresponding menu by using his/her fingers. In Fig-15 the user selects the submenu no.4, which is Media. The user has to show the special hand movement to approve that selection in order to go into Media submenu.

Figure 15 - Main Menu Interface

The Media submenu, which can be seen on Figure-16, has the same type with Main Menu. Desired submenu can be selected again by showing the correct number of the corresponding submenu to the screen. Approval movement should be made in order to pass to the submenu, which is Videos & Photos in the figure below.

Figure 16 - Interface of Media & Entertainment

Videos & Photos submenu is different from the menus or submenus above. To traverse in this submenu, the user should move his/her fingers up or down. These movements invoke Next and Previous functions. But like in the all menus or submenus, approval of the desired menu is made by the same hand movement. Videos & Photos submenu and scrolling menu system can be seen on Figure 17.

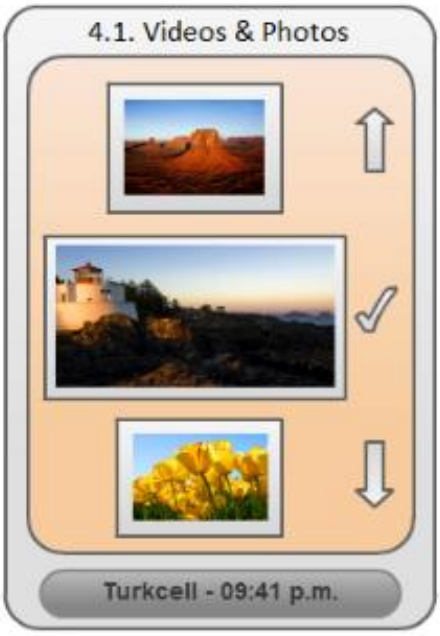

Figure 17 - Interface of Videos & Photos

There will be also a game in the mobile device, which is compatible with CMDGR system. The planned game will be a classic game, which can be played with left, right, up or down arrows, like Brick Breaker. The user interface of Brick Breaker game can be seen on Figure18. This game can be played by two simple finger movements which are the motions to left and right.

Figure 18 - Interface of a game

The two specific functions of CMDGR system are obviously Call and EndCall functions, which only take action while calling somebody or ending a call. Being in Contacts submenu is mandatory for these two functions in order to go into action. The Contacts submenu user interface can be seen on Figure19. Unlike the other submenus above (except Videos & Photos submenu), this submenu can be controlled by Next and Previous functions. Desired contact can be called by specific hand movement, which invokes Call function. In order to show the information of the current contact, Approve function can be called.

Figure 19 - Interface of Contacts Submenu

When a contact is chosen and called from Contact submenu, the user encounters the user interface below in Figure20. On this interface, the name of the corresponding contact, the duration of the current call, the group name that the contact belongs to, the picture of the contact, the status of the phone call (Dialing, Calling etc.) and the End Call selection will be included. As it appears from its name, EndCall function can be only invoked from this interface by using its special hand movement. EndCall function ends the current phone call and sends the user back to the contacts menu above.

Figure 20 - Interface of a Call Action

6.3. Screen Objects and Actions

CMDGR system simply consists of combination of two different screen objects, which are numerical submenus and scroll-based submenus. The most important difference between these two objects is the way that they are selected.

As mentioned in previous section the numerical submenus can be selected by showing its number with fingers, to the camera. Main Menu, Media submenu, Settings submenu, Agenda submenu are some examples of numerical submenus. The approval of these submenus are made by the approve movement.

Second type of screen objects, which is scroll-based submenus, can be controlled by the special hand movements, which refers to up and down. This type of menu is used where the number of submenus is over the amount of numbers that can be shown by ten fingers. Contacts submenu, Videos & Photos submenu, Music submenu are examples for scroll-based submenu, which usually consist of more than 10 entries. The approval of these submenus are again made by the approve movement.

7. DETAILED DESIGN

7.1. Face Recognition Component Detailed Design

7.1.1. Classification

Face recognition is a submodule of the Interface module.

