MIDDLE EAST TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

CENG491 DETAILED DESIGN REPORT

KORSAN YAZILIM

Table of Contents

1. Introduction

This document contains the overall system architecture and data architecture of “Context Aware User Interface Project” of group “Korsan Yazılım” and indicates descriptions of functions and specifications of this project in a detailed way which will be performed by senior METU Computer Engineering students.

1.1 Problem Definition

In the present day, mobile devices are commonly used by most of the people in any kind of environment and situation. Despite the fact that uses of these devices are so widespread, unfortunately, screens of mobile devices are not legible when we consider some difficult environmental cases. Therefore; we decide that we will develop a system which identifies the environment and state changes and then adapts the graphical user interface to provide convenience to users accordingly in these extreme situations. This system will use images taken by camera of mobile devices as inputs and will process them in order to determine how user interface will change itself for usability.

We decided to customize this problem to military problem because people who generally have to deal with these bad environmental situations we speak of are generally soldiers and we implement this project by sponsorship of Aselsan. We form our problem around a soldier who is a team leader who needs to get and accomplish missions.

1.2 Purpose

Aim of this report is definition of design issues of the overall system architecture and data architecture in a detailed way. Audience for this report is Aselsan (our sponsor for this project) and teaching members of course of Computer Engineering Design.

1.3 Scope

This detailed design document contains design information about Yaver which is a demonstration of architecture, modules, classes, use cases, functions, features, database model, graphical user interface, tools and special technologies. It describes how Yaver works

properly and with safety in detailed and understandable way. The architecture is intended as the basis for more complex and useful versions in the future.

1.4 Overview

The rest of DDR, in appearance order, contains pages generally involve design considerations, data design, system architecture, user interface design, detailed design, libraries and tools we will most probably use, planning of these and finally conclusion.

1.5 Definitions, Acronyms and Abbreviations

DDR: Detailed Design Report

DFD: Data Flow Diagram

GUI: Graphical User Interface

API: Application Programming Interface

P.K: Primary Key

DB: Database

JDBC: Java Database Connectivity

ODBC: Open Database Connectivity

1.6 References

[1] IEEE Std 1016-1998: IEEE Recommended Practice for Software Design Descriptions
[2] Java Database Connectivity(JDBC) :
http://en.wikipedia.org/wiki/Java_Database_Connectivity
[3] Java Swing:
http://en.wikipedia.org/wiki/Swing_%28Java%29
[4] Java Advanced Imaging API (JAI) :
http://en.wikipedia.org/wiki/Java_Advanced_Imaging
[5] Java Abstract Window Toolkit (AWT) library fundamentals
http://java.sun.com/developer/onlineTraining/awt/

[6] Java Media Framework API
http://en.wikipedia.org/wiki/Java_Media_Framework
[7] Google Map API official web site
http://code.google.com/intl/tr-TR/apis/maps/index.html
[8] Spiral Software Developing Model
http://en.wikipedia.org/wiki/Software_development_process

2. System Overview

As we said, Yaver is the application for helping a team leader to find his way in the map and accomplish his missions. In order to provide this, firstly, there will be a map screen which can show where user is, what mission route is and information about surroundings. In addition, there will be a section for taking missions and reading details of them and a section for message communication handling between command center and team leader. Most importantly, our application will adapt changes in environment by changing GUI such as; when user move fast, application will increase size of important sections to make them more visible. User will be able to determine which sections are important in certain environmental cases. We will develop a system which works in mobile devices to determine environmental conditions and to adapt user interface for these cases. Camera is the sensor which provides us to recognize environmental changes. This is a real-time system, since the system should determine environmental conditions immediately and change the user interface according to them. Additionally, we will try to make GUI as functional, understandable and user friendly as possible, because people who will use application must not be suffering from lack of usability of Yaver. We will design it to provide that they must be able to use it in a way that they can find anything they expect from application and adaptation of its' GUI to environment works perfectly, as much as we can.

3. Design Considerations

3.1 Design Assumptions, Dependencies and Constraints

We will design, implement and test our application in our laptops, because we are not given any mobile devices to work on, so we assume that the work we did for computer will be implementable in mobile devices too although there may be some incompatibility. We

haven’t done any image processing related work in our summer practices, so we will be learning and implementing during the process of doing the project at the same time. Therefore; as we learn, we might change our design and ideas about the project in order to make the application better. On the other hand, we have two general constraints:

3.1.1 Time Constraint

We have and will have other lessons and we will need to spare time for them too, which is not easy during the process of implementing our project. In order to handle this constraint, we will try to obey our plan as much as we can which is shown by using Gantt Chart in Time Planning section.

3.1.2 Performance Constraint

Since we are trying to implement an application for mobile devices, we should try to accelerate how our application operates due to fact that mobile devices have limited memory and process capabilities.

3.2 Design Goals and Guidelines

We aim to use memory as less as possible because this project is for mobile devices, but by doing that we will not let quality and correctness of application decrease. We will try to keep this rate at optimum level. Our other goals are:

3.2.1 Security

Network communication should be made in encrypted way in order to avoid network attacks in the system. Therefore, assignments and messages are sent after data is encrypted. This system guarantees that any information will not be stolen by unauthorized people.

3.2.2 Maintainability

The application should be banned for modification by users and allowed for extensions by developers. It should be implemented in a way that modification will not be necessary. However, it should be open for extensions such as keeping database for the past locations of the users, newer network connection technologies or its simulations, newer communication technologies. In other words, our system should accept common interface for all service it uses, the new ones should implement these interfaces in order not to modify previously coded parts.

3.2.3 Portability

The application should be platform independent. It should work in Windows and Linux successfully and also in mobile devices. By using Java for implementing this application we handle most of this issue.

3.2.4 Availability/Reliability

The application should represent all the information which comes from server in the map, message and assignment sections instantly, allowing dynamic changes whenever it executes. User and command center should be available to communicate with command center. Application guarantees that wrong information comes from users will not cause software crashes.

3.2.5 Scalability

Since available network management algorithms are designed to work on fixed or relatively small wireless networks and our communication technology has not been decided yet the exact number can vary. Our application is for using of one military team leader. However, this system is not an application such that many users cannot connect to device simultaneously. Application is suitable for all soldiers who are commanded.

4. Data Design

4.1 Data Description

This section shows attributes of data objects, and relationship between data and functions by using diagrams and tables. These data objects are made under the consideration of getting rid of unnecessary attributes and normalization factors.

4.1.1 Complete ER Diagram

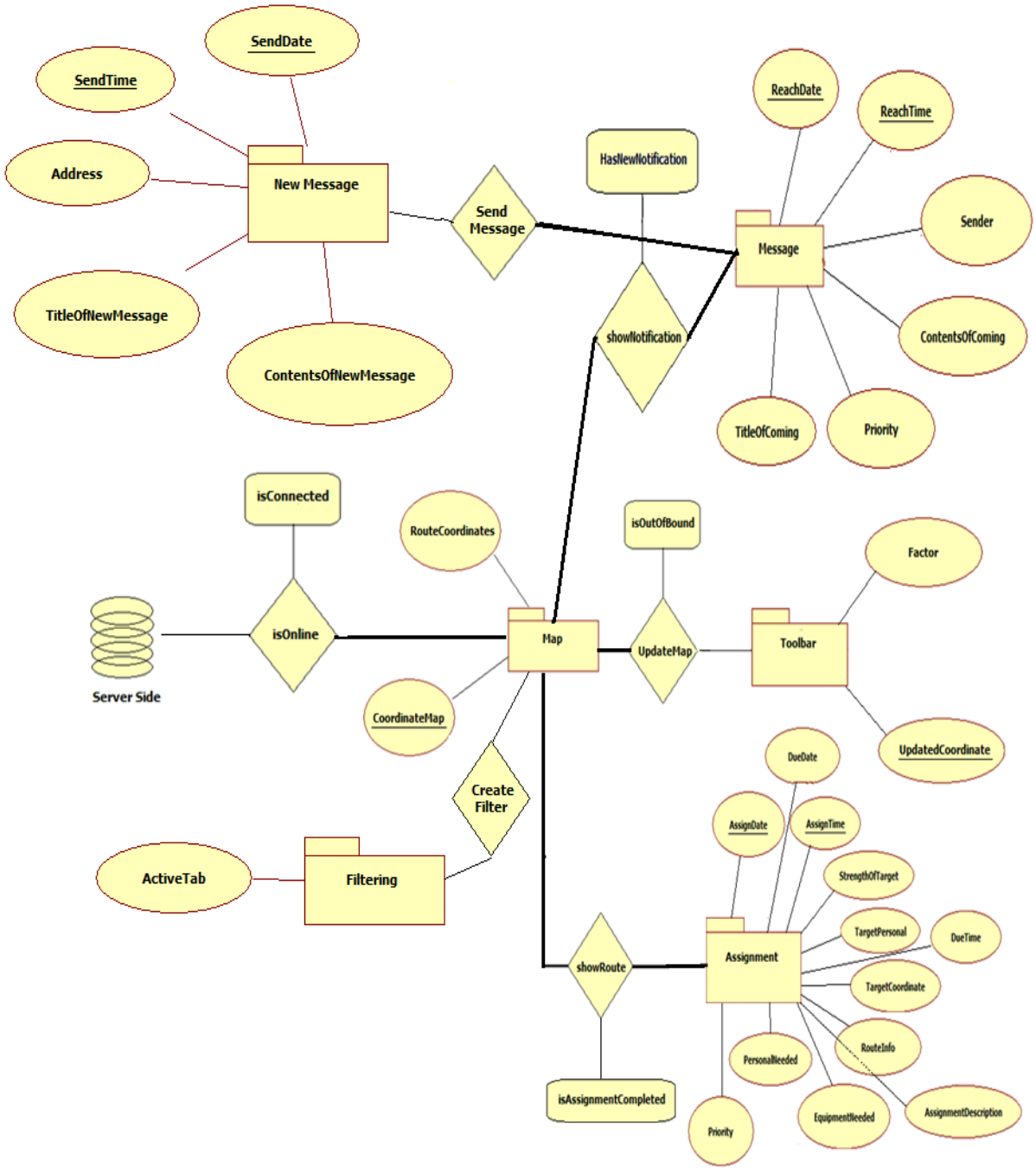

Figure 1: Complete ER Diagram

4.1.2 Data Objects

4.1.2.1 Map Data

This table is heart of the YAVER. Mainly, it shows the map related to the coordinate values and route coordinates of the assignments. This part is connected to the server to take map by using given coordinates. Detailed information will be seen at section 6.1.3.

Field	Type	Null	Foreign Key	References
CoordinateMap (P.K)	Varchar(24)	No	No	No
RouteCoordinates	Varchar(24)	No	No	No

Table 1: Yaver's Map Data

4.1.2.2 Toolbar Data

Users can change coordinate values by using move left, move right, move up, move down. Therefore, this new coordinate values are set coordinate of Map objects. Similarly users can change visibility amount of map by using zoom in, zoom out features. In order to change this feature, Factor can be incremented or decremented with special values with respect to the zoom in or zoom out. Detailed information will be seen at section 6.1.2.

Field	Type	Null	Foreign Key	References
UpdatedCoordinate (P.K)	Varchar(24)	No	No	No
Factor	Integer	No	No	No

Table 2: Yaver's Toolbar Data

4.1.2.3 Message Data

It's so similar to general purpose mail program. Simply, it provides connection between command centre and user. Coming messages are listed and user chooses whatever s/he wants. These messages are stored at internal database. Users can delete message from database using

Message interface. Coming messages are listed with respect to the priority by default. Detailed information will be seen at section 6.1.4.

Field	Type	Null	Foreign Key	References
ReachDate (P.K)	DateTime	No	No	No
ReachTime	DateTime	No	No	No
Sender	Varchar(100)	No	No	No
TitleOfComing	Varchar(100)	No	No	No
Priority	Integer	No	No	No
ContentsOfComing	Varchar(500)	No	No	No

Table 3: Yaver's Message Data

4.1.2.4 Assignment Data

One of the YAVER's facilities is showing the assignment assigned by command centre. When a duty is assigned, then user sees properties of this assignment by using this application. Each assigned assignments are listed. User can change order of the listed assignments with respect to the entities. Priority is default entity to order all uncompleted assignments. Detailed information will be seen at section 6.1.5.

Field	Type	Null	Foreign Key	References
AssignDate (P.K)	DateTime	No	No	No
AssignTime (P.K)	DateTime	No	No	No
DueDate	DateTime	No	No	No
DueTime	DateTime	No	No	No
AssignmentDescription	Varchar(500)	No	No	No
Priority	Integer	No	No	No
PersonalNeeded	Integer	No	No	No
EquipmentNeeded	Varchar(100)	No	No	No
RouteInfo	Varchar(500)	No	No	No
TargetCoordinate	Varchar(24)	No	No	No
TargetPersonal	Integer	No	No	No
StrengthOfTarget	Integer	No	No	No

Table 4: Yaver's Assignment Data

4.1.2.5 New Message Data

In order to compose a new message and send it, user chooses new message part. New message part consists of three basic features which are address information, title and contents of the message. It is similar to well-known mail programs' send mail feature. Detailed information will be seen at section 6.1.5.

