INITIAL DESIGN REPORT

TARGET TRACKING USING SEISMIC SENSORS

Group Name & Members:

BG_S3

Edona Fasllija

Erdoğan Cem Evin

Serkan Yanar

Umut Erdal

Table of Contents

1. Introduction

This document is the Initial Design Report of the Target Tracking by Using Seismic Sensors project. The first task of this document will be reviewing system in terms of problem definition,solution, main design issues and then decomposing the system into components that are structurally and functionally distinct from each other in order to provide a proper modularization of the system. After decomposing the system, we will describe each component's responsibilities and the user interface in detail. The document ends with a plan of the implementation of the system.

1.1. Problem Definition

In this project, we propose a solution to the problem of motion tracking within an area where seismic sensors are previously deployed. Target tracking has a wide range of military applications. The study of this problem is based on signal processing and analysis. Hence, we will try to solve the problem of position determination over a field using by means of wireless sensor networks. sensor networks. Seismic sensors provide a reliable sensing solution to the problem of real time object tracking by analyzing the waves the source signal object generates while it moves over the ground.

1.2. Purpose

In this document, a design model for the target tracking system with its architectural, interface, component level and deployment representations will be described. The components and their dependencies of the design model will be clearly described with the help of diagrams providing a comprehensive design documentation which will be further improved in the Detailed Design Report before the implementation.

1.3. Scope

In this document, we will specify the architecture and initial design for the 'target tracking by using seismic sensors system'. We will describe each component by giving its identification, responsibilities, purpose, function, subordinates, dependencies and interfaces. Further explanation of the interfaces (methods and attributes) will however be left to the Detailed Design Report.

1.4. Overview

In the first chapter, the problem definition, purpose and scope of this document are described. Information about terms, acronyms and references which will use on our project will also be provided.

In the second chapter, there is a general description of the software system including information about its functionality, dependencies, limitations etc. Moreover, the goals, objectives and benefits of our project will be explained in this chapter.

In the third chapter, necessary information about design assumptions, dependencies and constraints together with the goals, guidelines and priorities for design of the system's software will be described.

The fourth chapter will describe how the information domain of our system is transformed into data structures. We will describe how the major data or system entities are stored, processed and organized. We will also provide an alphabetical list of the system entities.

In the fifth chapter, we will provide a pictorial representation (UML component diagrams), to show the major subsystems, data repositories and their interconnections. We will explain the relationships between the modules to achieve the complete functionality of the system. There will be a decomposition of the subsystems in the architectural design.

In the sixth chapter, we will explain how the user will be able to use expected features. We will explain the functionality of the system from the user's perspective.

The seventh chapter will give information about the libraries and tools which we are using. The eighth chapter will describe the tentative time plan and the nineth chapter will conclude the document.

1.5 Definitions and Abbreviations

GIS : Geographic Information System
GPS : Global Positioning System
ORDBMS: Object-Relational Database Managment System
UI : User Interface
GUI : Graphical User Interface
GPS : Global Positioning System
Jinput : Java library used to process Joystick input

SRS : Software Requirement Specification

IDR : Initial Design Report

DDR : Detailed Design Report

1.6. References

1. Geophone specification provided at http://www.oyogeospace.com/technologies.htm.
2. "Geographic Information Systems as an Integrating Technology: Context, Concepts, and Definitions". ESRI. http://www.colorado.edu/geography/gcraft/notes/intro/intro.html. Retrieved on31 October 2011.
3. http://www.dis.uniroma1.it/~iocchi/ARGOS/docs/NNMHTracking.pdf
4. https://researcher.ibm.com/researcher/files/us-the/HeEtal10MILCOM.pdf
5. www.alphatechinc.com
6. http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA409361
7. www.aselsan.com.tr
8. http://www.libelium.com/products/waspmote
9. http://wiki.eclipse.org/index.php/Rich_Client_Platform
10. http://www.java.com/
11. http://www.postgresql.org/
12. http://java.net/projects/jinput/
13. http://www.java2s.com/Open-Source/Java-Document/PDF/PDF-Renderer/com.sun.pdfview.htm

2. System Overview

In this section, we will give a general description of our system. First, we will describe the main goals of our system. Then we will explain interfaces that the system will provide and functionality that supported by these interfaces. Then we will look closer to the system and its sub components.

In the end of our project, we will release a software that will be used for target tracking using seismic sensors. The software will compute the data that come from sensors (virtual or real) and produce a predicted path for target. In addition, it will be able to analyze each tracking session and give out detailed report that contains graphics and diagrams.

Our system will provide three interfaces; user interface, joystick interface and sensor

interface. The user interface will be the control panel of our software and it will provide some functions for managing the system and give visual feedback of these functions. The user of our software will be able to determine in which mode the software will run and to switch between these modes in run time. These modes will be of two types, “visual simulation using real sensors” and “visual simulation using virtual sensors”. In “visual simulation using real sensors” mode, our software will listen seismic sensors which will be located previously by using sensor interface and track targets relying on the data that comes from these seismic sensors. During this process, our software will also do a real-time simulation of the predicted path.

In “visual simulation using virtual sensors” mode, our software will provide an interface to user to locate virtual sensors. When the user finishes locating virtual sensors, our software will start listening joystick input using joystick interface and track target which will be controlled by user with a joystick or another joystick-like device. Like “visual simulation using real sensors” mode, our software will do a real-time simulation of the predicted path and in addition to “visual simulation using real sensors” mode, it will also simulate real path of the target hence the user will be able to see the difference between real and predicted paths.

The sensor interface will use Waspmote Gateway to communicate with seismic sensors. This interface will listen to data that come from seismic sensors continuously and forward it to our system after converting data to human-understandable form. This interface will only be used in “visual simulation using real sensors” mode.

The joystick interface will need a joystick or other joystick-like device to be connected to our system. In case of not recognition of the device, the user will be able to assign movements his/her device's keys. This interface will only be used in “visual simulation using virtual sensors” mode.

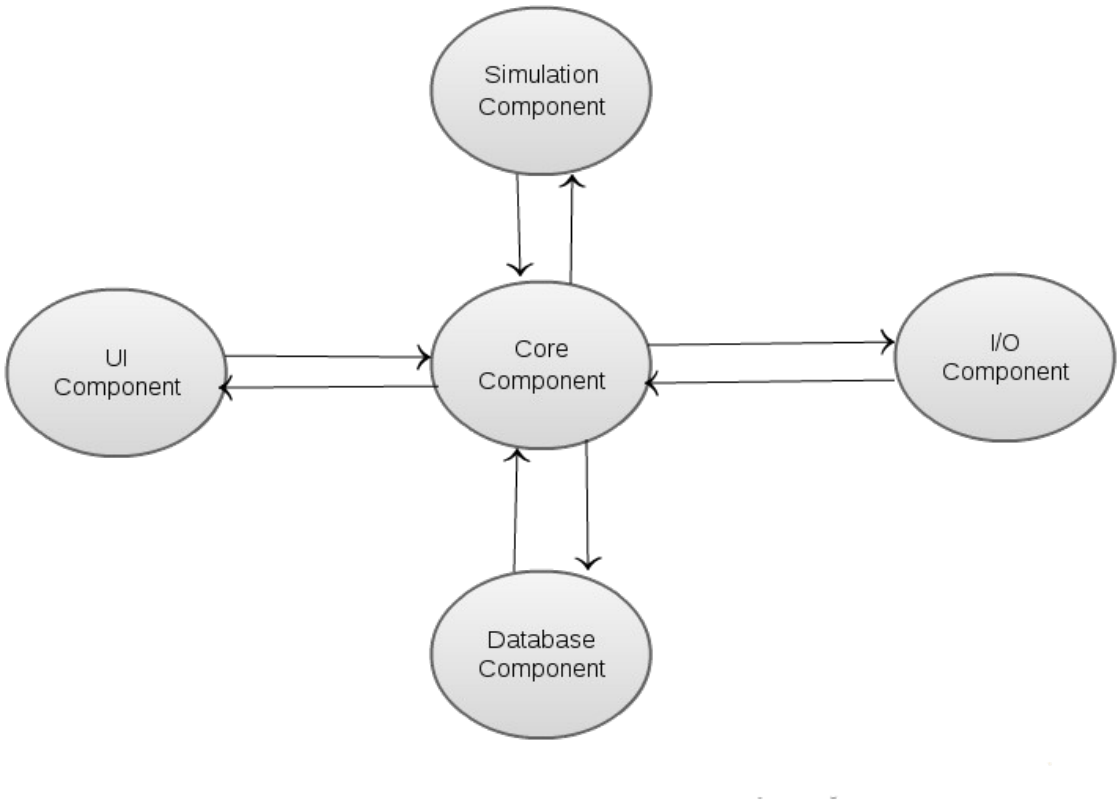

Figure 1: System overview

Our system will consist of five main components, user interface component, I/O component, database component, simulation component, core component. User Interface component will manage communication between user and the system. It will provide a graphical interface to user to control the system and it will also visualize path simulations.