7.1.2. Definition

Face recognition component is the submodule that basically deals with the recognition process of face of user. Basicly, it recognizes face of user and send appropriate signal to interface module.

7.1.3. Responsibilities

Face Recognition is responsible for detecting and validating of user's face. This component is also responsible for sending appropriate signal to interface module. Furthermore, it must have high success rate and it sends false signal when face of stranger is appeared from camera. Also, getting the frames from camera is the responsibility of this component.

7.1.4. Constraints

Face Recognition component should be implemented such that it deals with face recognition process in real time. Providing a real time recognition system is the most problematic issue that we have to deal with. Another constraint for the Gesture Recognition component is that recognition of face gestures should be independent of complex background. There is a limitation that only one face can be recognized that is provided simultaneously in front of the camera, namely if there are more than one face in front of the camera, only one of them will be taken into consideration. Also, when the software is initialized at the very beginning, user must recognize his face to the system via camera. This process may vary between 15 to 30 seconds. If the user changes his facial attributes such as growing a beard, or changing his haircut unusually, face recognition component may result wrong. For this case, user must recognize his face to the system again.

7.1.5. Compositions

There is a subcomponent that is responsible for sending related signal to Navigation component. This subcomponent must have separate responsibility since there can be synchronization problems. Therefore, it runs on separate thread that creates related signal coming from face recognition and sending it to navigation component.

7.1.6. Uses/Interactions

Face Recognition component affects the system's behavior. It indirectly affects other components, if face recognition does not come, then user interface still kept in lock position. Therefore; gesture recognition and -obviously- phone call component does not work. After recognition, navigation module starts gesture recognition process and allows changes on user interface. When one of these gesture types is recognized, Gesture object is created representing one of these predefined hand gestures.

7.1.7. Resources

OpenCV library is needed for this module to retrieve frames from camera. Operations, manipulations on frames are all done by functions in openCV library. Also, face features of user is another resource.

7.1.8. Processing

Face Recognition component gets frames from camera with *IplImage getFrames( )* function. Face Recognition component includes some pre-defined complex pattern recognition algorithms in openCV face recognition part. Recognition process starts with extraction of features vectors that have been trained before as we discussed in constraints. According to feature vector extracted, implemented algorithm of Bayes Decision Theory tries to recognize face of user. If a valid face, decided by *bool isRecognize()* function call, is recognized its corresponding signal is created with *unlock()* function call and passed to Navigation component with *sendUnlock()* function call.

7.1.9. Interface/Exports

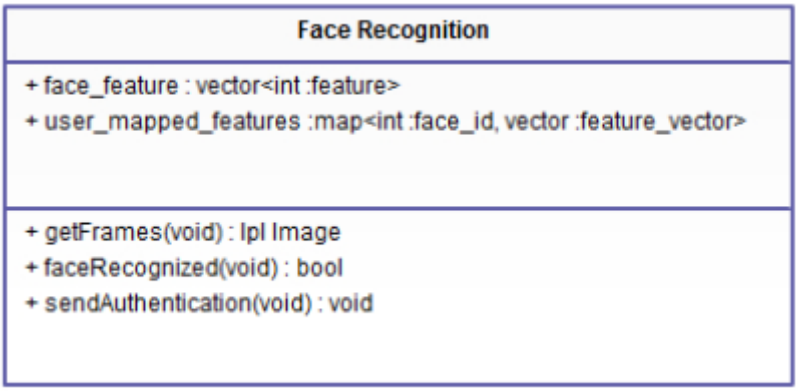

Figure 21 - Class Diagram of Face Recognition

As seen in the class diagram of Face Recognition component, only possible data type provided to other components is signal (can be seen on Figure 21) which is sent to Navigation component. Other data types included in class diagram are not visible to other components and these are available only in Face Recognition component. Recognizing face is sent to Navigation component via sendUnlock() function call. Thus, Face Recognition component has interface with one component namely Navigation.

7.2. Gesture Recognition Component Detailed Design

7.2.1. Classification

Gesture Recognition is a submodule of the Engine module.