Field	Type	Null	Foreign Key	References
SendDate (P.K)	DateTime	No	No	No
SendTime (P.K)	DateTime	No	No	No
Address	Varchar(100)	No	No	No
TitleOfNewMessage	Varchar(100)	No	No	No
ContentsOfNewMessage	Varchar(500)	No	No	No

Table 5: Yaver's New Message Data

4.1.2.6 Filtering Data

Main purpose of the project is being awareness on environmental changes and then dynamically redesigning GUI to provide more usability. In order to do this, some properties are hidden or sizes of the some properties are changed. Default behavior is identified by developers but user may want to change this behavior for own purposes. At this part, user creates own filter and chooses behavior of the properties. Detailed information will be seen at section 6.1.6.

Field	Type	Null	Foreign Key	References
ActiveTab	String	No	No	No

Table 6: Yaver's Filtering Data

4.1.2.7 Redesign GUI Data

This section contains some properties about changing design of the GUI. Contrast values of the screen, current button size and font size dynamically adapted with respect to environment. Each object (buttons, textfields etc…) has unique id called ObjectId.

Field	Type	Null	Foreign Key	References
ObjectId (P.K.)	Double	No	No	No
ButtonSize	Double	No	No	No
FontSize	Integer	No	No	No
Contrast	Integer	No	No	No

Table 7: Yaver's ReDesign UI Data

4.1.2.8 Map – Server Relation Data (isOnline)

Server sends map with respect to the coordinate value. This relation has an extra entity called isConnected, which shows status of server connection.

Field	Type	Null	Foreign Key	References
CoordinateMap (P.K.)	Varchar(24)	No	No	No
isConnected	Bool	No	No	No

Table 8: Yaver's isOnline Data

4.1.2.9 Map – Toolbar Relation Data (UpdateMap)

Toolbar contains some map-related functions such as Zoom in/out, move left, right, up, down. By using these functions, maps vision capability and area is changed. If obtained coordinate values by using move functions are out of the downloaded map, system wants new map by updated coordinates. This relation has an extra entity called isOutOfBound. Its value depends on whether UpdatedCoordinate is in or out of map.

Field	Type	Null	Foreign Key	References
CoordinateMap (P.K.)	Varchar(24)	No	No	No
UpdatedCoordinate (P.K)	Varchar(24)	No	No	No
isOutOfBound	Bool	No	Yes	Map

Table 9: Yaver's UpdateMap Data

4.1.2.10 Map – Assignment Relation Data (ShowRoute)

Route of the current assignment and target coordinate is showed on map.

Field	Type	Null	Foreign Key	References
CoordinateMap (P.K.)	Varchar(24)	No	No	No
AssignDate (P.K)	DateTime	No	No	No
AssignTime (P.K)	DateTime	No	No	No
IsAssignmentCompleted	Bool	No	Yes	Assignment

Table 10: Yaver's ShowRoute Data

4.1.2.11 Toolbar – Message Relation Data (ShowNotification)

When new message comes from command center, a notification is seen on toolbar, and user reaches this message by choosing this notification.

Field	Type	Null	Foreign Key	References
UpdatedCoordinate (P.K.)	Varchar(24)	No	No	No
ReachDate (P.K)	DateTime	No	No	No
ReachTime (P.K)	DateTime	No	No	No
HasNewnotification	Bool	No	Yes	Message

Table 11: Yaver's ShowwNotification Data

4.2 Overall Model Package

Figure 2 shows overall packages and relationship between them with data properties. Since context-awareness is too important it affects all modules. Moreover, New Message modul is submodul of Message i.e. to send a new message, user have to open message part first.

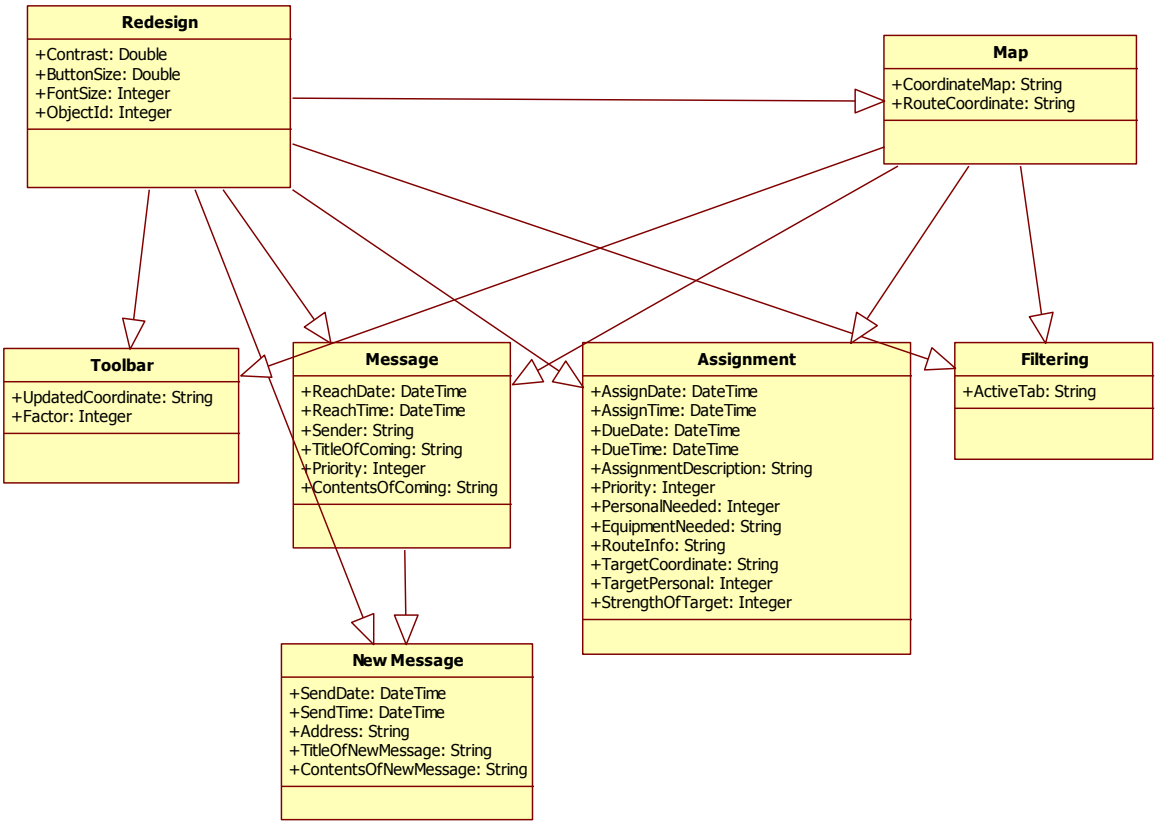

Figure 2: Overall Packages(Modules)

4.2.1 Packages with Functions

Redesigning GUI has several functions to change GUI's current view. Hide and show button change button status which is visible or not by taking information from filtering part. ChangeFontSize and ChangeButtonSize changes the size of the object having ObjectId given and return new size and sets them to related entity. ChangeContrast function changes the screen contrast, return and set new contrast to contrast entity. Hide and show text changes the visibility of text that has given ObjectId. getButton and getTextfield functions return related entity with respect to the ObjectId.

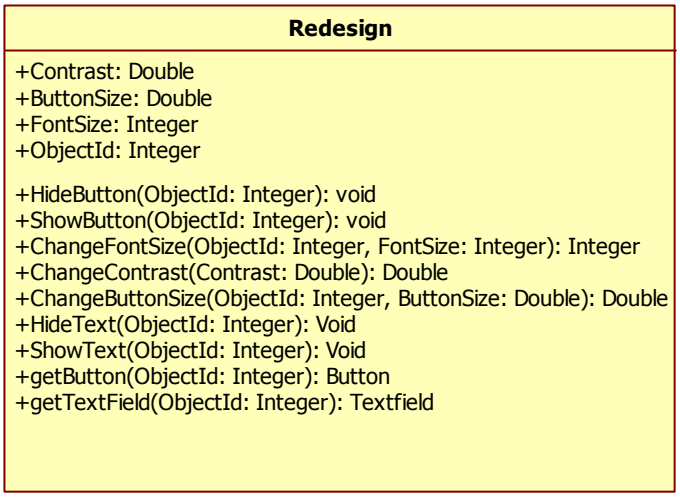

Figure 3: Redesign UI Class

DownloadMap function takes a Coordinate and return a map by downloading a map from server by using this coordinate.

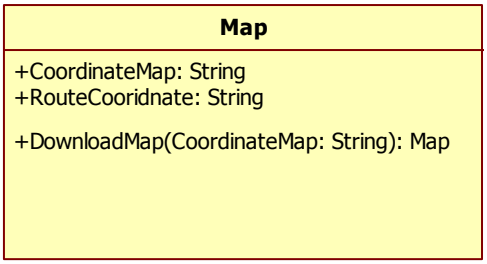

Figure 4: Map Class

ListAssignment function returns list of assignments that is assigned to user. OrderAssignment function is also return list of assignments. By using desired entity, assignments are listed as sorted.

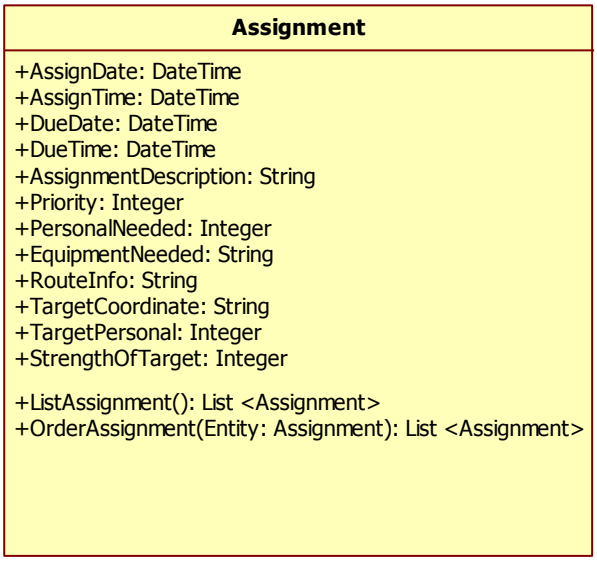

Figure 5: Assignment Class

Zoom In/Out function takes an integer called Factor which changes visibility amount of map.Therefore this function should be return a Map. Similarly, Move Left/Right/Up/Down function takes a coordinate value called UpdatedCoordinate and return a Map.

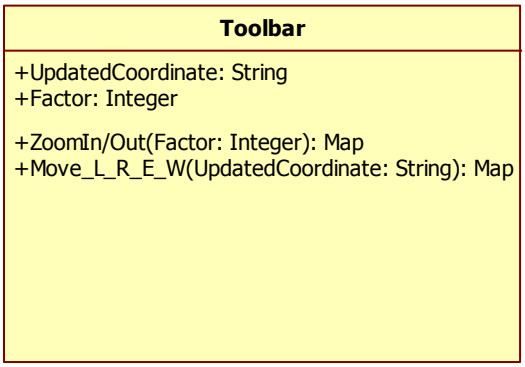

Figure 6: Toolbar Class

Yaver also provide users to create his/her own filter by Filtering options. User can describe the priority of each attribute with DescribePriorities function.

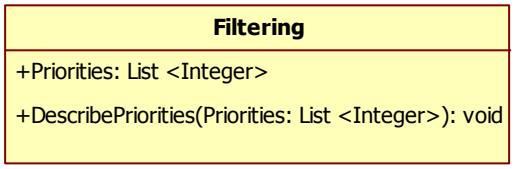

Figure 7: Filtering Class

User can communicate with command center by using Yaver. User can see Inbox and Sentbox with ShowInbox and ShowSentbox function. These functions return list of messages. User can see contents of the message by the method called OpenMessage. In order not to face confliction ReachDate and ReachTime are chosen primary id. Similarly Delete Message is used to delete messages from internal DB. To create and send message user choose New Message and an instance of New Message class should be created.

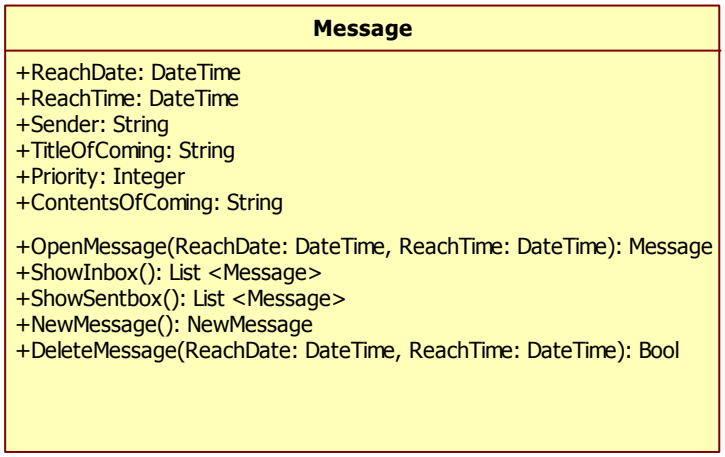

Figure 8: Map Class

After creating New Message instance user set related entity and system call SentMessage method. Its return value represents whether sending message is successfully or not.