I/O component will handle joystick or other joystick-like devices. The main purpose of this component is to convert raw device data into suitable form for other components.

Database component will handle communication between system and database management system. All other components will communicate with database management system through this component.

The main purpose of the simulation component is to predict a path for target using input data which may come from either real seismic sensors or virtual sensors and then to pass the simulation data to user interface component.

The duty of core component will be to construct all of the other main components and handle communication between them. In other words the core component is operator of the system. Every component will communicate with other components through core component.

3. Design Considerations

In this section we will give information about our design approach. First, we will describe assumptions, dependencies and constraints of our software, then we will explain the goals of our project.

3.1. Design Assumptions, Dependencies and Constraints

In this section, we will provide characteristics of users of our software. Then we will explain some software and hardware dependencies of our software.

Our software is being developed as result of the request of Turkish military company ASELSAN, hence our design highly depends on their needs. This situation causes some hardware and software dependencies which we will explain later. Since the only user of our software will be ASELSAN, we do not have to worry about support a wide range of users and we can be sure of the technical knowledge of the users of our software.

In our project, we will use seismic sensors and gateways of Waspmote which are provided by ASELSAN and, because of this we will develop sensor interface of our software according to application programming interface of these Waspmote cards. The other hardware dependency of our project is joystick devices that will be used for virtual simulation. Since there is not a limitation on which device we will use, initially we will support classic analog joysticks and in case of usage of other types of joysticks, keyboards etc, we will provide a configuration interface for users where they can assign actions to their device's keys.

Our software will be developed using Java programming language, hence we will use object oriented approach for our design. The necessity of communication between the classes of our software caused a problem. We have solved this issue by deciding to use singleton pattern. The communication of our software will be established by calling related methods of target classes' instance.

The last constraint of our project occurs at developing our path generating algorithm due to insufficient number of seismic sensors. We have only one seismic sensor to try, hence we will develop our algorithm relying on the data that will come from virtual seismic sensors. The data that virtual sensors will produce will be similar to data that come from real sensor but since measurements we will do in the field may change due to air conditions, earth type etc. we do not guarantee an error-free result.

3.2. Design Goals and Guidelines

In this section we will explain the goals of our project and give guidelines to reach that goals.

The first main goal of our project is to generate a realistic path from the data that come from seismic sensors. As a first step, we will implement our path generating algorithm using Kalman Filter and then we will try to make our generated path error-free by changing and making additions to our algorithm. Since our path generation algorithm will work real_time, we may need to fasten our algorithm.

The second main goal is efficiency of our software. After we finish the back end programming of our software, we will optimize our code to improve efficiency. We will mainly focus on decreasing CPU usage.

The last but not primary goal of our project is to achieve simplicity for user interface. The all elements of user interface except path visualization will be implemented later and be tested for best user experience.

4.Data Design

4.1. Data Description

This section provides a description of all the data structures of the information domain of the target tracking system, including internal, global, and temporary data structures.

4.1.1. Internal software data structures

The data structures that are passed among the components are described below.
All the components will be able to send a Log entry (string) data to the Core component which eventually will call the method responsible of writing to the log together with the string to be written to the Log Component.
The I/O component will send the a position array from the joystick which the Core component will handle and call the function responsible for generating the path from joyStick input of the Simulator Component together with the array of positions inputed with the joystick.
The Sensor component (both real and simulated) will send alarm strength and position array to the Algorithm component.
The UI component, which is itself event-driven will send update query parameters to the Core for

every user command such as adding a sensor , changing a sensor location, starting or ending a track, toggling the visibility of a layer , generating a report etc. The Core component will then call the corresponding methods of the Report and Database component.
The Report component will send the report information as two date limits and the Database will return the result set after the execution of the corresponding select query.

4.1.2.Global data structures

The data structures that are available to the majority of the components are described below:
The Core Class will be a singleton class, since it is appropriate to have only one instance of the Core Class and it provides a global point of access to the object.
The Log class will be a class of static functions so that all the components will be able to write the changes made by them and record them in the log.

4.1.3. Temporary data structures

The files created only for temporary use are the report files (pdf or doc). These files will not be persistent, unless the user selects to save the report document.

4.1.4. Database description

The database created as part of the target tracking application is described below:

4.1.4.1. Data Entities

The data descriptions of each of these data entities is as follows:

Alarm Data Entity

Data Item	Type	Description	Comment
ID	Integer	ID of an alarm	
SensorID	Pointer	Sensor entity	Used as foreign key to the Sensor entity.
Strength	Float	Strength of an alarm sent	A number from 0-1024 that specifies the strength of an alarm sent.
TimeStamp	Date	Time of receiving the	

		signal	

Sensor Data Entity

Data Item	Type	Description	Comment
ID	Integer	ID number of a sensor	
Location	Pointer	Position of a sensor in the map	Points to a location entity element.
State	Boolean	On/Off state	

SensorLocation Data Entity

Data Item	Type	Description	Comment
SensorID	Pointer	The sensor this position belongs to.	
Latitude	Float	Latitude of the selected map	
Longitude	Float	Longitude of the selected map	
XComponent	Float	X position in the map	
YComponent	Float	Y position in the map	
ZComponent	Float	Z position in the map	

TrackLocation Data Entity

Data Item	Type	Description	Comment
TrackID	Pointer	The track this location is part of.	
Latitude	Float	Latitude of the selected map	
Longitude	Float	Longitude of the selected map	
XComponent	Float	X position in the map	
YComponent	Float	Y position in the map	
ZComponent	Float	Z position in the map	
TimeStamp	Date	The time of the position recorded	Needed to be able to reconstruct a track after its been recorded.

RealTrack Data Entity

Data Item	Type	Description	Comment
ID	Integer	Id of the track input	
Type	Boolean	JoyStick or real input	0 for real, 1for joystick input
SessionID	Pointer	Id of the session	A session is kept for a specific time interval

GeneratedTrack Data Entity

Data Item	Type	Description	Comment
ID	Integer	Id of the track generated	
SessionID	Pointer	Id of the session	A session is kept for a specific time interval

Session Data Entity

Data Item	Type	Description	Comment
ID	Integer	Id of the session	
StartTime	Date	Starting time of the session	A session is kept for a specific time interval
EndTime	Date	End time of the session	