7.2.2. Definition

Gesture Recognition component is the submodule that basically deals with the recognition process of hand gestures of the user of the system. It is the heart of our system since functionality of the system is available only if this component works properly.

7.2.3. Responsibilities

Gesture Recognition is responsible for detecting hand gestures and recognizing them. This component is also responsible for creating corresponding gesture objects for recognized gesture, sending to Navigation component in order to trigger Navigation for desired action. Getting the frames from camera is the responsibility of this component.

7.2.4. Constraints

Gesture Recognition component should be implemented such that it deals with gesture recognition process in real time. Providing a real time recognition system is the most problematic issue that we have to deal with. Another constraint for the Gesture Recognition component is that recognition of hand gestures should be independent of complex background. There is a limitation that only one hand gesture can be recognized that is provided simultaneously in front of the camera, namely if there are more than one hand gesture in front of the camera, only one of them will be taken into consideration. There should be at least one second between two hand gestures that are provided to system as input. This means that recognition of one hand gesture, whether it is applicable or not according to state of the interface in Navigation component, lasts one second.

7.2.5. Compositions

There is a subcomponent that is responsible for sending created gestures to Navigation component. Since trying to get a gesture before this component creates and sends Gesture object results in failure, sending and getting Gesture objects might have synchronization problem. This is handled by separating the sending operation of Gesture Object. This separate responsibility is given to a thread and that thread creates a Gesture object which is protected by a mutex and increases the mutex by one.

7.2.6. Uses/Interactions

Gesture Recognition component affects the system's behavior. It indirectly changes the UI, input to Navigation component is provided as a Gesture object so that interface of the system is changed. Predefined hand gestures are Select, Approve, Quit, Next, Previous, Call, EndCall and

Number. When one of these gesture types is recognized, Gesture object is created representing one of these predefined hand gestures.

7.2.7. Resources

OpenCV library is needed for this module to retrieve frames from camera. Operations, manipulations on frames are all done by functions in openCV library. Another resource for this component is the hand data set that is used for training.

7.2.8. Processing

Gesture Recognition component gets frames from camera with *IplImage getFrames( )* function. Gesture Recognition component includes some complex pattern recognition algorithms like Bayes Decision Theory, Kalman Filtering. Recognition process starts with extraction of features vectors that have been trained before. According to feature vector extracted, implemented algorithm of Bayes Decision Theory tries to recognize predefined valid gestures. If a valid gesture, decided by *bool isValid(Gesture: gesture)* function call, is recognized its corresponding object is created with *createGesture(int gesture)* function call and passed to Navigation component with *sendGesture( )* function call.

7.2.9. Interface/Exports

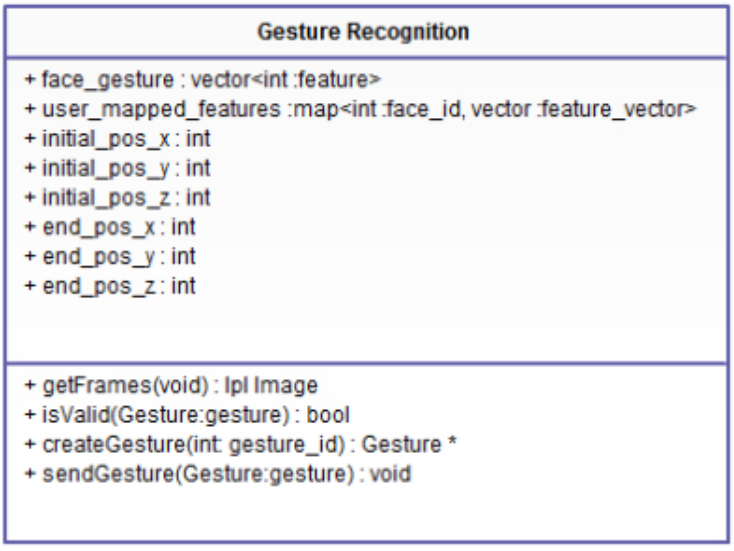

Figure 22 - Class Diagram of Gesture Recognition

As seen in the class diagram(Figure 22) of Gesture Recognition component, only possible data type provided to other components is Gesture object (can be seen on Figure 2) which is sent to Navigation component. Other data types included in class diagram are not visible to other components and these are available only in Gesture Recognition component. Created Gesture object is sent to Navigation component via sendGesture() function call. And this component is triggered by authentication signal of Face Recognition component. Thus, Gesture Recognition component has interface with two components, namely Navigation and Face Recognition.