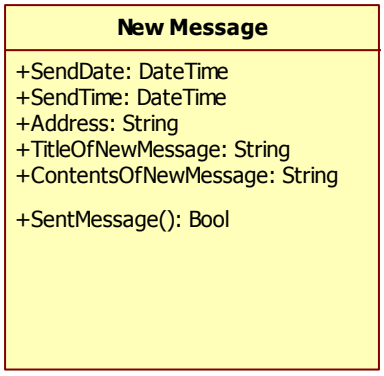

Figure 9: New Message Class

4.3 Data Flow Diagrams

4.3.1 Level 0 DFD

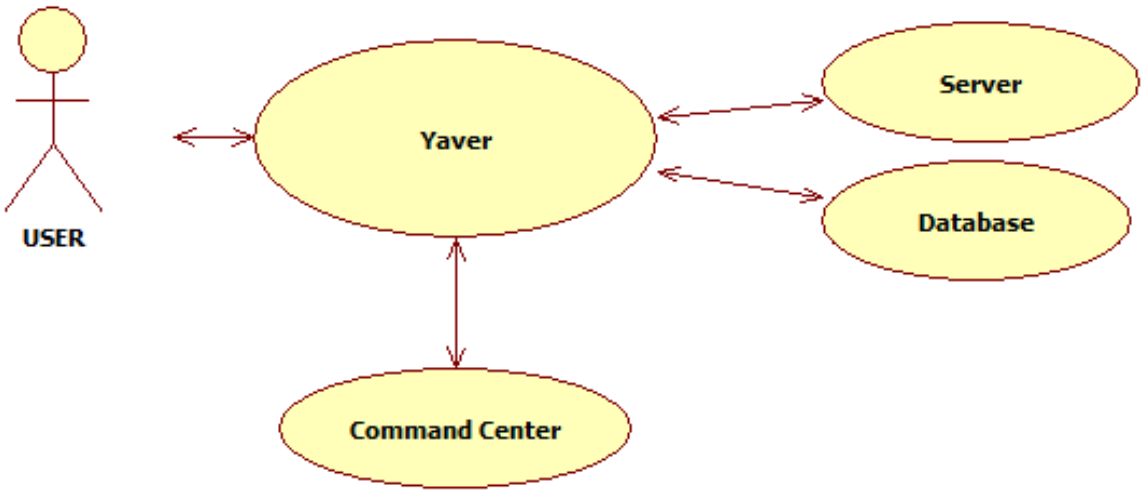

Figure 10: Level0 DFD

4.3.2 Level 1 DFD

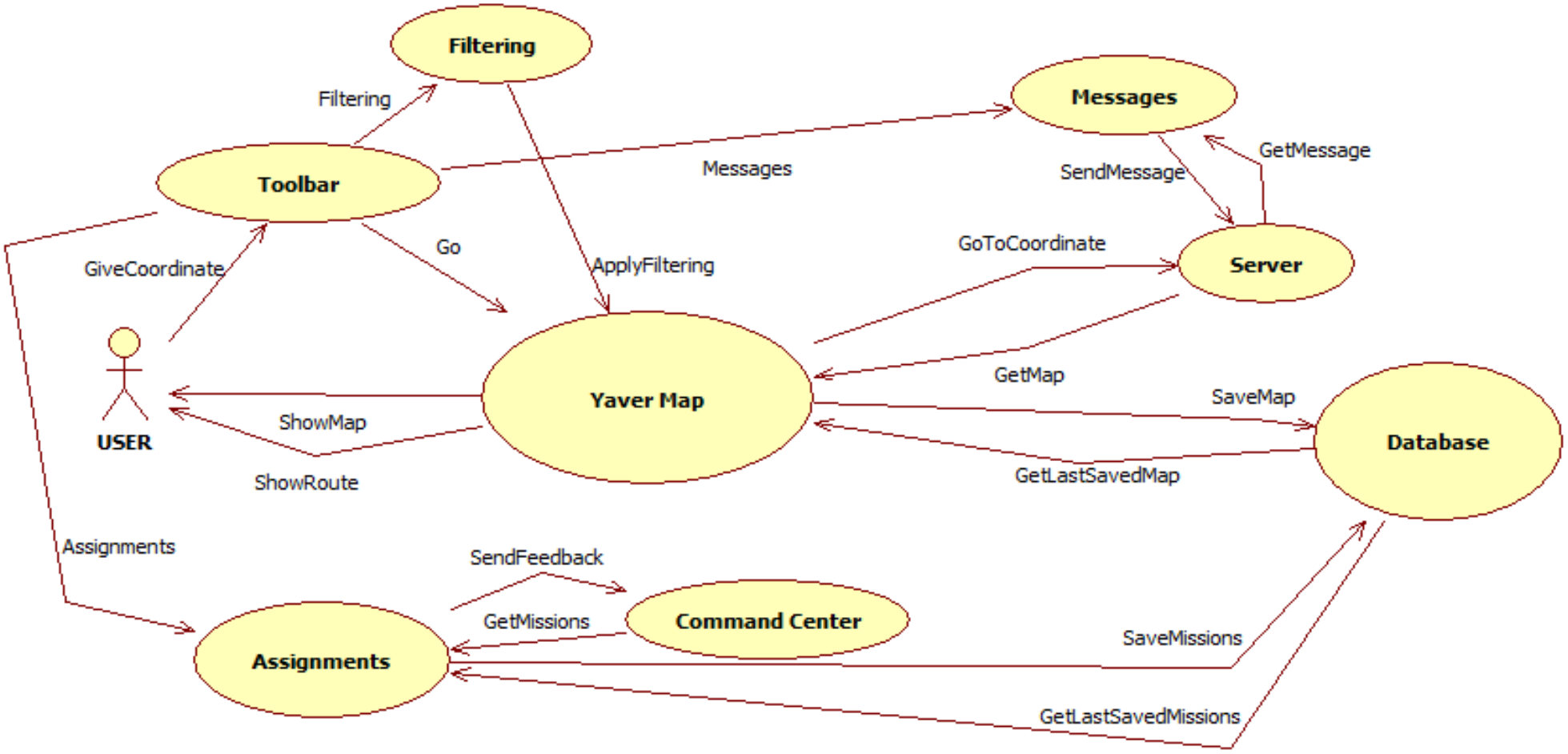

Figure 11: Level1 DFD

4.4 State Transition Diagram

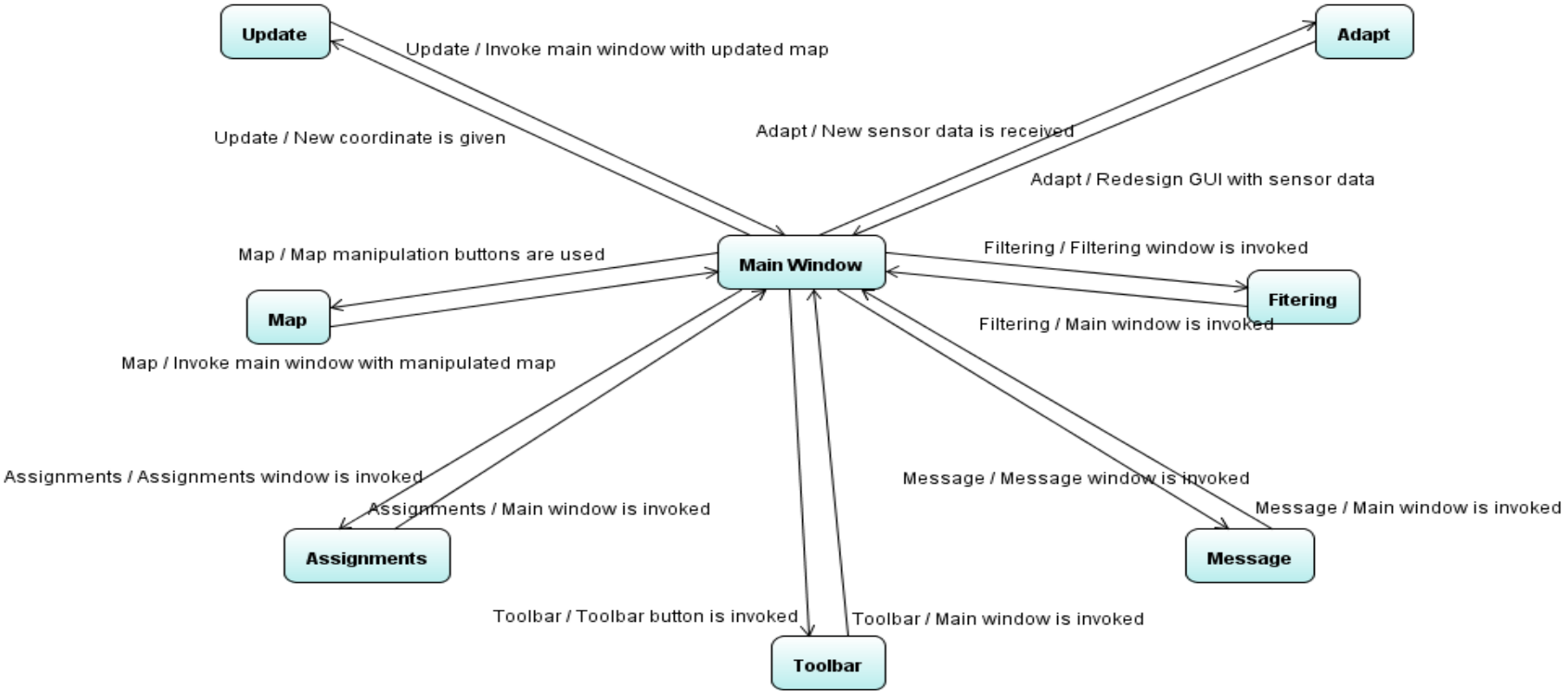

Figure 12: State Transition Diagram

As it can be seen in the above state transition diagram, “Yaver” has 8 main states. When the software is launched, user enters the main state (main window is initiated) immediately. In this state user will see an interface which has a map and toolbar. When user gives a coordinate to the system, the map will be updated and system will show some previously selected information on the map. Also, the user interface adapts itself according to the environment and user’s motion independently from user actions, which system will automatically redesign the graphical user interface according to the data taken from camera sensors. These two operations are done by the system’s itself.

In the main state, user can go into the toolbar, message, assignments, filtering and map states. When one of the toolbar buttons is pressed, the system will enter the toolbar state. According to the type of the button, the system responses back. Similar demand / response operations are done between the other states, namely message, filtering, assignments, map and the main state.

5. System Architecture

5.1 Architectural Design

Yaver consists of three main components namely User Package, Server Package and database. The application connects to server and server connects to database via JDBC interface. Below is the deployment diagram:

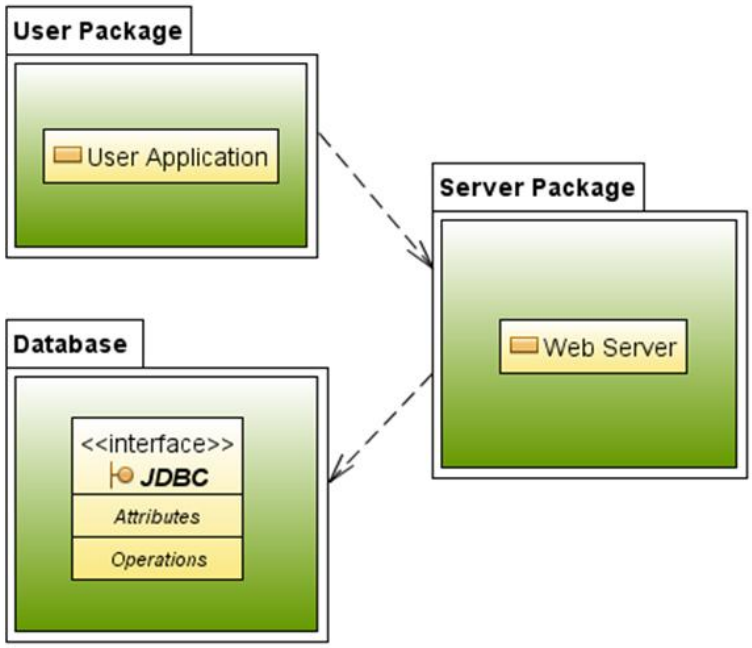

Figure 13: Deployment Diagram

Yaver consists of two main packages namely Server Package and User Package. Server package is responsible for responding to user requests. User package is responsible for handling user actions.