4.1.4.2. Entity-Relationship Diagram

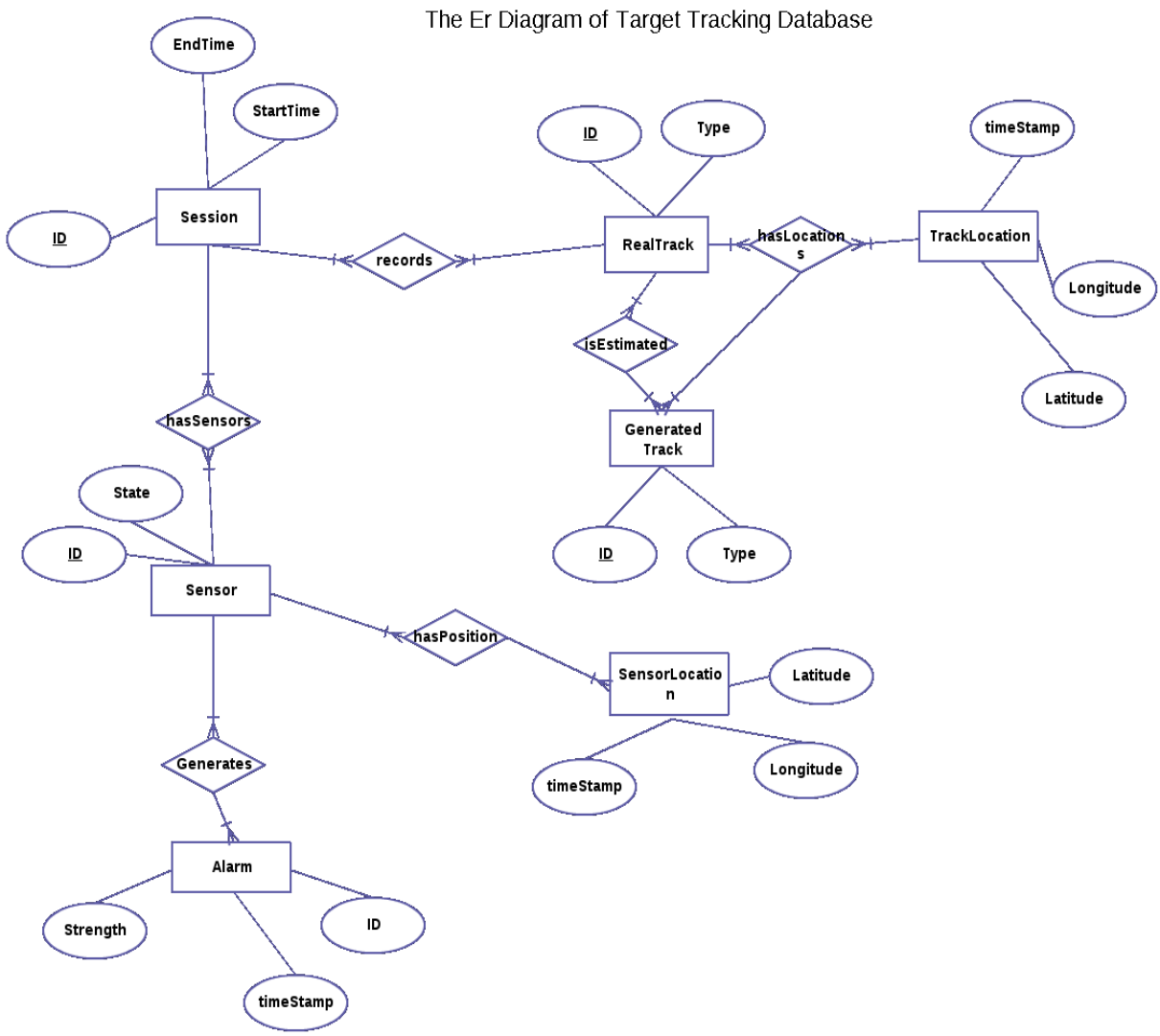

Figure 2: ER Diagram

4.1.4.3. Data Relationships

The Logical Structure of the data to be stored in the Target Tracking Database is as follows:

A RealTrack or GeneratedTrack data Entity is in aggregation type of relationship with TrackLocation data Entity.

A Track(real or generated) will be composed of a continuous array of TrackLocation Objects ordered according to their timestamps.

A Sensor is in association type of relationship with Alarm entity, each Alarm is associated with a sensor id belonging to the sensor the alarm is sent from.

Finally each track, and all the information it includes is associated with a session data entity, keeping track of all the tracks recorded and generated during a specific time interval.

4.2. Data Dictionary

The system entities or major data along with their types and descriptions are listed in this section. If Since we provided an OO description, we list here the objects and their attributes, the methods and the method parameters.

Name	**Type**	**Refer to Section**
layers	Component	5.2.1.
updateSensorLayer	Method	5.2.1.
updateTrackLayer	Method	5.2.1.
displayReport	Method	5.2.1.
addSensor	Method	5.2.1.
removeSensor	Method	5.2.1.
replaceSensor	Method	5.2.1.
enableSensor	Method	5.2.1.
disableSensor	Method	5.2.1.
addTrack	Method	5.2.1.
removeTRack	Method	5.2.1.
startTrackRecording	Method	5.2.1.
stopTrackRecording	Method	5.2.1.
generateReport	Method	5.2.1.
ui	Component	5.2.2.
db	Component	5.2.2.
simulation	Component	5.2.2.
io	Component	5.2.2.
forwardData	Method	5.2.2.
log	Method	5.2.3.
logError	Method	5.2.3.
device	Component	5.2.4.
startProcess	Method	5.2.4.
stopProcess	Method	5.2.4.

sendData	Method	5.2.4.
intervalStartTime	Data Entity	5.2.6.
intervalEndTime	Data Entity	5.2.6.
reportFormat	Data Entity	5.2.6.
getIntervalStartTime	Method	5.2.6.
setIntervalStartTime	Method	5.2.6.
getIntervalEndTime	Method	5.2.6.
setIntervalEndTime	Method	5.2.6.
getReportFormat	Method	5.2.6.
setReportFormat	Method	5.2.6.
generateReport	Method	5.2.6.
createQueryStatement	Method	5.2.5.
createUpdateStatement	Method	5.2.5.
executeQuery	Method	5.2.5.
executeUpdate	Method	5.2.5.
alarmlevel	Data Entity	5.2.10. ,5.2.11.
batterylevel	Data Entity	5.2.10.
send_alaram_level	Method	5.2.10. ,5.2.9.
send_battery_level	Method	5.2.10. ,5.2.9.
set_alarm_level	Method	5.2.11.
get_software_sensor_alarm_level	Method	5.2.7.
send_alarm_levels_of_real_sensor	Method	5.2.7.
send_alarm_levels_of_software_sensor	Method	5.2.7.
get_location	Method	5.2.7.
get_track	Method	5.2.7.
current	Component	5.2.8.
previous	Component	5.2.8.
currentTrack	Component	5.2.8.
set_current_location	Method	5.2.8.
generate_track	Method	5.2.8.

5. System Architecture

5.1. Architectural Design

We have designed our software following the singleton design pattern principles. We have developed a modular system structure.

The system's core is the Core component. The main communication among the components are handled and organized through the core. The communication is handled by the core by calling the instances of each component.

We have used aspect oriented approached in our logging system.

The pictorial presentation of the system structure is shown below.