7.3. Navigation Component Detailed Design

7.3.1. Classification

Navigation is a sub-module of the Interface module.

7.3.2. Definition

Navigation, as it appears from its name, is a component of CMDGR that provides navigations between menus of the mobile device. To explain it briefly, navigation makes the connection between the gestures and the user interface. Various gestures, which are responsible for “Select”, “Approve”, and “Call” etc., change the user interface by the action of Navigation component.

7.3.3. Responsibilities

Main menu and their sub-menus are connected to each other by tree structure. In order to change the current menu/sub-menu by the corresponding gesture that is given by the Gesture component, Navigation component must know the current state of the user interface (which menu/sub-menu does the user interface is in). Hence Navigation component’s major responsibility is to keep the current state of the user interface, in order to work properly. Another important responsibility of Navigation component is to keep the information that the current menu is a numbered or scroll-based menu. Except the authentication that is given to Navigation component from Face Recognition component, Navigation component takes all of the gestures from Gesture component. This component takes a gesture from Gesture component

and first looks if the gesture is compatible with the current state and type of the menu/sub-menu. If the current gesture is not compatible with the type of the current menu, for example number gesture for a scroll-based menu, the Navigation component cannot work properly. In situations, when the gesture and the current menu state and type is compatible, Navigation component paints a button or changes the menu/sub-menu that is been viewed, of the user interface.

Moreover, Navigation component is also responsible for to trigger the PhoneCall component when the gesture “Call” is given from Gesture component when the user interface is on “Contacts” sub-menu and the corresponding person that is desired to call is selected.

7.3.4. Constraints

In fact, Navigation component has the capacity to change the user interface expeditiously. But Gesture component tracks a gesture in 1 second. Since the Navigation component waits for Gesture component to get a gesture and then change the user interface by using the corresponding gesture, the tracking time of the Gesture component retards the action of the Navigate component. Likewise this deceleration also affects the PhoneCall component.

7.3.5. Compositions

Navigation component consists of three sub-components, which are in-menu action, between-menu action and synchronization sub-component, respectively. First two sub-components both navigate the menus/sub-menus but they differ in the type of affect that they provide on the user interface. As it appears from its name, in-menu action is a sub-component that is active when the gesture that is given from Gesture component, is a type of gesture that does not change neither the current menu/submenu nor the current state of the Navigation component. “Select”, “Previous” and “Next” are type of gestures that activate in-menu action.

Between-menu action is a sub-component that is active when the gesture that is given from Gesture component, is a type of gesture that changes the current menu/submenu and the current state of the Navigation component. “Call”, “Approve” and “EndCall” are some example type of gestures that activate between-menu action.

Third sub-component, synchronization is responsible for receiving created gestures from Gesture component. Since trying to receive a gesture from Gesture component before Gesture component creates and sends Gesture object results in failure, sending and getting Gesture objects might have synchronization problem. This is handled by separating the sending operation of Gesture Object. This separate responsibility is given to a thread and that thread creates a Gesture object which is protected by a mutex and increases the mutex by one.

7.3.6. Uses/Interactions

Navigation component is the only component of the CMDGR system that has interaction with all of the components of the system. These components are Face Recognition component, Gesture component and PhoneCall component, respectively.

The interaction of Navigation component and Face Recognition component is only active when the user unlocks the screen by using face recognition. Here, Face Recognition component sends the authentication to the Navigation component, even if the face that recognized is not valid. Navigation component unlocks the screen and changes the user interface to the main menu if the authentication is successful, else gives a warning to the user stating that the authentication is not valid.