User Package has four sub-packages:

- Message Package
- Toolbar Package
- Filtering Package
- Assignment Package

Server Package has two sub-packages:

- Image Processing Package
- Communication Package

Sub-package diagrams and also top view package diagram are given below:

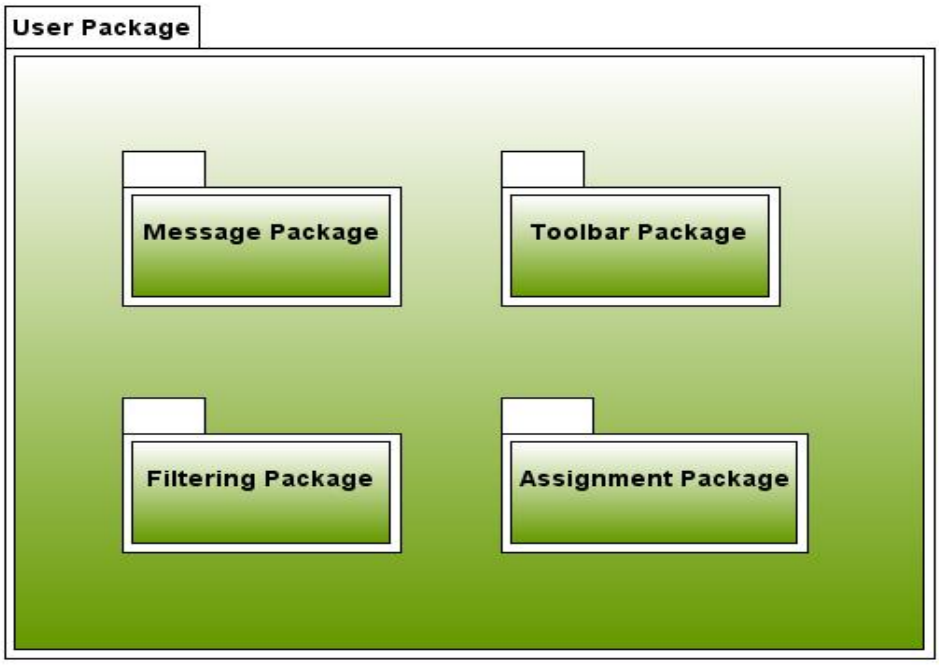

Figure 14: User Package and Its Subpackages

User package and its four sub-packages are responsible for responding to user demands done with the user interface. Message package is responsible for the operations related to message activities of the user. Toolbar package implements the toolbar operations. Filtering package implements the filtering operations and assignment package is responsible for the assignment operations.

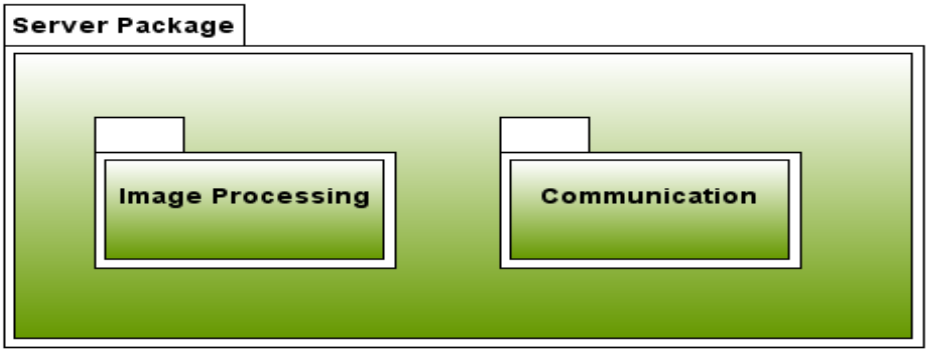

Figure 15: Server Package and Its Subpackages

Server package is responsible for responding actions which are independent from the user. These are the redesign of the graphical user interface according to sensor data and map updates according to the given coordinates.

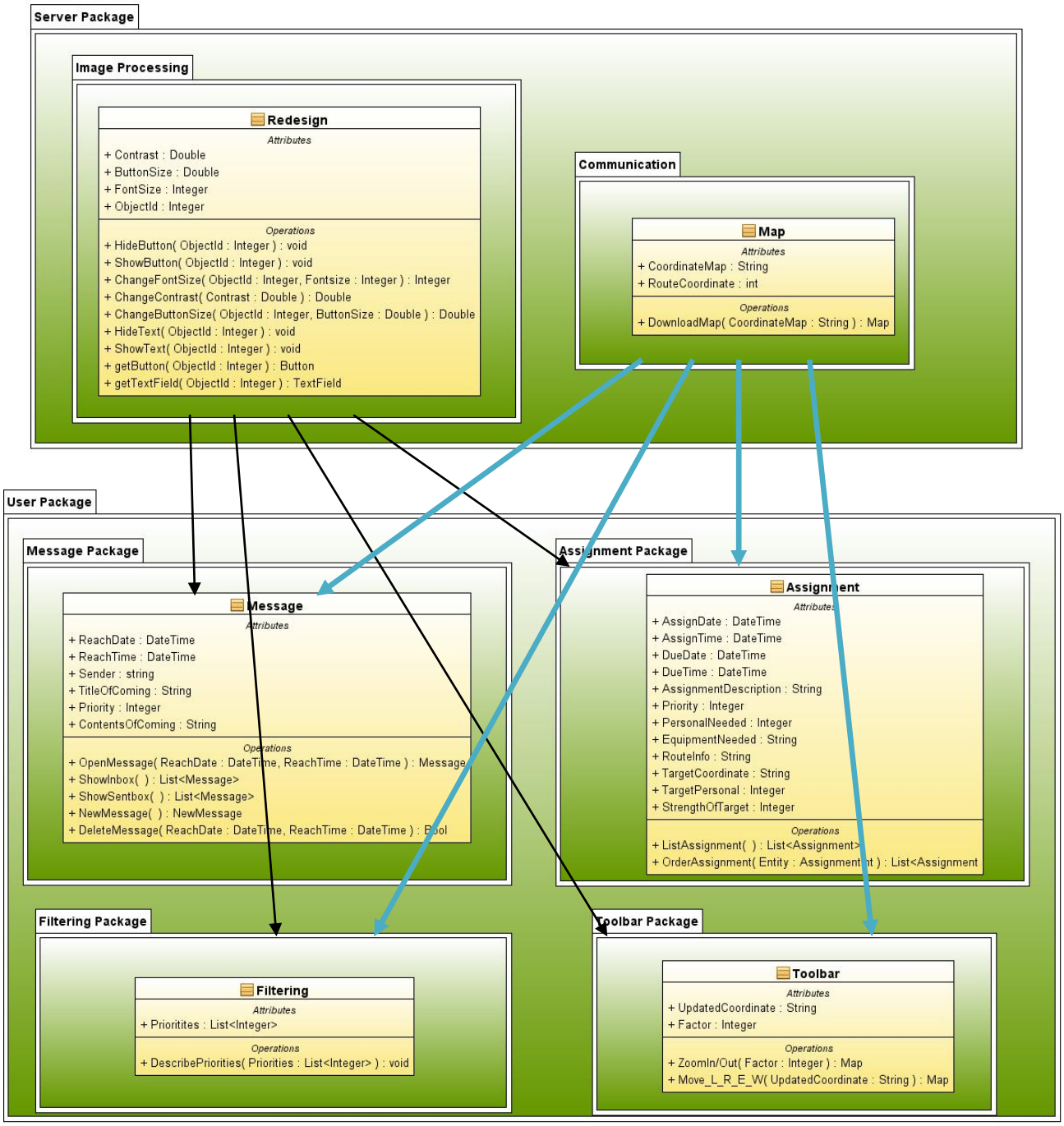

Figure 16: Modules and Classes Top View

Above diagram is the top view of the packages. It shows the relations between sub-package.

5.2 Description of Components

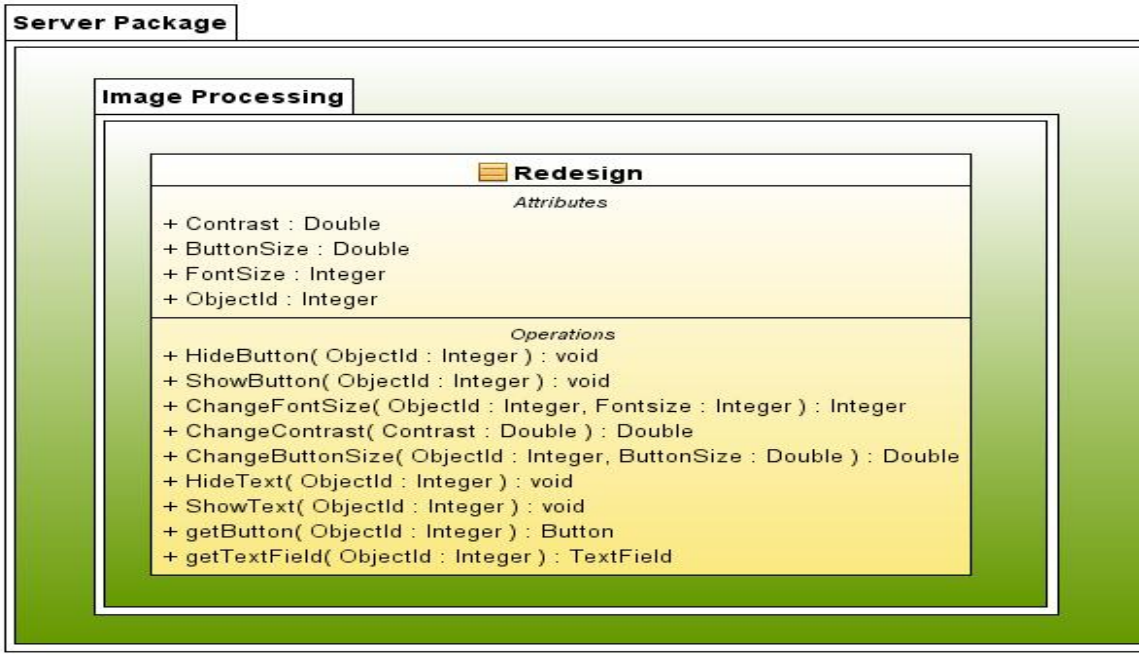

Figure 17: Server Package and Its Classes Part I

Server package has two classes. First one is the Class Redesign. This class has several methods to hide and to show interface elements, to change font size, contrast, button sizes.

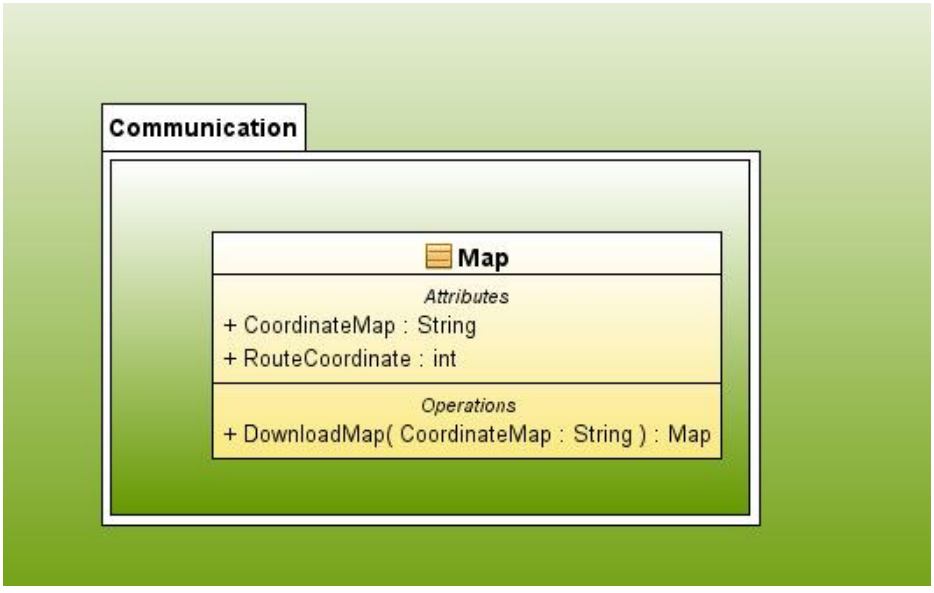

Figure 18: Server Package and Its Classes Part II

This is the second class of the server package, Class Map. This class's duty is downloading the map according to the given coordinate from user as it can be seen in the operations of Map class.