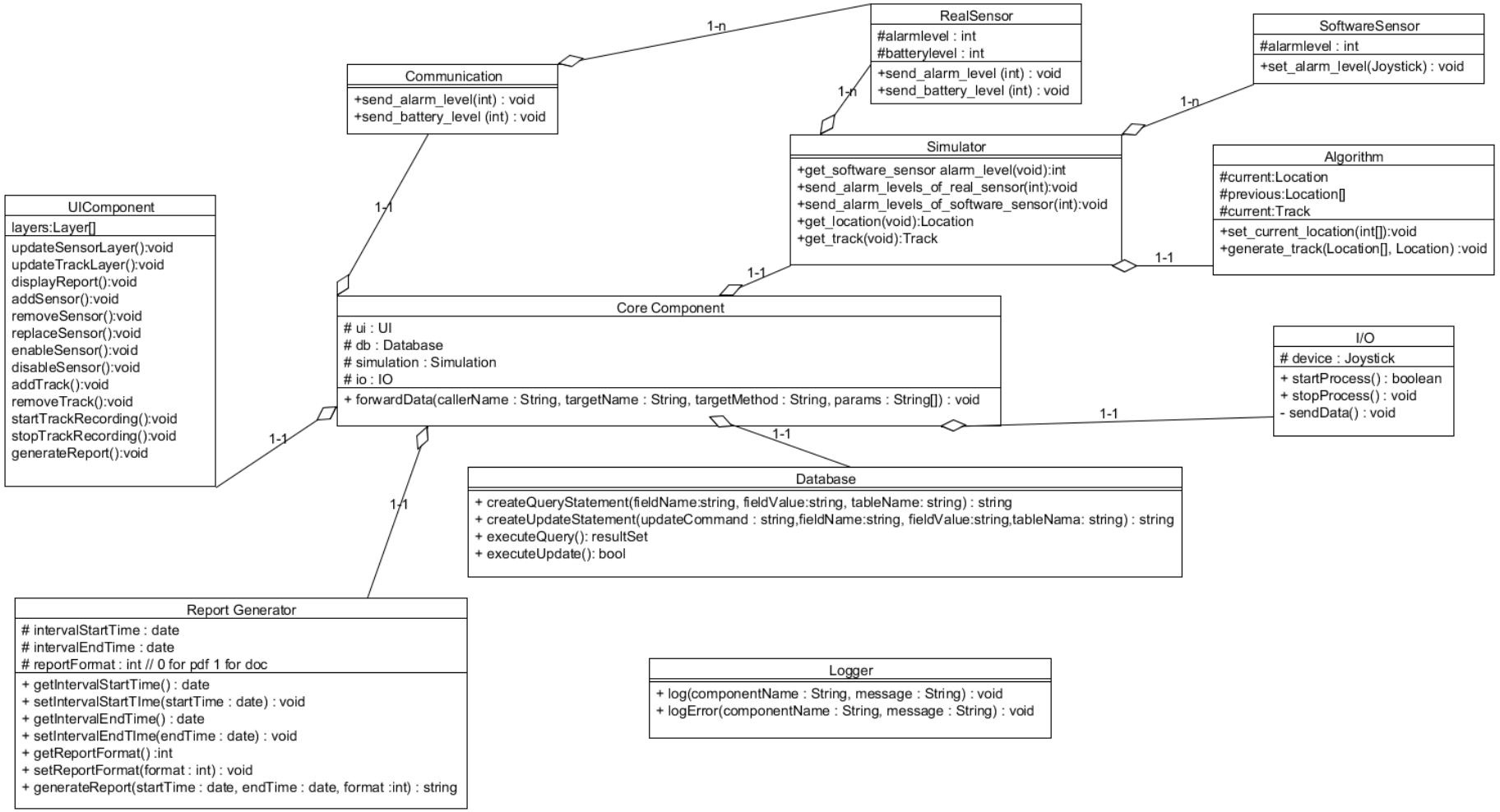

Figure 3: Class Diagram

5.2. Description of Components

5.2.1. UI Component

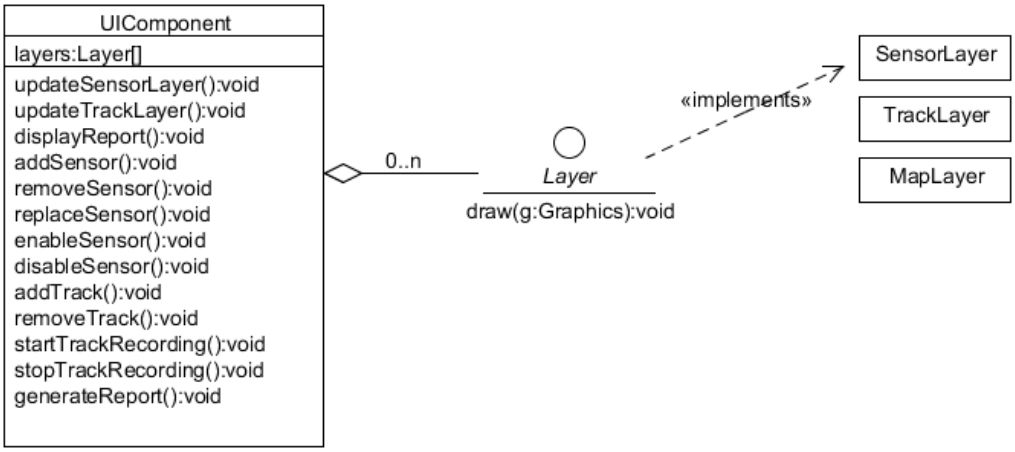

Figure 4: Class Diagram

The main purpose of the UI component is to handle the communication between the user and the core functions and provide a GUI.

5.2.1.1. Processing Narrative for UI Component

The functionality of the UI component is to handle the system calls from the user interface. This component includes map, sensor, track layers to show the geographic map, sensors and tracks visually. It also includes menus to call core and report functions.
This component is responsible for communicating with the Core and Report components.

5.2.1.2. UI Component Interface Description

The UI Component has interfaces with the Core and Report component. The Core component uses the UI interfaces' input interfaces to update the changes in the state of the current session, such as a track, sensor or a map change. The input interfaces for the report component is for showing the generated report in the GUI.
The output interfaces of the UI Component is for calling the Core component functions, such as add a new sensor or a new track input, and for communicating with the Report component for report generation.
Interfaces:
These functions will be called by Core and Report components:
updateSensorLayer()
updateTrackLayer()
displayReport()

These functions will call the related functions of Core and Report components.

addSensor()

removeSensor()

replaceSensor()

enableSensor()

disableSensor()

addTrack()

removeTrack()

startTrackRecording()

stopTrackRecording()

generateReport()

5.2.1.3. UI Component Processing Detail

The layers in the UI Component contain visual representation of the current state of the program session, such as the places of the sensors, the current points in the track or the current coordinates chosen for the map. When the user triggers this component through menus, the change is reflected to the GUI and the related Core or Report function is called.