The interaction of Navigation component and Gesture component is very extensive. The gestures are sent to Navigation component from Gesture component and Navigation component changes the user interface by the corresponding gesture. Actions “Select”, “Approve”, “Quit”, “Next”, “Previous”, “Call” and “EndCall” are the type of actions, which are given from Gesture component and activate Navigation component and consequently change the user interface.

Lastly, the interaction between Navigation component and PhoneCall component is only on action when the user selects the corresponding contact and makes the hand movement that invokes “Call” function. In other words, PhoneCall component is activated when the gesture that is given from Gesture component to Navigation component is “Call”. Being in the “Contacts” menu and selecting a contact is mandatory for the CMDGR system to connect Navigation component to PhoneCall component.

7.3.7. Resources

Since Navigation component is strongly related with the user interface, Qt Library is the most important resource of this component and its sub-components.

7.3.8. Processing

Since Navigation component can be both activated by Face Recognition component and Gesture component, there are two different scenarios for the processing of Navigation component.

First scenario is the interaction between Face Recognition component and Navigation component. Navigation component gets the authentication from the Face Recognition component and if the authentication is valid, Navigation component calls *bool unlockScreen( )* function and then changes the current state with *int* changeState( )* function.

Second scenario is about Gesture component and Navigation component. By *void getGesture(Gesture: gesture)* function, Navigation component gets the gesture from Gesture component. Right after that operation, Navigation component checks the type of the current gesture by using *int checkType(Gesture:gesture)* function. If the type of the current gesture is "Call", Navigation component calls *void startCall(string:phone_number)* and *int* changeState( )*, respectively. This situation also invokes the PhoneCall component. If the current gesture is "EndCall", by the same way, Navigation component calls *void endCall(Gesture:gesture)* and *int* changeState( )*, respectively. If the type of the current gesture is "Select", "Approve", "Quit", "Next" or "Previous", Navigation component directly activates *int* changeState( )* function.

If the user does not make any action, Navigation component automatically calls *void lockScreen( )* function in order to lock the screen.

7.3.9. Interface/Exports

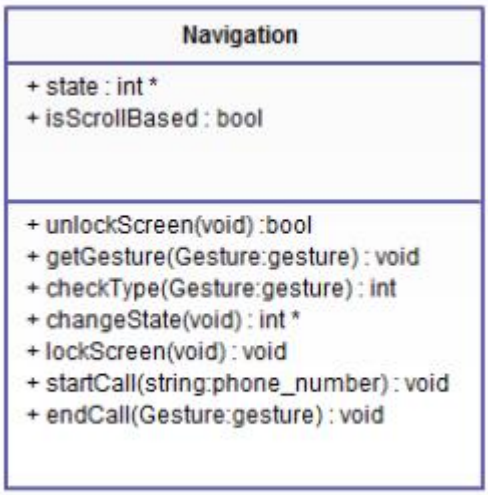

Figure 23 - Class Diagram of Navigation

As seen in the class diagram of Navigation component (Figure23), the only data that is provided by Navigation component is the phone_number variable. When the Navigation component gets the “Call” gesture (Being in the “Contacts” menu and selecting a contact is mandatory for the CMDGR system to connect Navigation component to PhoneCall component.), Navigation component changes the user interface and invokes PhoneCall component in order to success call operation. Navigation component makes this activation by providing the phone_number data.

7.4. PhoneCall Component Detailed Design

7.4.1. Classification

PhoneCall is a sub-module of the Interface module.

7.4.2. Definition

PhoneCall component, as it appears from its name, is a component of CMDGR system which is activated when a receiver is tried to be called from “Contacts” sub-menu. The main obligation of this component to take action is being in “Contacts” sub-menu and selecting a person from this sub-menu.

7.4.3. Responsibilities

PhoneCall component is responsible for handling phone calls to corresponding receiver and ensuring the phone calls to continue until “EndCall” function invoked, indicating the end of the call, from Navigation component.

Another major responsibility of PhoneCall component is to be in interaction with the corresponding mobile operator, consistently, because making a phone call is impossible when the mobile device is unable to get signal from the mobile operator’s base station.