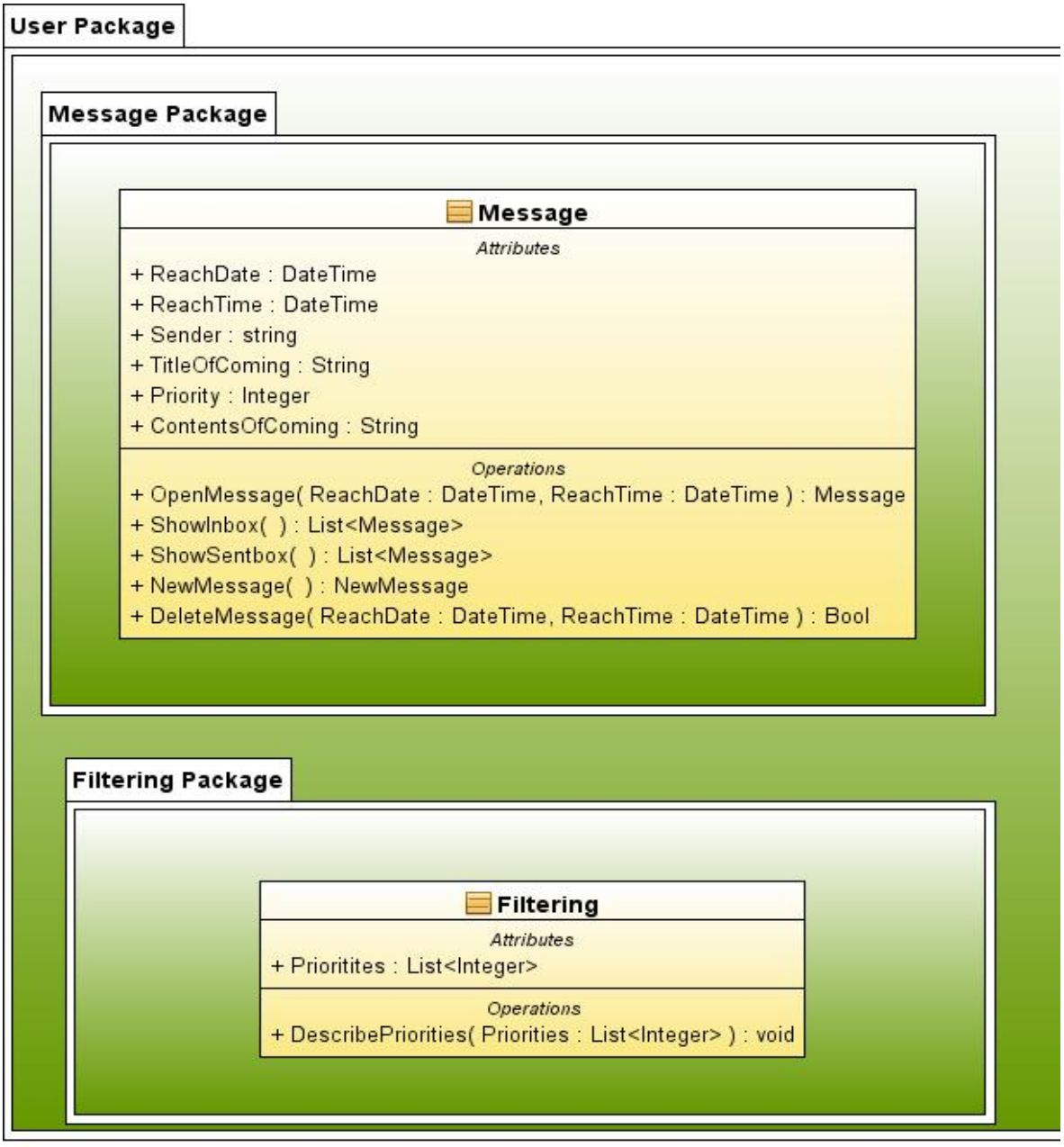

Figure 19: User Package and Its Classes Part I

User package has four classes. Message class has methods to open messages, to show inbox and sentbox, to write a new message and to delete a message. Filtering class is simply responsible for listing the filters.

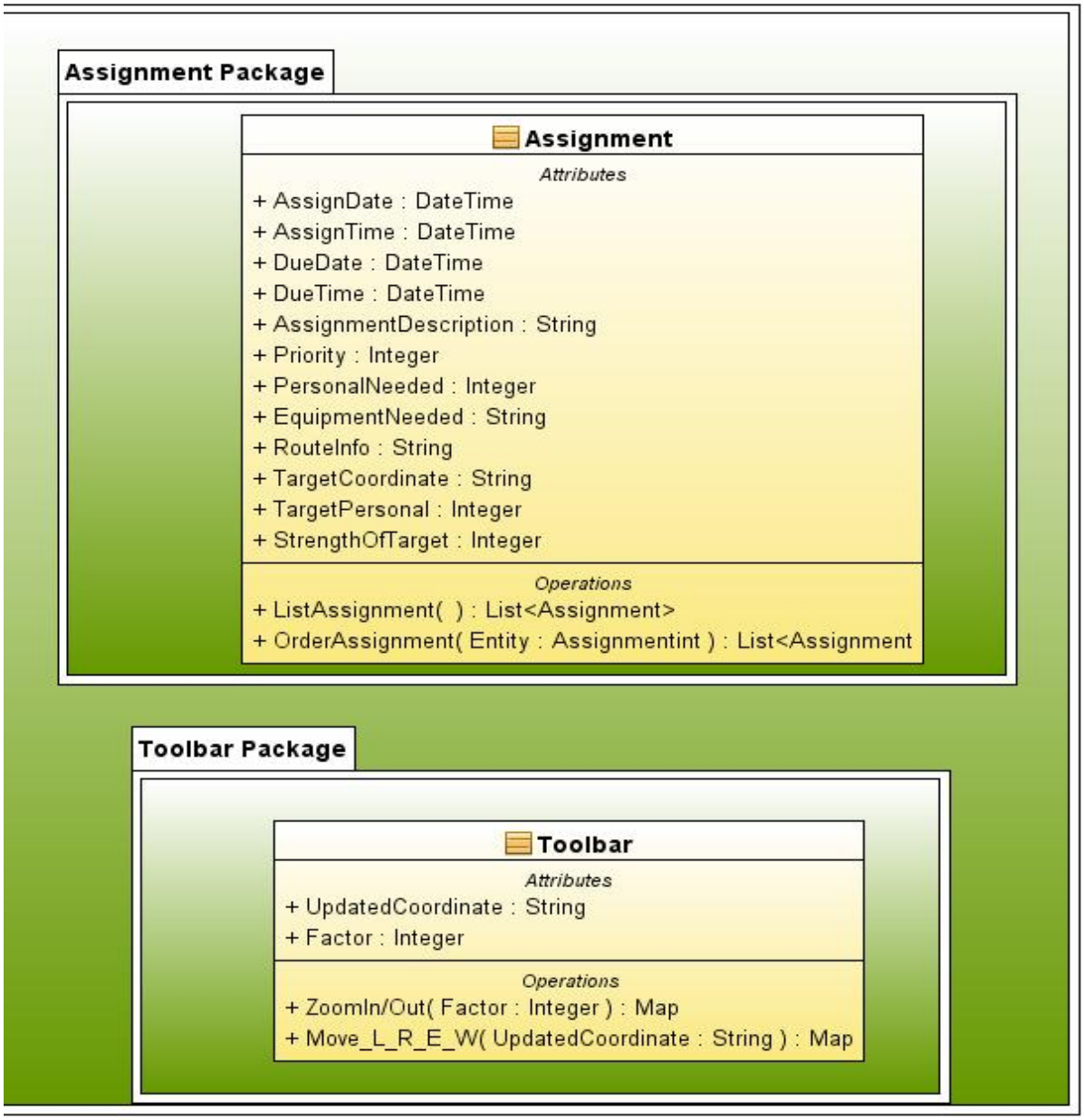

Figure 20: User Package and Its Classes Part II

Assignment class has two methods to list the assignments and order some assignmnets. Toolbar class has also two methods. Their main responsibility is manipulating the map by zooming or moving on map in main directions.

- **Server Package**
  - **Redesign Class:**

    This class is responsible for updating the graphical user interface according to the data collected from camera sensors. It is the class where context-aware function of the application is provided.
  - **Map Class:**

    This class is responsible for updating and refreshing the map in the graphical user interface according to the given coordinates.

- **User Package**
  - **Message Class:**

    This class is for sending and getting messages, and manipulation the mail boxes.
  - **Toolbar Class:**

    This class is for enabling the user to manipulate the map window by zooming in and out, moving left, right, up and down.
  - **Assignment Class:**

    This is the class whose responsibility is to provide ordering and listing assignments.
  - **Filtering Class:**

    Filtering class defines which entities will be shown in the application according to application modes.

5.2 Sequence Diagrams:

- **Message Component**

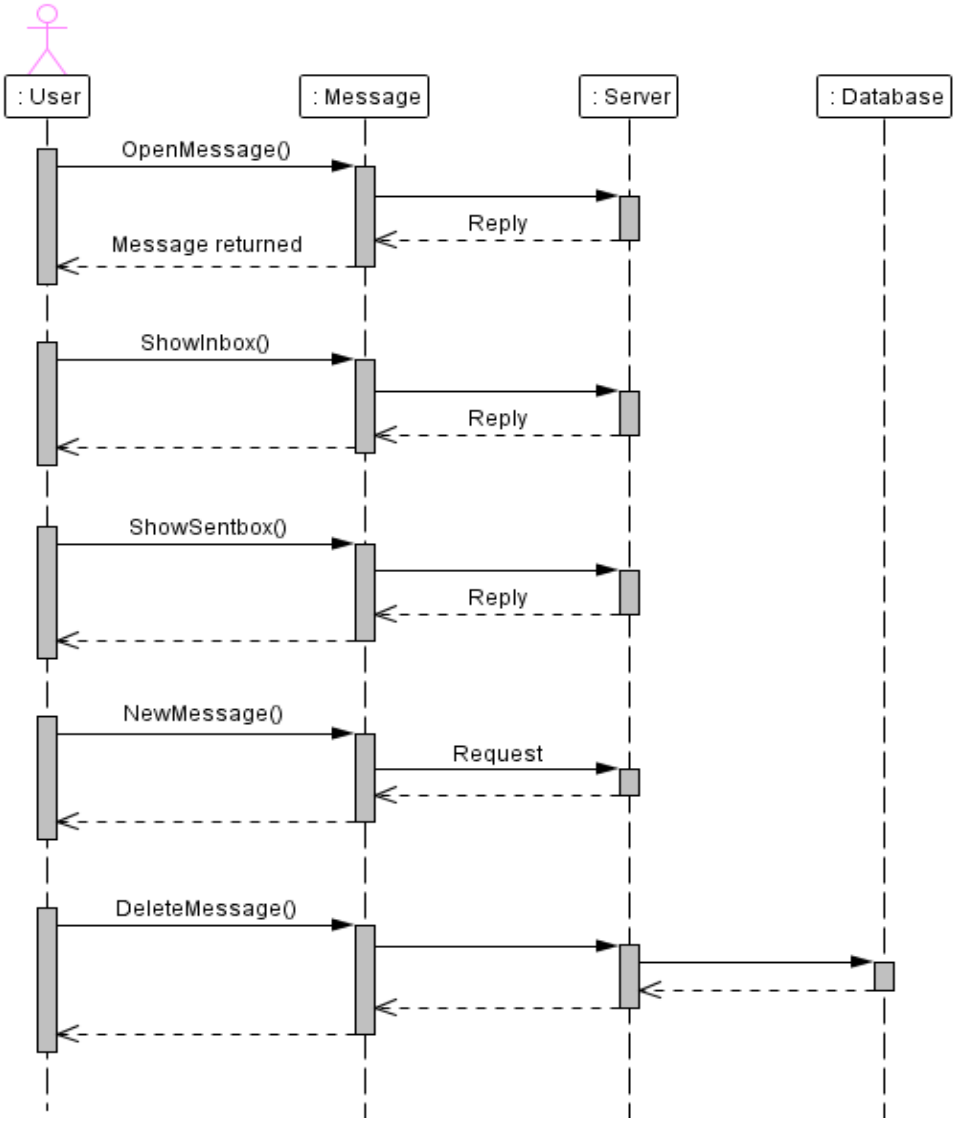

Figure 21: Message Sequence Diagram

Message sequence has four stages. User, Message component , Server and Database. To open a message, to show inbox or sentbox, and to write a new message user-message-server sequence occurs and server replies back to the user. To delete messages user accesses database and database returns the result back to user.

- **Assignment Component**

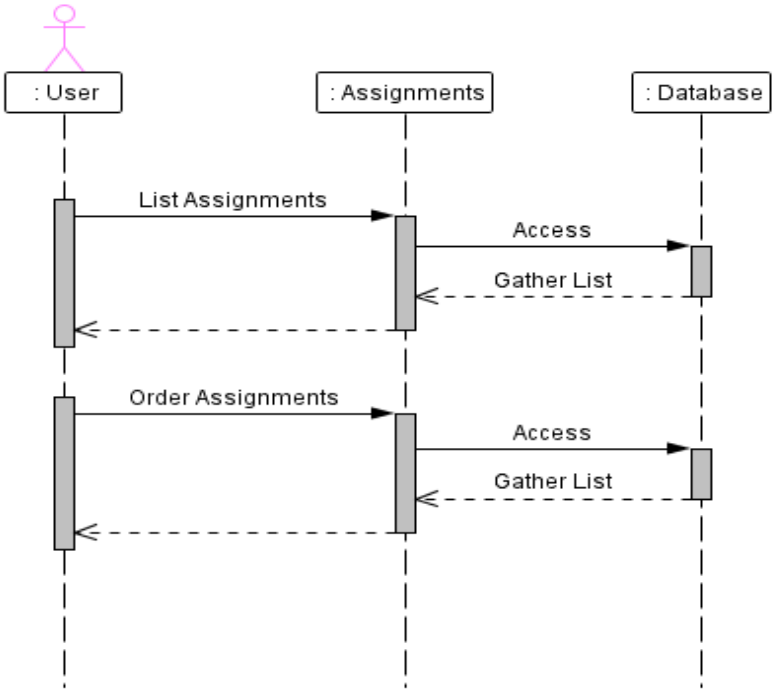

Figure 22: Assignment Sequence Diagram

Assignment component has two functions: List Assignments and Order Assignments. Since assignmnets are kept in the database, when these functions are called, user accesses from assignment component to database and the list of assignments are returned to the component.

- **Filtering Component**

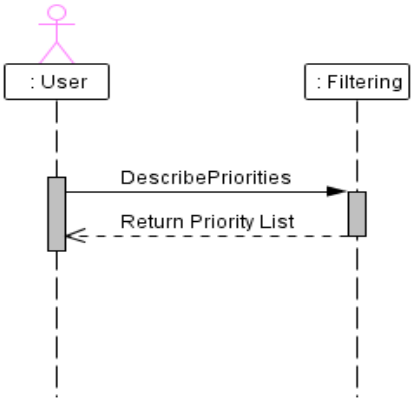

Figure 23: Filtering Sequence Diagram

When filtering component's "DescribePriorities" function is called it simply returns the list of the priorities to the user.