5.2.1.4. Dynamic Behavior for UI Component

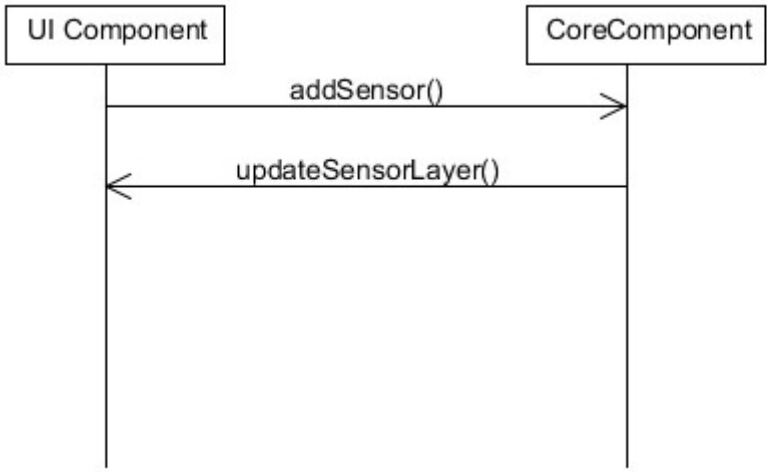

Figure 5: Sequence diagram for adding a sensor

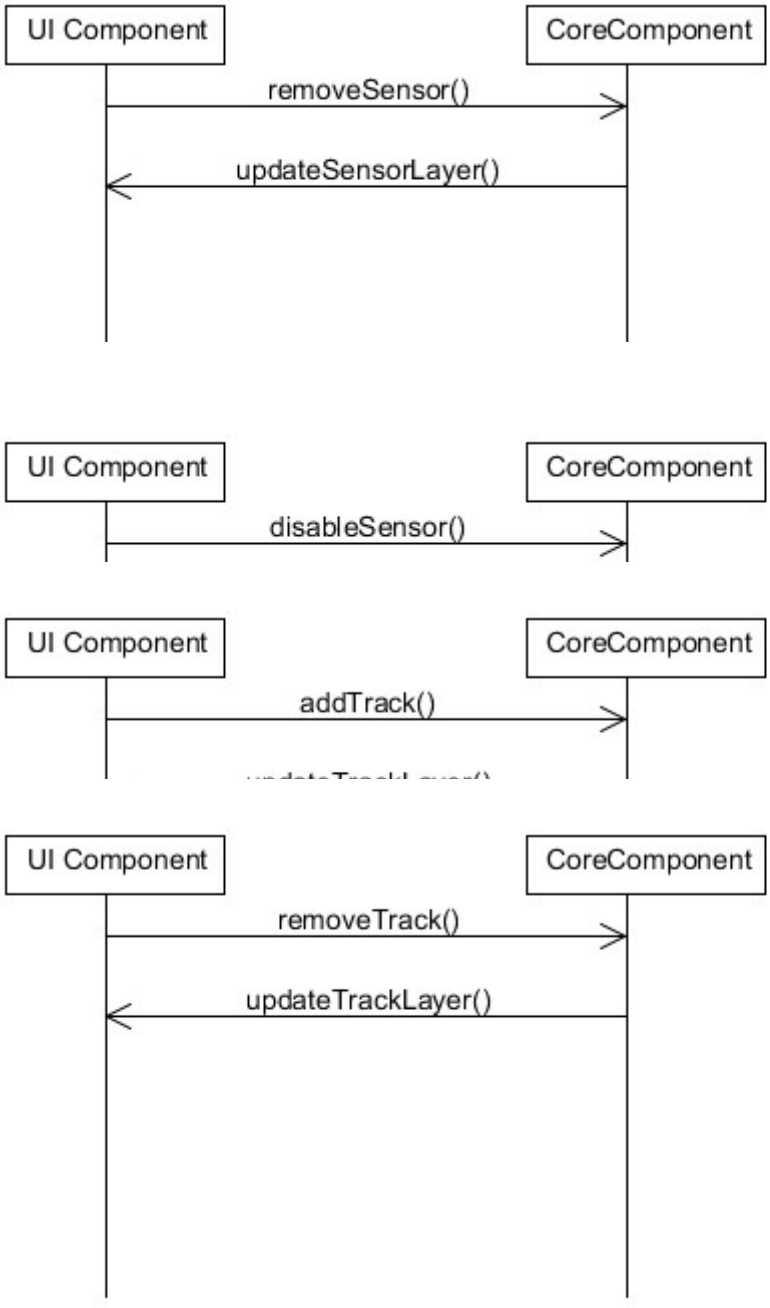

Figure 10: Sequence diagram for removing a track

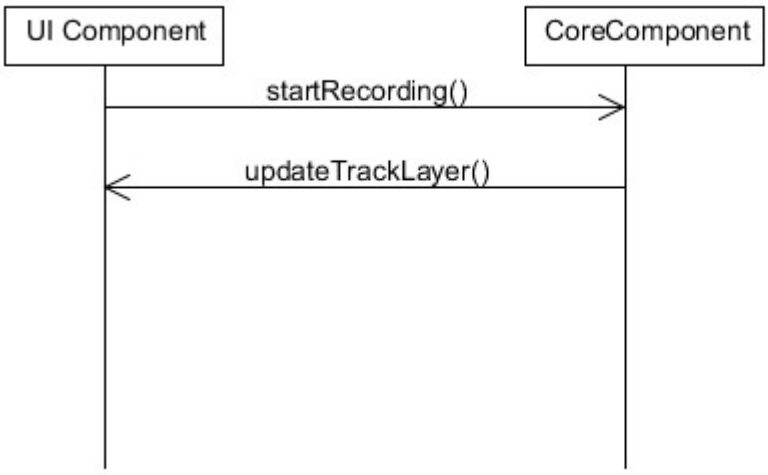

Figure 11: Sequence diagram for starting to record a track

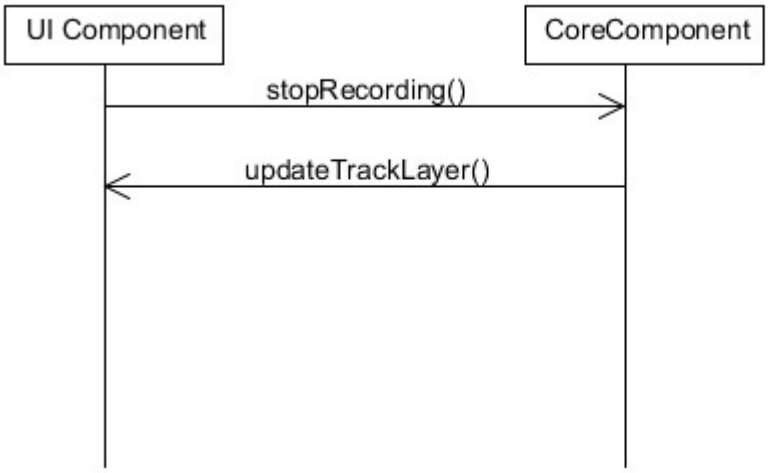

Figure 12: Sequence diagram for stop recording a track

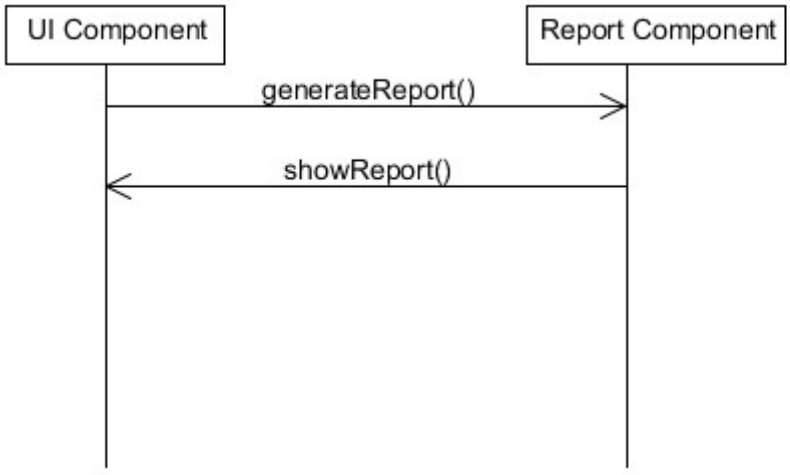

Figure 13: Sequence diagram for generating a report

5.2.2. Core Component

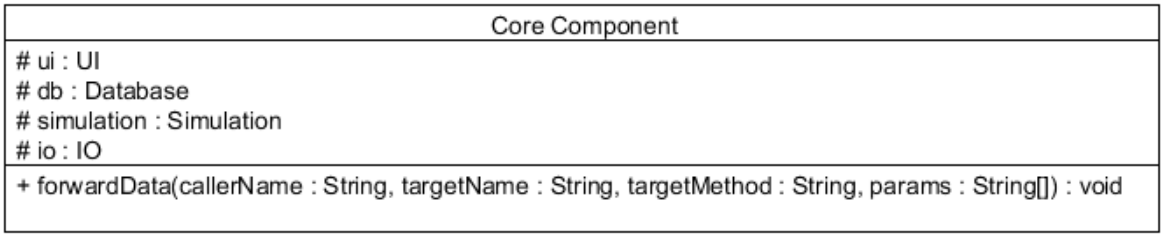

Figure 14: Class Diagram

This will be the main component of the software. It will manage communications between software components. Every component in the software except core component will communicate with other components indirectly through core component. Hence the main purpose of this component is to forward messages and requests to related components. This component will also construct other components at the beginning of the program.

5.2.2.1. Processing narrative for core component

The communication between core component and other components should be managed by calling forward method of core component. The core component will get caller component's name, target's name, target's related method name, and parameters of this method and then it will call related method of target component and pass specified parameters to it.

5.2.2.2. Core Component Interface Definition

void forwardData(String callerName, String targetName, String methodName, String parameters) – This method will call related method of target component with specified parameters.

5.2.2.3. Core Component Processing Detail

The processing detail of this component is explained step by step below:

1. Construct software components
2. Wait for forwardMethod function to be invoked

3. Parse parameters and call related method of target component

5.2.2.4. Dynamic Behavior of Core Component

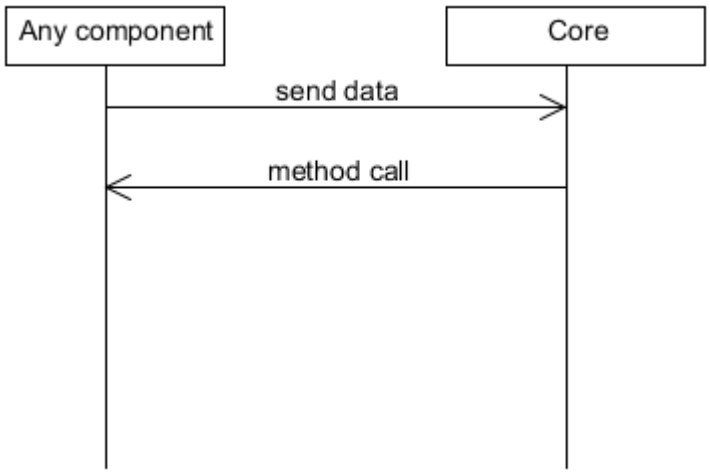

Figure 15: Sequence diagram

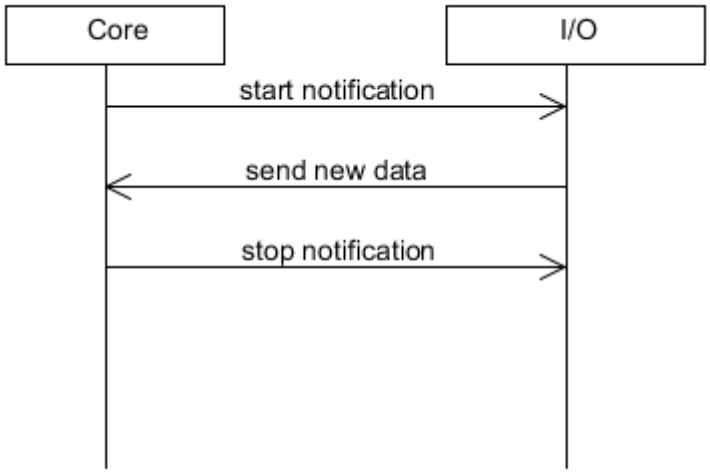

Figure 16: Sequence diagram

5.2.3. Logger Component

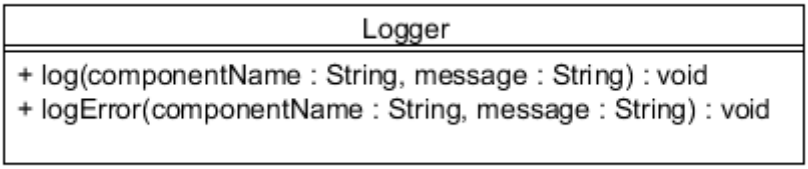

Figure 17: Class Diagram

This component will be used to log every action and error of the software. The component will be standalone component and be reachable from every other component in the software.

5.2.3.1. Processing Narrative for Logger Component

When this component is called, it should open a log file and log the messages it received to that file with current date and time. Since logs can be of two types – error and action-, the name of the file which is opened by this component depends on the type of the log message. Each session should have its own unique log file but error messages should be held in a common error log file.

5.2.3.2. Logger Component interface definition

void logError(String componentName, String message) – It will be used for logging errors. It will

open error log file write error message, name of the caller component, date and time to that file.

void log(String componentName, String message) – It will be used for logging actions. It will open log file with current session id and write action message, name of the caller component and current date and time to that file.