7.4.4. Constraints

PhoneCall component is strictly related with the mobile operator that the mobile device uses. When PhoneCall component receives a signal from Navigation component meaning that the user invoked the function “Call”, PhoneCall component tries to call the corresponding user no matter what. Since it is impossible to call a person without any mobile operator signal from base station,

PhoneCall component cannot work properly. This situation is same for all mobile devices in the world.

7.4.5. Composition

There is no subcomponent for PhoneCall component since this component is only invoked with a unique gesture and its responsibility is not complicated.

7.4.6. Uses/Interactions

The only interaction of PhoneCall component is with Navigation component. This interaction only appears when the system user selects a person to call from “Contacts” submenu, by making the appropriate hand movement. The gesture is received by the Navigation component and after the type of the gesture is checked, PhoneCall component is invoked.

7.4.7. Resources

MeeGo SDK libraries are used in order to accomplish PhoneCall tasks.

7.4.8. Processing

When the system user selects a person to call from ”Contacts” submenu, Navigation component sends the phone_number variable of the corresponding receiver to the PhoneCall component. PhoneCall component receives this variable by using *string getNumber( ) function.*PhoneCall component then initializes MeeGo SDK and invokes the appropriate function, which generates the call. As mentioned in Navigation component part, Navigation component uses *void endCall(Gesture:gesture)* function to notify PhoneCall component in order to end the current call, when desired. PhoneCall component ends the call using MeeGo SDK function.

7.4.9. Interface/Exports

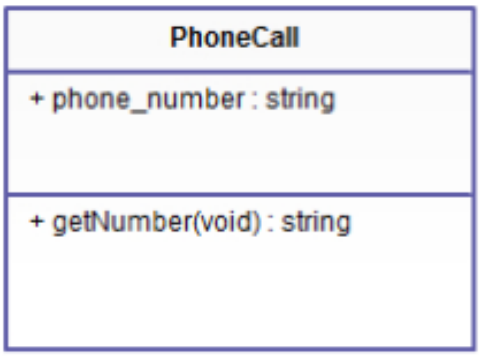

Figure 24 - Class Diagram of PhoneCall

As seen in the class diagram of PhoneCall component (Figure24), there are no possible data types provided to other components.

8. LIBRARIES & TOOLS

As it is stated before, the project consists of two main parts, namely, engine and interface. They show differences both in integrated development environment and in used libraries, also both are development in **C++** language and compiled and run in latest version of **UNIX** kernel. Since we use C++, we will strictly follow object-oriented design concepts.

Firstly, in the engine part, we will use **NetBeans** as an IDE. For thread operations, we will use **pthread**, **openCV**, **gstreamer** and Standard C++ libraries. Most important library - of course - is openCV, which is a powerful computer vision library and most of latest developed vision algorithms comes pre-defined within it.

Secondly, in the interface part, we will use Standard C++, Qt libraries and Qt Creator as an IDE. QT is a cross-platform application and UI framework. For mobile process such as calling, sending messages, we will use MeeGo SDK.

9. TIME PLANNING (GANTT CHART)

9.1. Term 1 Gantt Chart

Gantt Chart of Term 1 can be seen on Figure 21.