- **Toolbar Component**

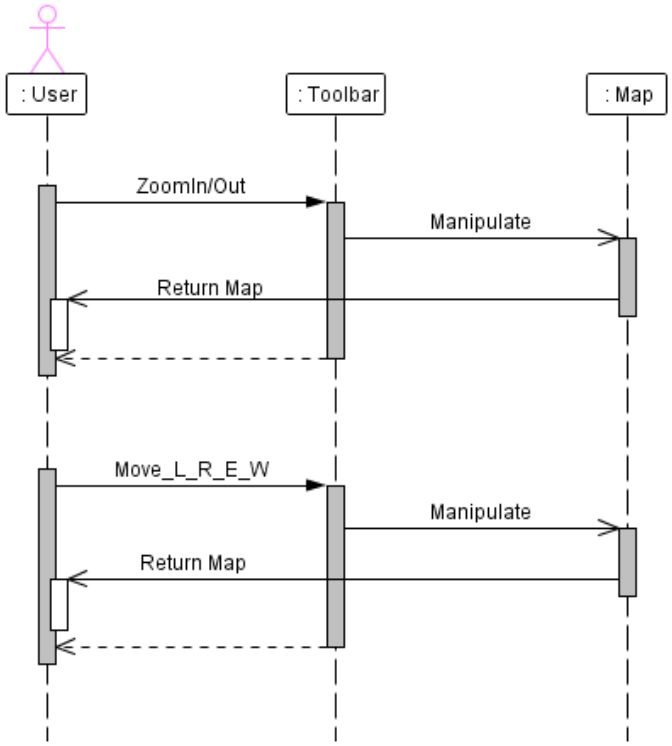

Figure 24: Toolbar Sequence Diagram

There are two features available in the toolbar component for the user. First one is zoom in/out option. When user presses this button toolbar component's corresponding function is called and then it manipulates the map and returns the result to the user. Another option in the toolbar component is moving on map in main directions. When this button is pressed, corresponding function is called and after the manipulation of the map, the map is returned to the user.

6. User Interface Design

6.1 Overview of User Interface

6.1.1 Main Window

This screenshot shows the main window of our system “Yaver”. This window mainly consists of map screen and toolbar screen. The physical map of environment which user stands is shown in the upper part of window. At the bottom of this screen, there is a toolbar which has features to manage the map screen and the main window. Toolbar has some buttons to manage the window and map screen. These are zoom-in/out tool, move-left/right/up/down tool, open messages button, list the assignments button, go to the specified coordinate button, and filtering button. Furthermore, if the user opens a new window such as messages or assignments window, the map screen becomes smaller and new window is located right of the main window.

6.1.2 Toolbar Screen

This screen is displayed at the bottom of the map area in the main window. As it was mentioned, there are several buttons and tools in this screen. Zoom tool and move tool are used for traveling on the map. There is a slider for zoom operations and there are four arrows for moving operations. These tools are on the left of toolbar. There are two text boxes to write X and Y coordinates and one button namely go to. User can go to desired coordinates by pressing this button after describing the coordinates. Messages button opens a window which has properties to read or send messages. After being pressed this button, user sees a new window on the right. Assignments button opens a window which displays assignments like messages button. This screen is also located on the right. Filtering button also opens a new window. This filtering screen is for setting the filters.

6.1.3 Map Screen

The physical map of environment which user stands is shown in this screen. Required information to represent the map is downloaded via the internet connection. This download operation is conducted by the system as independent from user. Specific coordinates that user wants to see can be seen by helping marker. User determines these by writing coordinates into text box in the toolbar. Moreover, routes and some objects (buildings, units etc.) whose coordinates sent by command centre can be displayed on the map. User travels on the map by dragging mouse. Traveling operation is also done by using the tools in the toolbar. This screen becomes smaller when the user opens a new window such as assignments and messages window. Furthermore, if the user wants to learn more about objects on the map, s/he can click on the object and see a small information window which displays properties of clicked object.

6.1.4 Messages Screen

This screen is a simple program resembling Thunderbird which user can read messages sent by command and send messages. There are several tools in this window which are inbox, sentbox, new message and delete message tools. User sees messages sent by command centre by using inbox tool with title and name of the sender. Unread messages titles are shown as bold. After the user clicks the inbox tool, the messages which are sent by command centre can be seen on the bottom right of this screen. To read messages user should click the title of message which s/he wants to see. Moreover, user sees messages sent by him/her to the command centre by using sentbox tool. This tool displays the sent messages with titles on the bottom right of screen. User clicks on the title of message and see the content of message. Also new message tool opens a new window to write a new message. New window has also some features which are one send message button, one text plain and two text boxes. Text boxes are for mail address and title. After message is written into text plain, this button is pressed and the message is sent. There are also some features in this window such as font size and font style. User can change text properties by using these tools. If user wants to delete a message, s/he selects message that is wanted to be deleted by clicking the message title, and then press the delete button.

6.1.5 Assignments Screen

Assignments which are assigned by command centre via internet connection are shown in this window as a table. Uninvestigated assignments are shown as bold. Table has some features about the assignment. These are assign date, due date, description, priority, personal need, equipment need, route, target coordinate, target personal and strength of target. Also, each assignment has a name and these features are represented next to the name as a table. User can arrange assignments in order with respect to some features which are due date, priority, target coordinate and strength of target. User click on the feature name and table contents are arranged according to this entity.

6.1.6 Filtering Screen

In this window, user defines filters for all other windows buttons and entities. There are three tabs for each window in this system. Each tab has all buttons or entities in the specified window. Firstly, toolbar tab has only buttons of toolbar window, since entities are not important as buttons are, and three check boxes next to name of the button. These check

boxes are the same in all three tabs and they are stationary, walking and running. These indicate the status of user. If user ticks one of them or more, this means that user sees this button during only these statuses. For instance; user ticks stationary box of “filtering” button in the toolbar, then user sees this button only in stationary mode, while running or walking s/he doesn’t see this button. Secondly, tab for message screen has all buttons and entities in this window with check boxes. Thirdly, Assignments tab has only entities in this window with check boxes, since there is no button there. Moreover, end of the entities or buttons list there is a check box to set filters as a default. After user selects the status of entities or buttons, s/he should press the Set Filter button.

6.1.7 Information Window

This window is popped up when the user clicks an object on the map. This screen shows information about clicked object. When the user clicks an object on the map screen this screen is displayed. Information window shows some properties of object. These properties can be length of road or characteristics of a building. This screen shows these properties as a text.