5.2.3.3. Logger Component Processing Detail

The processing detail of this component is explained step by step below:

1. Receive log request
2. Get log type
3. Get session id from core component
4. Open log file containing session id in its name
5. Write log message, sender component name, date and time into the file
6. Close file

5.2.3.4. Dynamic Behavior of Logger Component

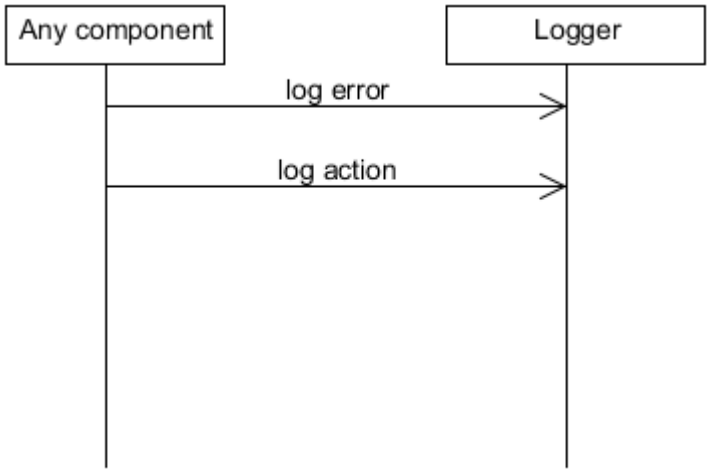

Figure 18: Sequence diagram

5.2.4. I/O Component

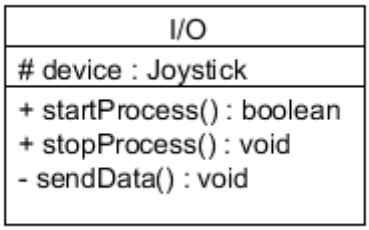

Figure 19: Class Diagram

The software will be able to listen input from a joystick-like device to simulate real path when working with virtual sensors. When the software is running in virtual mode, this component will wait for input from joystick-like device and convert it into human understandable form for simulation component.

5.2.4.1. Processing Narrative for I/O Component

The component should start processing when it receives a notification directly from core component (which triggered by UI component). First of all, it should check whether the connected device is suitable for listening. After this check it should listen from this device and convert these input data to human understandable form as x and y axis values. Then the component should dispatch an event for core component to send converted human data to simulation component.

5.2.4.2. I/O Component Interface Description

boolean startProcess(): It will be called by core component. This function will check whether the device is suitable for listening and returns true or false due to check result. If device is working properly it will start listening input from device.

void stopProcess(): It will be called by core component too. This function will stop listening from input device.

void sendData(): For every converted input data the component will send these new data to core component. It basically manages communication between I/O component and core component.

5.2.4.3. I/O Component Processing Detail

The processing detail of this component is explained step by step below:

1. receives notification from core component
2. checks input device
3. initializes x and y variables to zero
4. starts listening from input device
5. receives data from input device
6. increases or decreases x and y variables due to input data
7. sends new data to core component
8. if not received stop notification goes to 5
9. stop listening from device

5.2.4.4. Dynamic Behavior of I/O Component

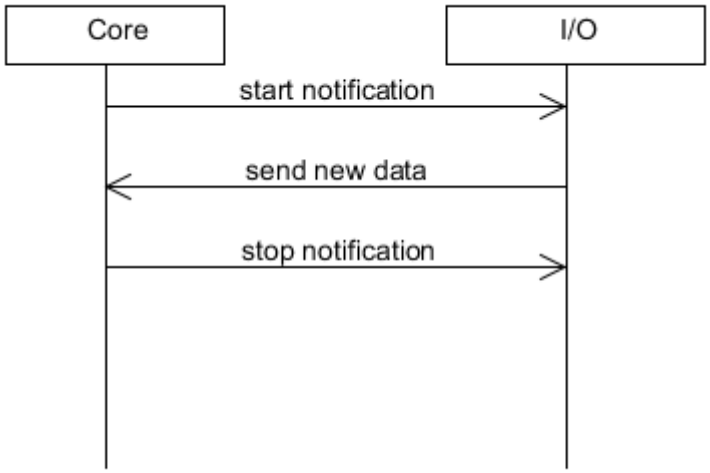

Figure 20: Sequence diagram

5.2.5. Database component

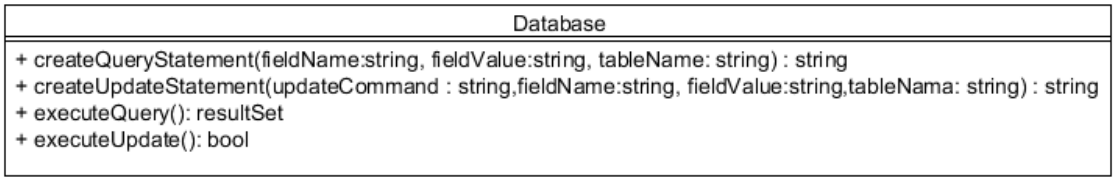

Figure 21: Class Diagram

The Database component is described in the section below, together with its responsibilities, input and output interfaces, processing details, dynamic behavior and the corresponding diagrams (class, interaction etc.)

5.2.5.1. Processing narrative for Database component

The purpose of the Database component is to organize the database in order to provide an efficient way of accessing the data items of the application. This component primarily serves to the Report Generator component and the Core Component.

The Report Generator component will use the Database component in order to retrieve the information required to generate a report.

The Core component will send the Database component the entities to be stored in the database, such as sensors, alarms, real and generated tracks, etc.

5.2.5.2. Database component interface description

The Database component will not have any modeling attributes. Instead, it will have the intuitive createQueryStatement(), createUpdateStatement(), executeQuery(), executeUpdate(). The input to these methods will be provided by the Core and Report Generator component. The core component will input the entities created during the application's running time. Each sensor along with its location, and the alarms it sends, and the tracks generated using this information are stored in the database.

The Database component will output a result set at the end of the execution of any update/query statement.

5.2.5.3. Database component processing detail

The Database component has no subordinates. Hence a single class diagram for the Report Generator component is provided below:

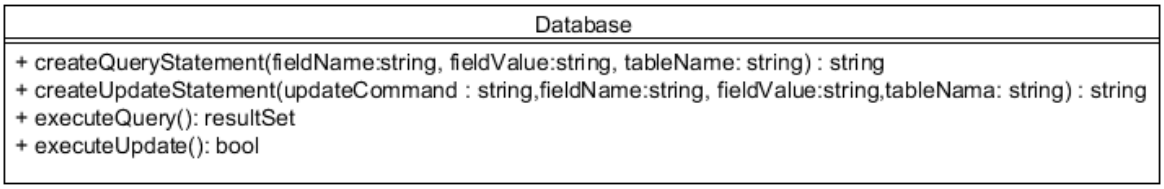

Figure 22: Class Diagram for the Database component

There are some restrictions/limitations for Database component. PostgreSQL will be used as the object-relational database system since it allows storage of position items as longitude and latitude and it also strongly conforms to the ANSI-SQL:2008 standard.

Very frequent join operations performed during the queries required for the report generation is a performance issue for the Database component. Design decisions serving the purpose of increasing the efficiency of these operations will be explained further in the Detailed Design Report.

5.2.5.3.5. Processing detail for each operation of Database component

The processing details (narrative and algorithmic) for the operations realized by the Database component are as follows:

1. The user creates adds/removes a sensor, changes the location of a sensor, or starts/ends the track generation from the UI component. The changes information is sent to the Core Component.
2. The Core component calls the createUpdateStatement and as a result an update query is created according to the entry or field value to be changed.
3. The Database component calls the executeUpdate() method and executes the created update statement. An integer showing the number of the rows and the name of the table affected is returned to the Core Component.

Or alternatively,

1. The user decides to generate a report for a specific time intervalfrom the UI component. The report information is sent to the Report Generator Component.
2. The Report Generator component calls the createQueryStatement and as a result an select query is created according to the sessionIDs of the time intervals for which a report is to be generated.
3. The Database component calls the executeQuery() method and executes the created select

statement. A result set containing the rows and the name of the table created is returned to the UI Component.

5.2.5.4. Dynamic Behavior for Database component

A description of the interaction of the classes is presented. A sequence diagram is presented for the Generate Report use case realized by this component.