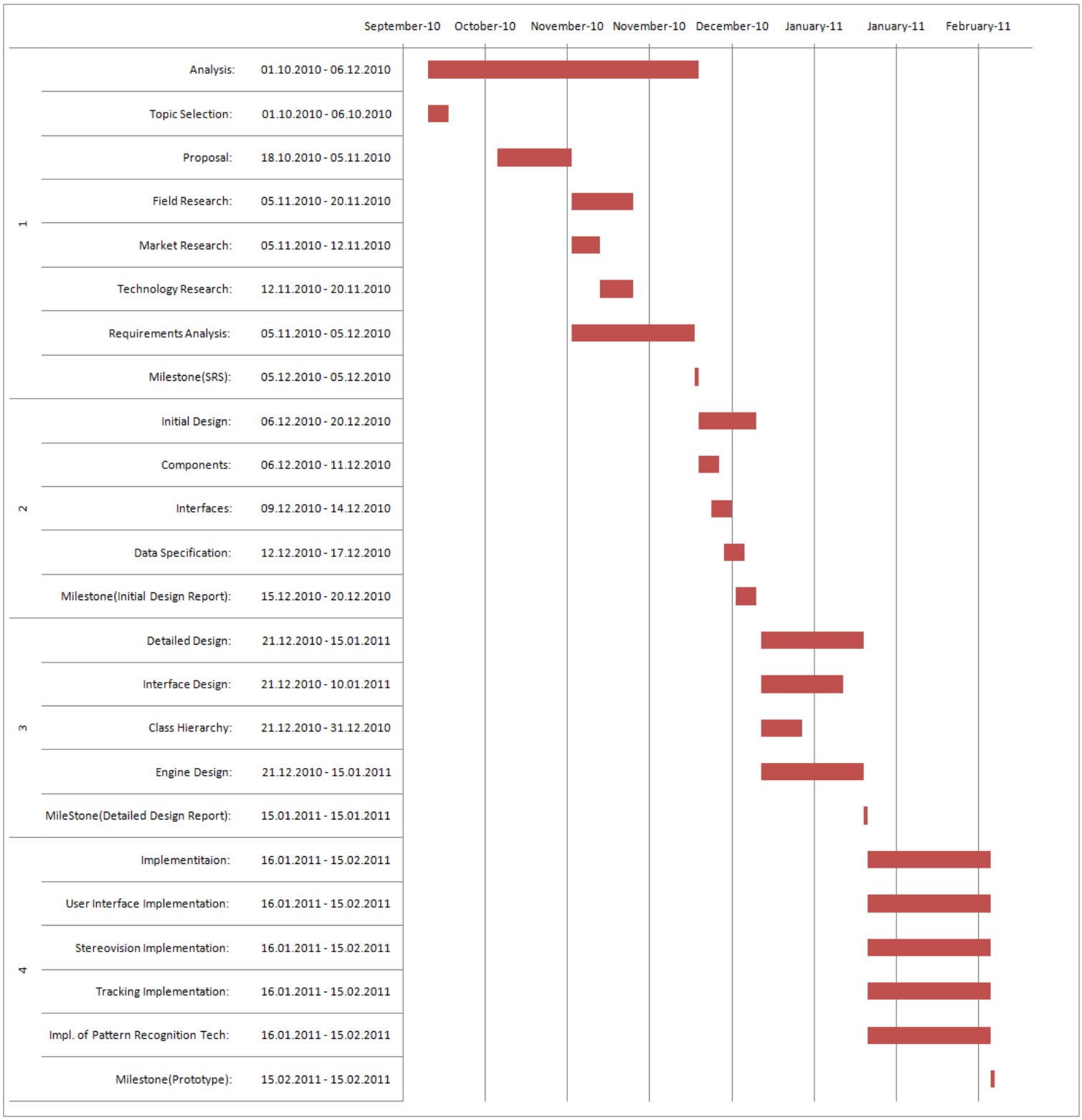

Figure 25 - Gantt Chart of Term 1

9.2. Term 2 Gantt Chart

Gantt Chart of Term 2 can be seen on Figure 22.