6.2 Screen Images

6.2.1 Main Window (Toolbar Screen and Map Screen)

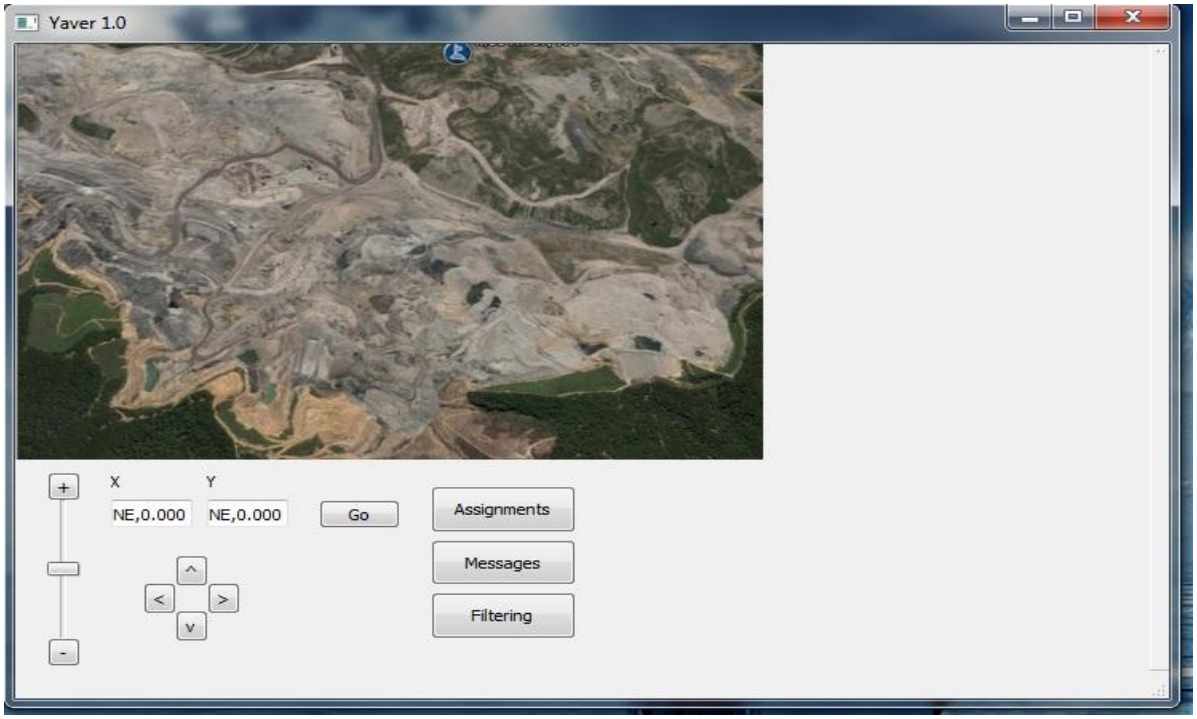

Figure 25: Appearance of Main Window

6.2.2 Assignments Screen

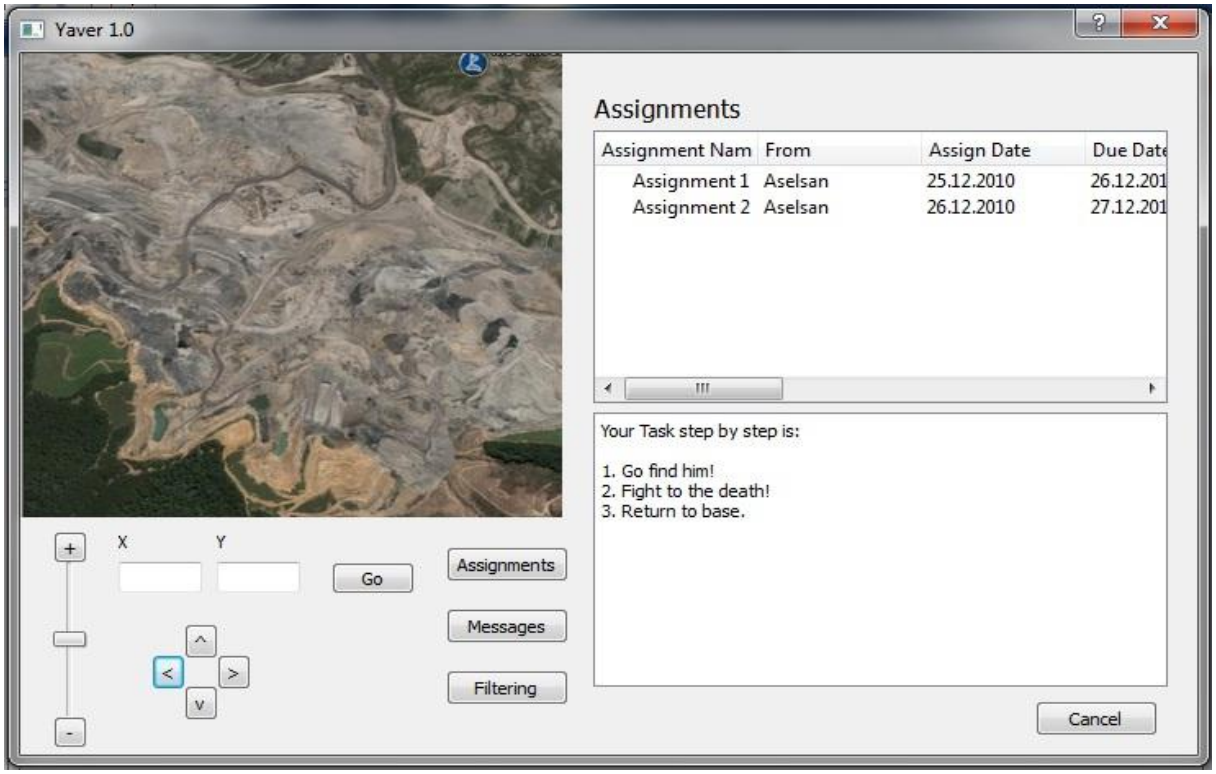

Figure 26: Appearance of Assignment Window

6.2.3 Messages Screen

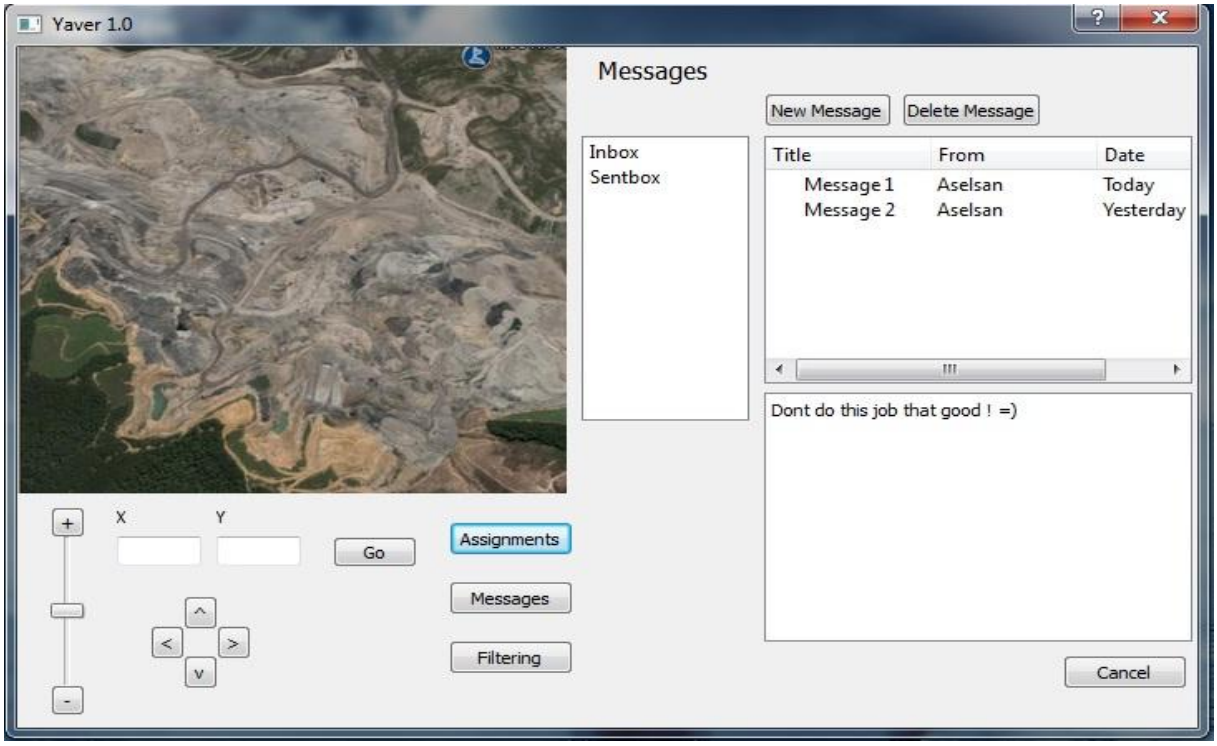

Figure 27: Appearance of Messages Window

6.2.4 New Message Screen

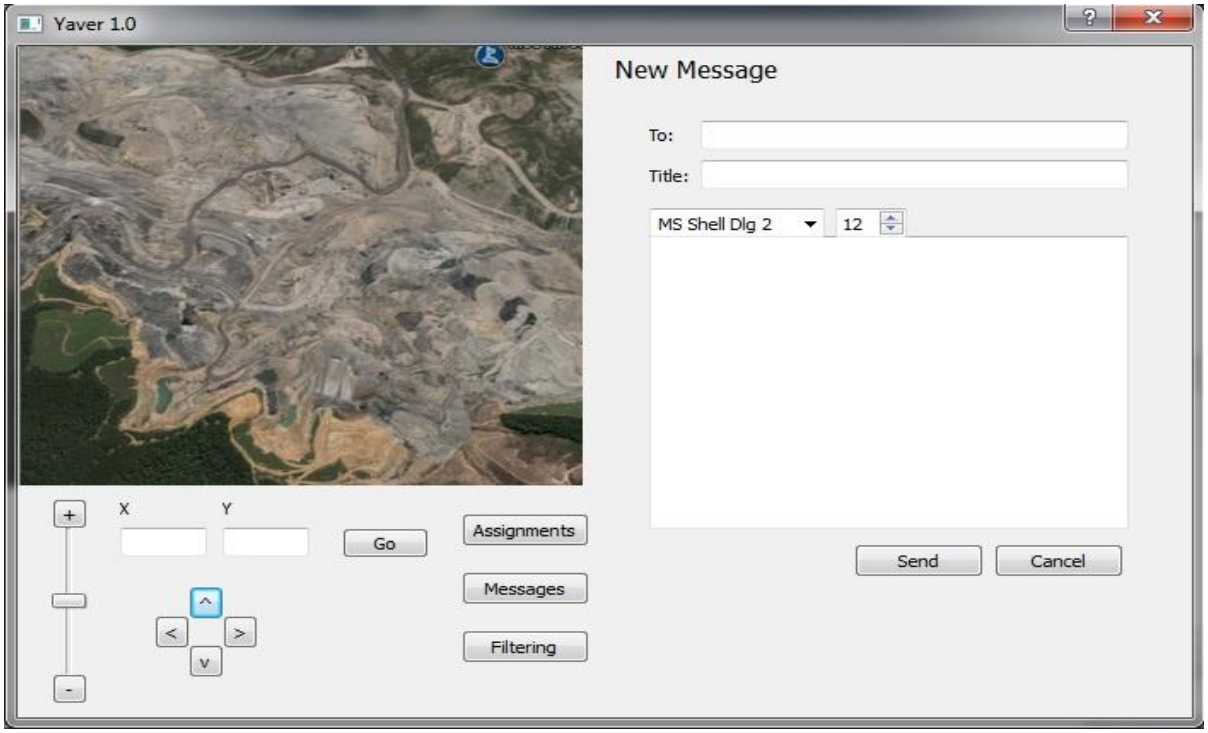

Figure 28: Appearance of New Message Window

6.2.5 Filtering Screen

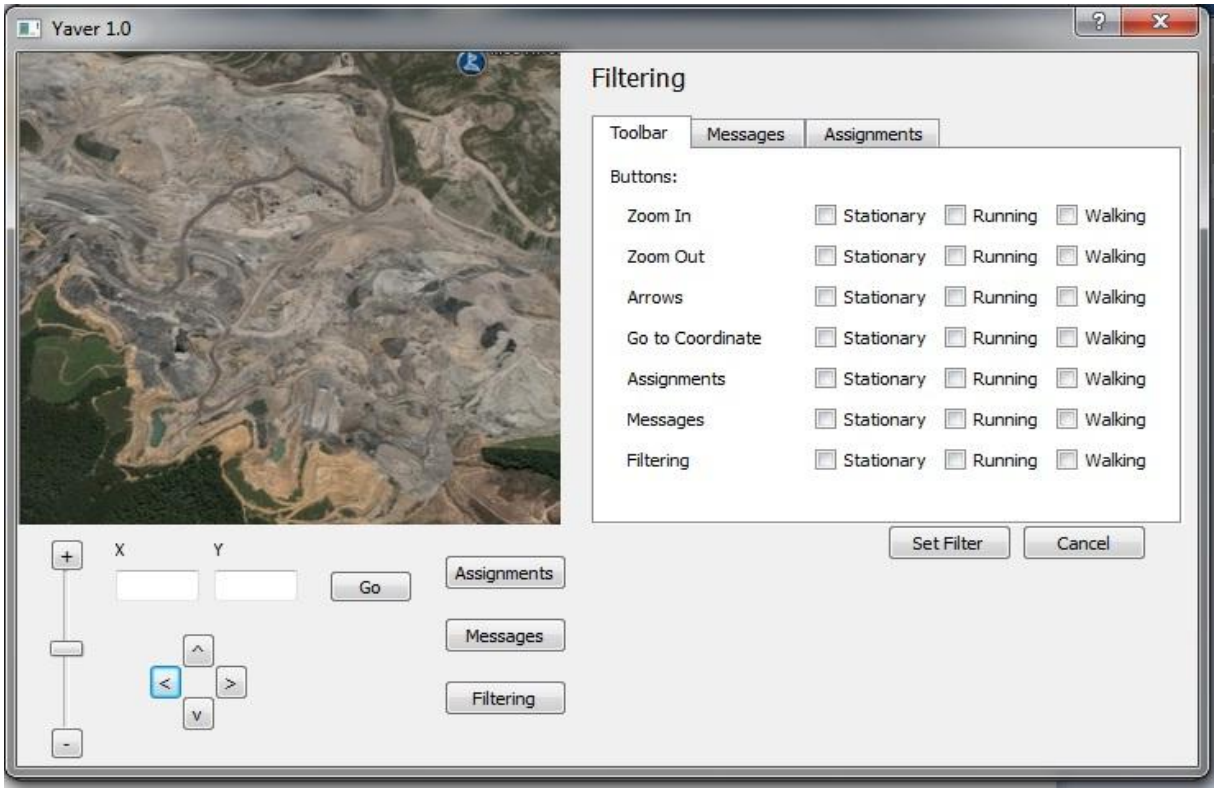

Figure 29: Appearance of Filtering Window

6.3 Screen Objects and Actions

6.3.1 Map Object

Map object is kept on the database and shown on the map screen. Map screen is in the main window. This object is kept with center coordinate value and on the screen environment of this coordinate value is displayed. Zoom factor determines the size of environment. This zoom factor changes when the user zooms in or zooms out. Moreover, when the user travels on the map, the center coordinate value changes. Physical environment on the map also changes with respect to this value.

6.3.2 Message Object

Each message is an object which was explained in the data design section. In the user interface, contents of message can be seen by clicking on message title in the messages window. Furthermore, user can delete a message object by pressing delete button after select a message.

6.3.3 Assignment Object

Each assignment is an object which was explained in the data design section. In the user interface, assignments can be seen by clicking on assignment title in the assignments window. Furthermore, user can arrange assignments by pressing an entity of assignment. These assignments are kept in a list and display order of these is determined by an entity.

7. Libraries and Tools

7.1 Java Swing

Swing is the primary Java GUI widget toolkit. It is part of Sun Microsystems' Java Foundation Classes an API for providing a graphical user interface (GUI) for Java programs. We chose this toolkit because we are kind of familiar with it, it is easy to use, it is applicable in many platforms (platform-independent) and it has good GUI options.

7.2 Java Advanced Imaging API

The Java Advanced Imaging API extends the Java 2 platform by allowing sophisticated, high-performance image processing to be incorporated into Java applets and

applications. It is a set of classes providing imaging functionality beyond that of Java 2D and the Java Foundation classes, though it is designed for compatibility with those APIs. We chose it because we want to be consistent and using Java again will not cause any time loss probably.

7.3 Java Database Connectivity (JDBC)

JDBC is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the JVM host environment. JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections. We will use JDBC, since messages, assignments and old maps' information in the database needs to be updated by a server.

7.4 Abstract Window Toolkit (AWT)

The Abstract Window Toolkit (AWT) is Java's original platform-independent windowing, graphics, and user-interface widget toolkit. The AWT is now part of the Java Foundation Classes (JFC) — the standard API for providing a graphical user interface for a Java program. AWT is also the GUI toolkit for a number of Java ME profiles. For instance, Connected Device Configuration profiles require Java runtimes on mobile telephones to support AWT. Since the main functionality of our system is redesigning GUI, AWT will be very suitable for this functionality.

7.5 Java Media Framework (JMF)

The Java Media Framework (JMF) is a Java library that enables audio, video and other time-based media to be added to Java applications and applets. This optional package, which can capture, play, stream, and transcode multiple media formats, extends the Java Platform, Standard Edition (Java SE) and allows development of cross-platform multimedia applications. JMF abstracts the media it works with into Data Sources (for media being read into JMF) and Data Sinks (for data being exported out). It does not afford the developer significant access to the particulars of any given format; rather, media is represented as sources (themselves obtained from URL's) that can be read in and played, processed, and

exported (though not all codecs support processing and transcoding). JMF will be used in our project, since our system takes images from camera as input. These images will be added to system by using JMF.

7.6 Google Maps API

Google Maps API allows developers to integrate Google Maps into their websites. By using the Google Maps API, it is possible to embed Google Maps site into an external website, on to which site specific data can be overlaid. Although initially only a JavaScript API, the Maps API has since expanded to include an API for Adobe Flash applications, a service for retrieving static map images, and web services for performing geocoding, generating driving directions, and obtaining elevation profiles. We will use this tool for taking map from Google.

7.7 Netbeans

Netbeans is a free, open-source Integrated Development Environment for software developers. We chose it because we are experienced about using Netbeans and implementing Java applications with it.

8. Process Model

Since none of our group member is experienced in a big java application development, the details will be learned while developing the project. Therefore, we could make some mistake during the development process. For this reason, using a development model like “Waterfall” will be unreasonable, because making a mistake at the beginning of the development process is inevitable by an inexperienced team. For this reason, we need a more flexible development methodology like Spiral model. We decided to use flexible development methodology like “Spiral”. The idea behind this model is that at each choice point in the software development process, one assesses the risk that the project could fail to meet its goals. Based on that analysis the next step is to mitigate that risk. It might mean doing a prototype, refining the requirements, or doing more testing. Therefore, it lets the development team learning by making mistakes and even an inexperienced team can form a solid structure after enough number of iterations.

However, making an adequate number of iterations, could take lots of times. Hence, taking an advantage of Agile Methodology can really speed up our development process. For

this reason, we will mainly follow Spiral methodology during development process. However, for the parts that needs special attention Agile methodology can be feasible.