5.2.5.4.1. Interaction Diagrams

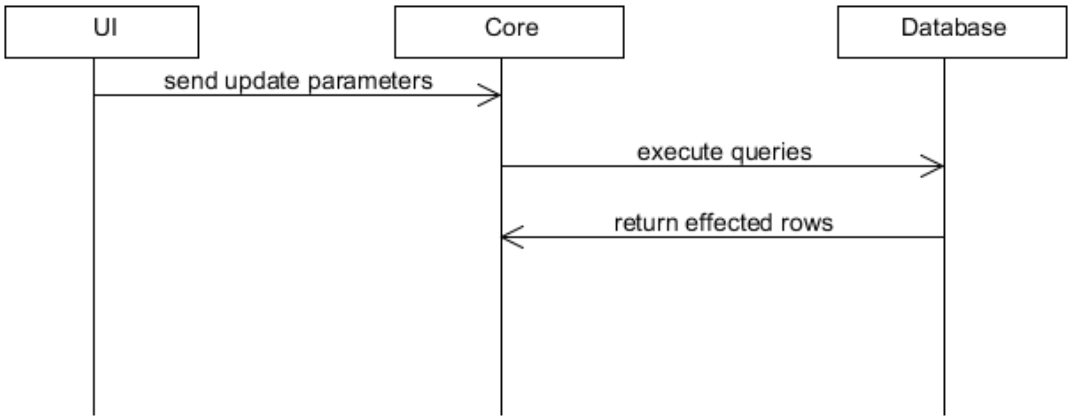

Figure 23: Sequence diagram

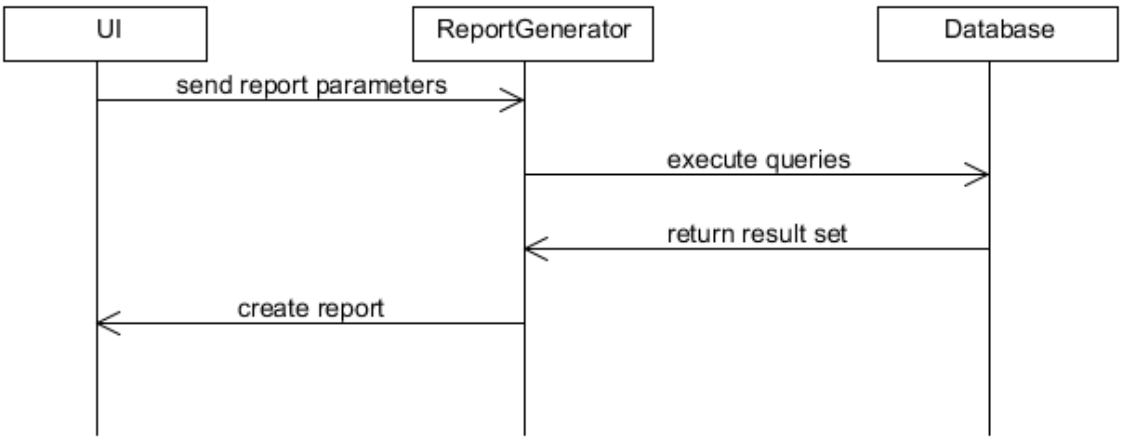

Figure 24: Sequence diagram

5.2.6. Report Generator component

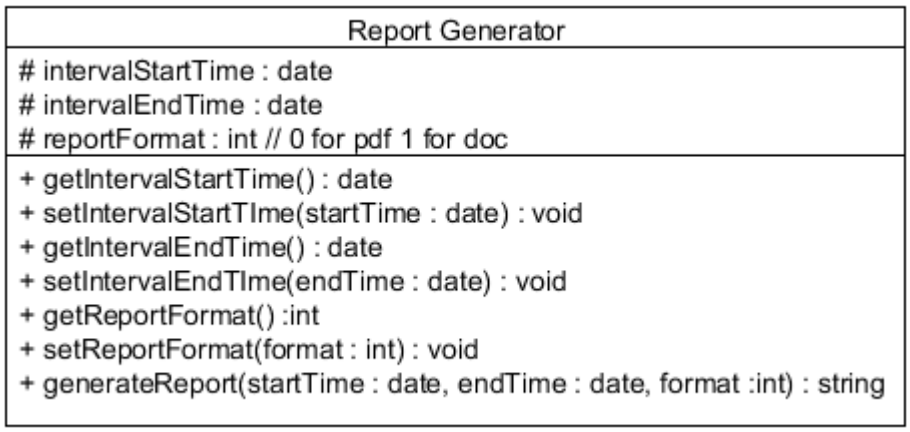

Figure 25: The class diagram of the Report Generator Component

The Report Generator component is described in the section below, together with its responsibilities, input and output interfaces, processing details, dynamic behavior and the corresponding diagrams (class, interaction etc.)

5.2.6.1. Processing narrative for Report Generator component

The Report Generator component is responsible for the report generating function provided to the user.

As described in the SRS, this feature enables the user to view all relevant information of a session, after he/she provides the information related to the report to be generated in the form prompted to him and a document containing information retrieved by the database such as the map together with the sensor locations, the log of the alarms, and the real and estimated tracks so that one can evaluate the correctness of the path generation algorithm.

5.2.6.2. Report Generator component interface description

The Report Generator component will get as input from the user the start and end time of the interval for which the report is to be generated, and the format of the report. The output will be a PDF of DOC file displaying the information about the selected time interval, plotting all the generated tracks along with the real tracks corresponding to that time interval.

For all practical purposes, this component is modeled by the three attributes provided by the user as input, i.e. intervalStartTime, intervalEndTime, and reportFormat. These attributes will be protected and they will have getter and setter methods.

5.2.6.3. Report Generator component processing detail

An algorithmic description of the Report Generator component is provided below.

The Report Generator component has no subordinates. Hence a single class diagram for the Report Generator component is provided below:

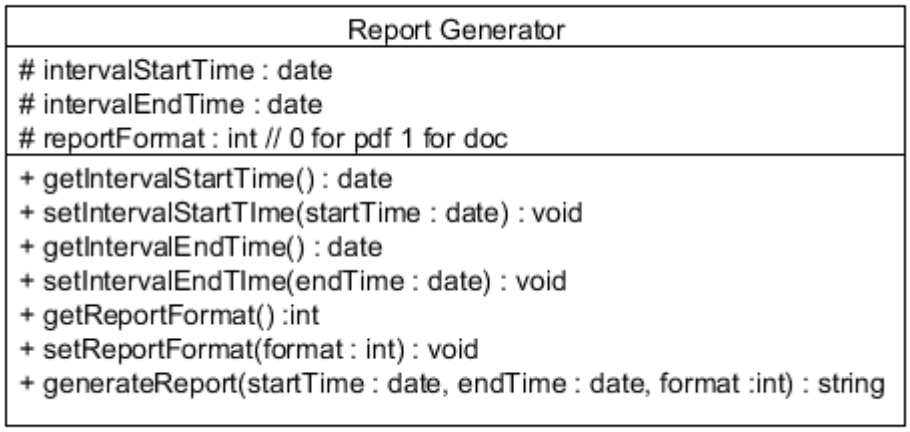

Figure 26: The class diagram of the Report Generator Component

There are restrictions/limitations for Report Generator component. The format of the report can either be PDF or DOC. The time interval selected should not exceed one day time interval and should not be shorter than a session time (the time for a track to be started, ended, and estimated).

5.2.6.3.5. Processing detail for each operation of Report Generator component

The processing details (narrative and algorithmic) for the report generation operation realized by the Report Generator component are as follows:

1.The user inputs the start and end time of the interval of the report, as well as the format of the document from the UI component.

2.The Report Generator component receives these parameters and calls the generateReport(startTime, endTime, format) method.

3.The generateReport() method performs a query to the database in the Database component and selects all the information available in all the tables by using the sessionIDs, their starting and ending times to join them. A result set is returned from the Database to the Report Generator component.

4.A PDF/DOC document is generated using the information retrieved from the database present in the result set returned form the generateReport() method.

5.2.6.4. Dynamic Behavior for Report Generator component

A description of the interaction of the classes is presented. A sequence diagram is presented for the Generate Report use case realized by this component.

5.2.6.4.1. Interaction Diagrams

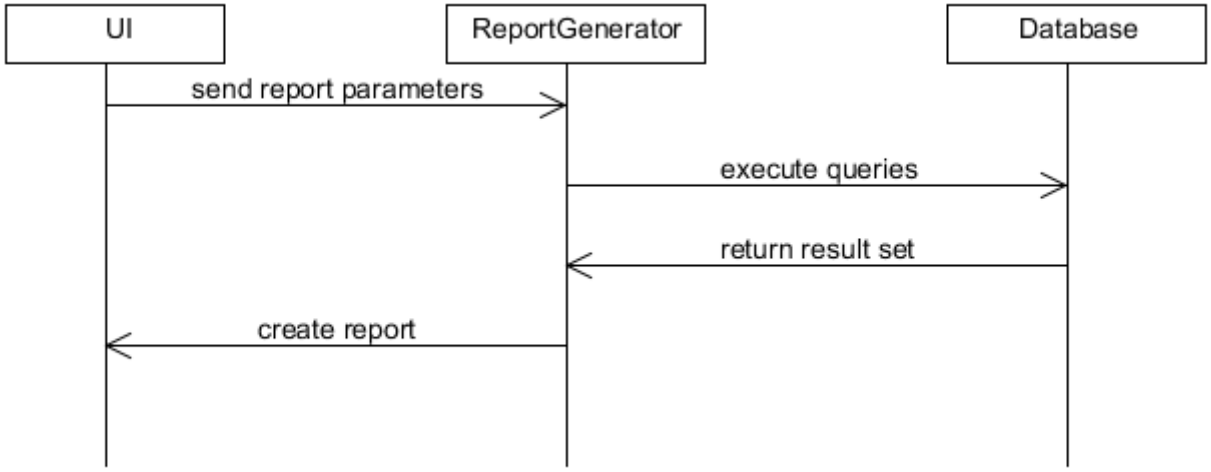

Figure 27: Sequence diagram

5.2.7. Simulator Component

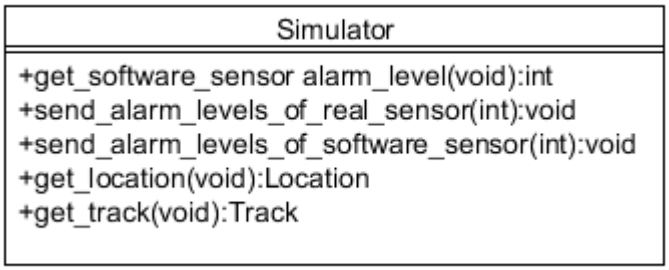

Figure 28: Class Diagram

Simulator component communicates with algorithm component, software sensors and communication components of the system. Simulator has its own clock. This clock generates a signal in every few milliseconds. So, according to the signal from this clock simulator components gets and sends the states of the real/software sensors. It also gets path of the target's motion and position of the target/jinput from algorithm components.