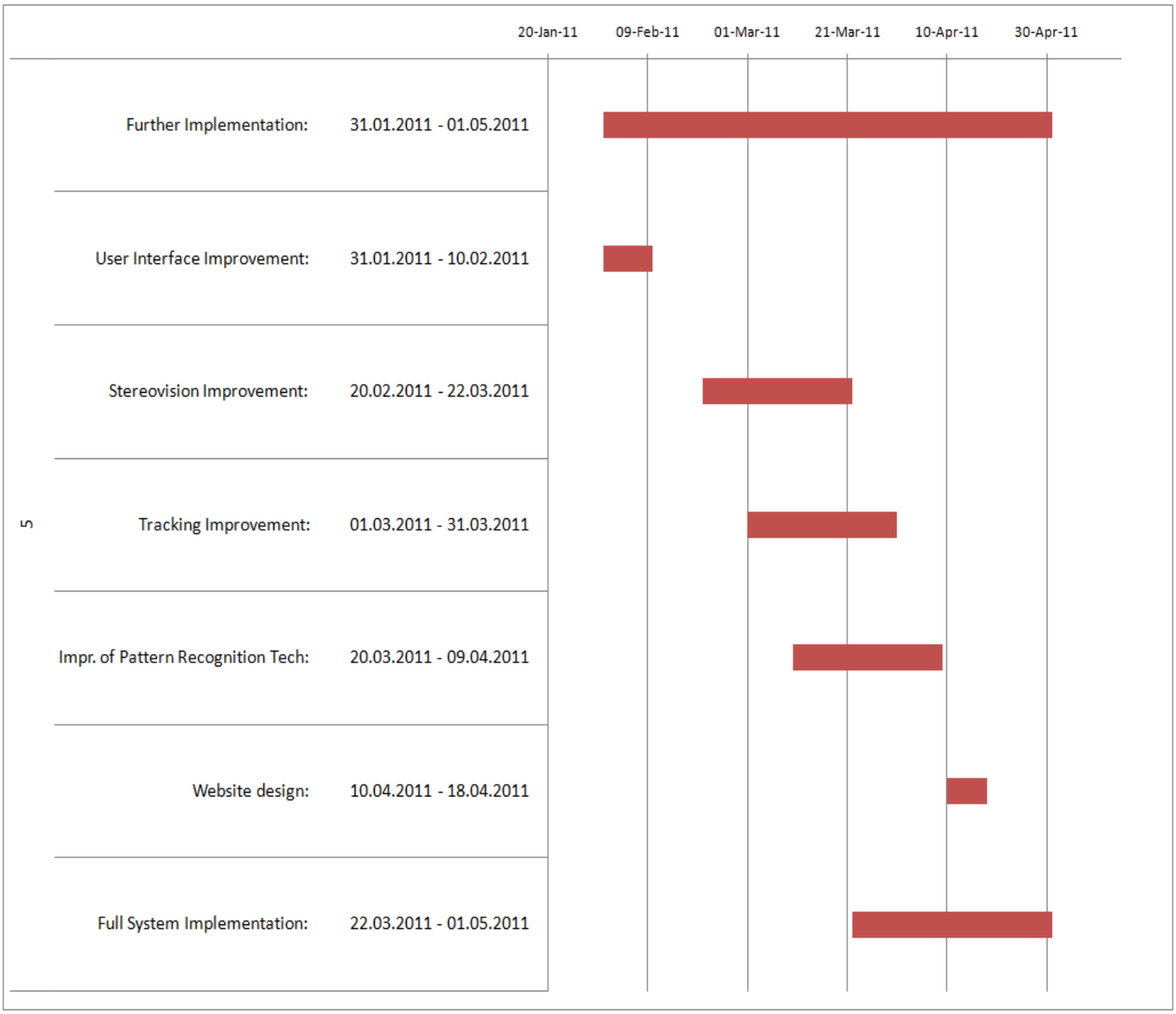

Figure 26 - Gantt Chart of Term 2

10. CONCLUSION

In conclusion, Detailed Design Report of the Controlling Mobile Phone via Gesture Recognition gives the definition, purpose and scope of the project. The detailed design is explained and the constraints that it possibly faced are clarified. Data structures and the architecture of the system are explained and corresponding design issues are stated with UML diagrams that the Detailed Design will be more understandable.

After the user interface is visually illustrated, the goal of the final product is clearer and more apparent.

The works that are done so far provides a point of view on the design process, so the schedule is added to the document in order to show the timetable for the future fork to be done.

Preparing this report was an important milestone in our schedule, since it will provide a common understanding of the expected outcomes of the application, components of the system, and the responsibilities of team members.

With the help of this document, programming part of the project can be started in near future. No confusion will be faced if strict observance of the document is pursued while programming part.