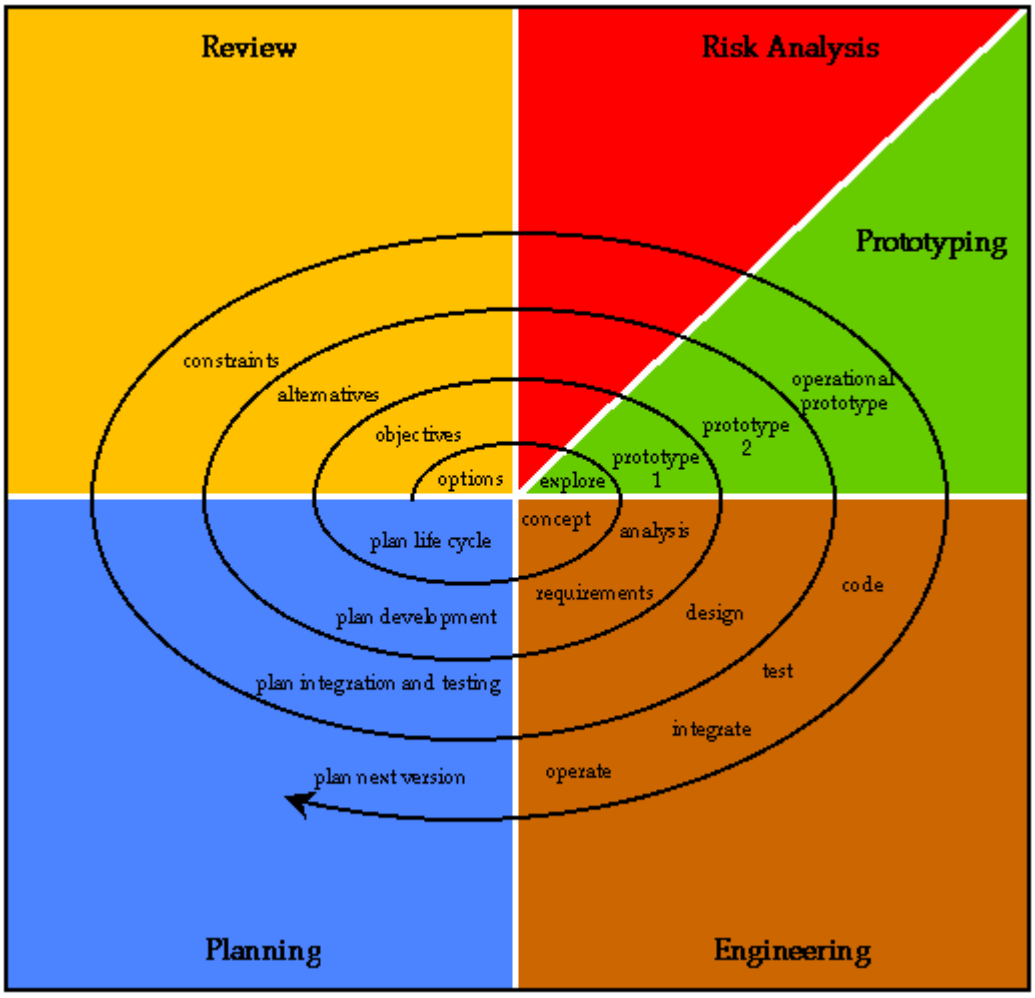

Figure 30: Spiral Development Model

9. Testing Strategies and Procedures

In order to present a robust, bug-fixed product we will make many tests on our application. As a computer science genius Edsger W. DIJKSTRA states “Program testing can be used to show the presence of bugs, but never to show their absence! Keeping this in mind, in order to have a program as free of bugs as possible, it is a must to have a good testing plan. We divided testing procedure into 3 parts.

i. Unit Testing
ii. Integration Testing
iii. Validation Testing

9.1 Testing Strategy

We will make unit testing while creating modules. After implementing a class we will test this already implemented unit. If we find any bug then we will focus on fixing that bug. We will assume that a module takes correct input from another module; that is we will not try to fix inter-module bugs in unit testing. Inter-module bugs will be fixed during integration test. In validation testing, we will check whether our product runs on different platforms or not. During unit testing and integration testing, we will work on Microsoft Windows 7 operating system with already installed Java SDK (jdk). Initially, testing will be held on just after implementation server module.

9.1.1 Unit Testing

We choose white box testing mechanism as strategy since it considers the internal details of a module and tries to test every unit inside. We will implement test mechanism to cover all possible situations that will be tried. Each module will be tested separately. We will test below modules and classes sequentially. At unit testing, we will test modules under excessive (more than expected) stress and sure to be still alive for each module. Moreover, we will test Yaver under invalid inputs to protect system against errors occurred by this kind of inputs.

- ✓ Server Module
  - Communication Class
- ✓ User Module
  - Toolbar Class
  - Assignment Class
  - Message Class
  - Filtering Class
- ✓ Server Module (Again)
  - Redesign Class

Also, we found that for decreasing the errors of our codes and making test easier for us, we have to put into our consideration these characteristics of codes that we will test. (Details will explain at Section Syntax Specifications)

- Simplicity
- Stability
- Understandability
- Controllability
- Operability
- Decomposability

9.1.2 Integration Testing

After we have tested the modules individually, we are going to test all the units of the project by integrating each other and test for any collision between them. Moreover, we will test all the expected features of the system user interface are prepared for easy use, the module functionalities work properly. Finally, we will look for if all the modules are integrated successfully and the functionalities are interacting properly.

9.1.3 Validation Testing

Validation test asks for whether the product is working in the right way or not. After integrating all modules, we will test whole system under different environment conditions with lots kind of inputs.

10. Syntax Specifications

In order to read, understand, maintain and debug codes easily, we have decided to determine syntax specifications. By using below specifications, we will prevent communication problems between developers and some name conflicts.

Function Names:

Each function should start with lowercase letters. If a function name consists of more than one word, each word concatenated and starting letter of each word will be capital.

Example:

function() , myFunction(), findAllPossibleMapsFromServer() etc… are valid.

Get(), getcoordinate(), set_label(), ISVALID() etc… are not valid.

Variable Names:

Since poorly-named variables make code harder to read and understand, it is better to determine a general name convention for this project. As a result, all variables begin with a lowercase letter, and if consisting of multiple words, each are connected with underscore (_) character. Since same named local variables hide global variables, starting with under

underscored name is better. Loop iterators (i,j,k …), and intermediate variables declared as anything developer wants.

Example:

coordinate, current_coordinate, button_size, _contrast(only valid if global) etc… are valid.

x (it is not math lesson), Objectid, reachDate, Priority etc… are not valid.

Class Names:

All classes begin with capital letters. If a class name consists of multiple words, each starting letters of words should be capital. Abstract classes begin with underscore.

Example:

Toolbar, MapExplorer etc… are valid.

map, messageSearcher, Assignment_Lister etc… are not valid.

Comments:

Increasing the understandability of the codes, we write comments after some lines. We will use Javadoc style commenting. Moreover, developers can add extra commands and tricks, if he doesn't be satisfied with Javadoc.

11. Time Planning (Gannt Chart)

11.1 Term 1 Gannt Chart

Kimlik	Görev Adı	Süre
1	Learn Java Swing	31 gün
2	Prepare Initial Design Report	11 gün
3	Get Ready for Team Presentation	16 gün
4	Design Look of GUI	16 gün
5	Prepare Detailed Design Report	11 gün
6	Prototype Demo	11 gün

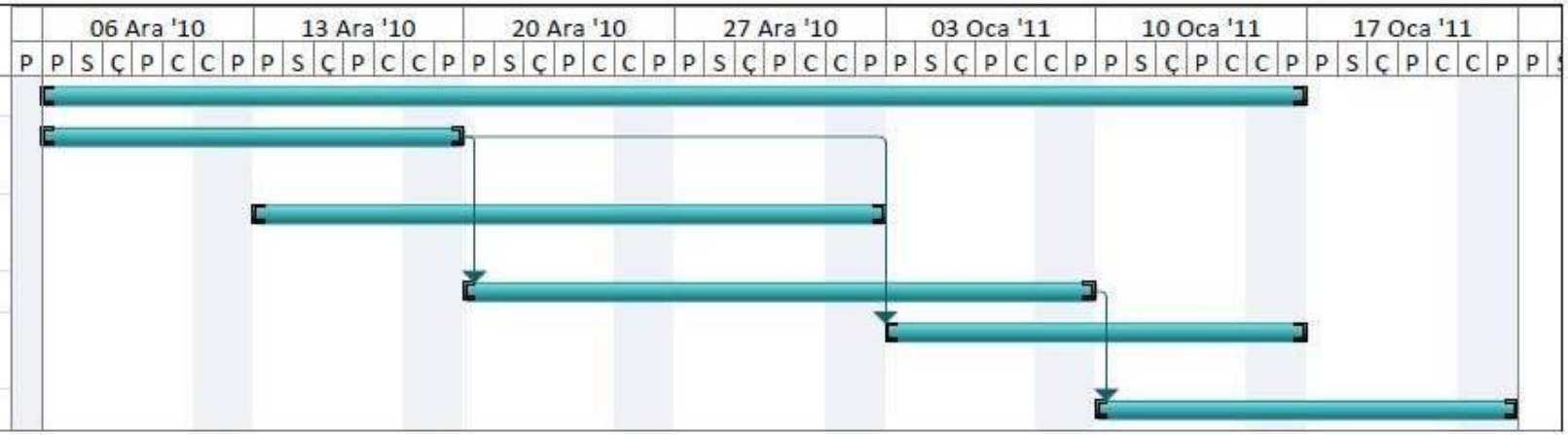

Figure 31: Workload of First Term

11.2 Term 2 Gannt Chart

Kimlik	Görev Adı
1	Implementing GUI Functions
2	Simulate mission,message and map handling
3	Make GUI changeable by using a menu for developers
4	Image Handling
5	Camera Handling
6	Solve Performance Issues
7	Test Application
8	Software Demo
9	Final Presentation

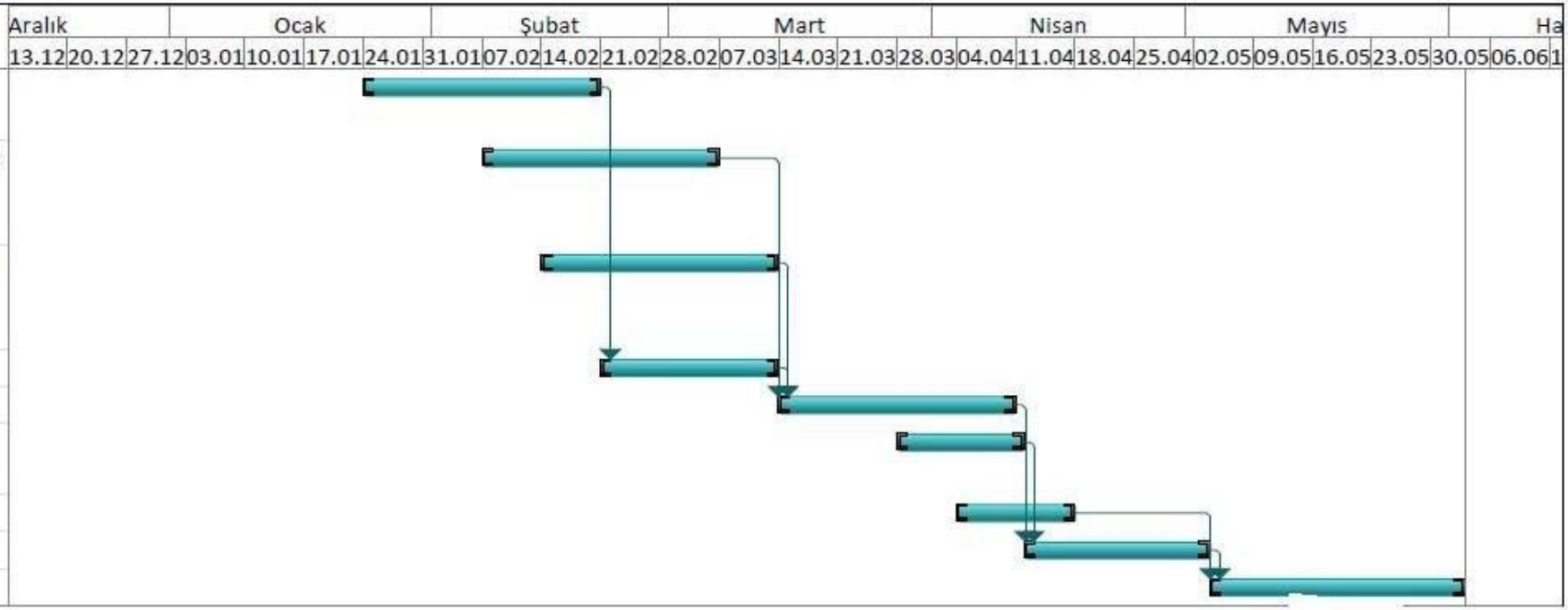

Figure 32: Workload of Second Term

12. Conclusion

This document states the design level approach taken by “Korsan Yazılım” team for the “Context Aware User Interface Project” project. In this document, we make some new statements about design and we got into technical details of project. Our goals are clarified and our work path is started to be drawn. We explained user interface visually. Further information on the technical design is given with detailed explanations of the modules which are supported with class and sequence diagrams. Finally, the progress made by the project team is summarized. We managed to decide many design issues; however we avoid deciding issues which can become very complex, leave that issues to Detailed Design Report. We will start thinking immediately about these deep issues try to clarify what we will do in this and next semester in order to guarantee our success. We will obey our plans as much as possible, although some situations may arise in the process of implementing the project.