5.2.7.1. Processing narrative for Simulator Component

Simulator receives the new location of the target from algorithm components and it also recieves the track information of the target from algorithm components. The simulator component is responsible to send the states of the real/software sensors to algorithm component. Algorithm stores the previous and current location of the target and generates the path of the target's movement.

5.2.7.2. Simulator Component interface description

This component has the fuctionalities to communicate with algorithm component, software sensors and communication components of the system. It is responsible to get and send the states of the real/software sensors. Also it has functions to get the location of the target/jinput and path of the target's position.

5.2.7.3 Simulator Component processing detail

When the system is on 'simulation mode' and jinput is given by user, simulation component gets the alarm level of the software sensors. If the system is not on 'simulation mode' when a vibration occurs because of the target's motion, communication component sends alarm level of the real sensors. Then, simulator component sends the states of the real/software sensors to algorithm component. Algorithm finds and stores the previous and current location of the target. So, it generates the track of the target by using kalman filter. Simulator receives the new location of the target and path of the target' motion from algorithm component.

5.2.7.4. Dynamic behavior Simulator Component

Trigger: communication component sends real sensors alarm level

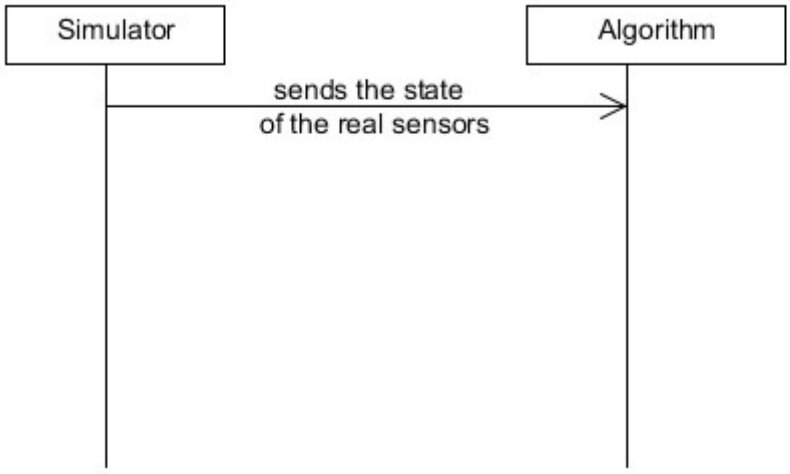

Figure 29: Sequence diagram

Trigger: jinput is given from user

Simulator component -> get the states of the software sensors

Trigger: a signal comes from simulator's internal clock

Simulator component -> get the location of the target/jinput

Trigger: a signal comes from simulator's internal clock

Simulator component -> get the path of the target/jinput motion

5.2.8 Algorithm Component

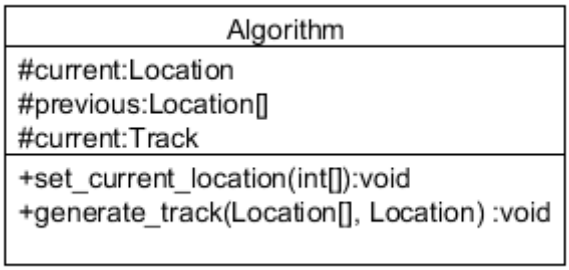

Figure 30: Class Diagram

This component is the heart of our system because it is responsible to find location of the target and generate track of the target's motion by using kalman filter.

5.2.8.1. Processing narrative for Algorithm Component

Using the states of the real/software sensors, algorithm finds the current location of the target. Also it generates the track of the target's motion by using kalman filter. The previous track information and the current location of the target are two necessary data to generate track. So, algorithm component stores this data at its own memory.

5.2.8.2. Algorithm Component interface description

Algorithm component has the fuctionalities to communicate with only the simulator component of the system. The generated locations and track information will be used by simulator component.

5.2.8.3 Algorithm Component processing detail

Simulator component sends the states of the real/software sensors to algorithm. Using only this information algorithm finds the current location of the target. This location information is updated when simulator components sends new states of real/software sensors. In addition, algorithm component of the system stores and uses previous locations of the target to generate the track of the target' motion.

5.2.8.4. Dynamic behavior Algorithm Component

Trigger: Simulator sends the states of the real/software sensors

Algorithm finds the location of the target and generates the track by using kalman filter.

5.2.9 Communication Component

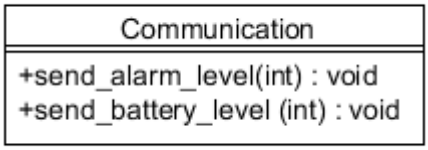

Figure 31: Class Diagram

It is a hardware device that communicates with real seismic sensors and simulator component.

5.2.9.1. Processing narrative for Communication Component

A real seismic sensor sends alarm level of target position or its battery level to the communication component. Then, communication component sends it to the simulator. In other words, the communication component is responsible to send these messages to the simulator component periodically.

5.2.9.2. Communication Component interface description

The communication component has interfaces with simulator component and real sesimic sensor component. It gets information about battery level and position of the target from the real seismic sensor component and sends to the simulator component.

5.2.9.3 Communication Component processing detail

When real seismic sensor generates an alarm signal showing the strength of the vibration, it sends this alarm to the communication component. Moreover, when battery level of a real seismic sensor drops below the critical level, a 'battery level low' message is sent to the communication component. After getting these messages, communication component sends them to the simulator component.

5.2.9.4. Dynamic behavior Communication Component

Trigger: Real seismic sensors send signal,

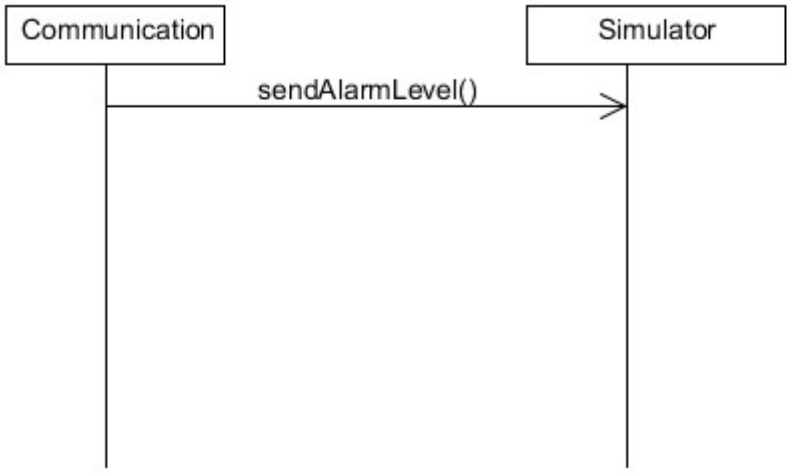

Figure 32: Sequence diagram

Trigger: Real seismic sensors send signal,

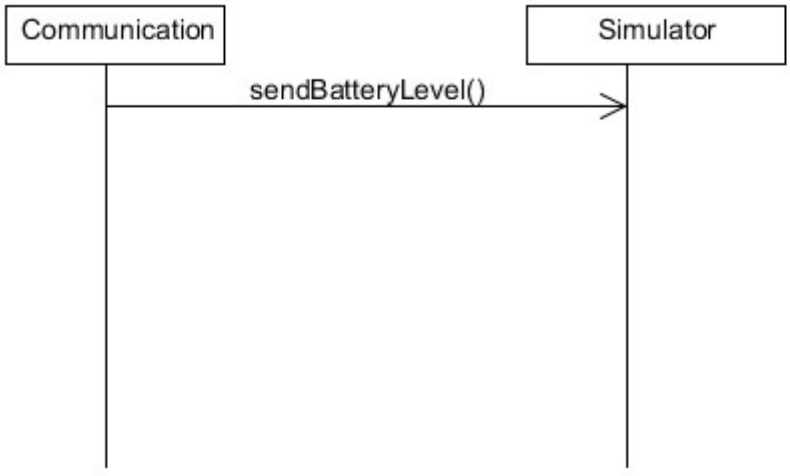

Figure 33: Sequence diagram

5.2.10 Real Seismic Sensor Component

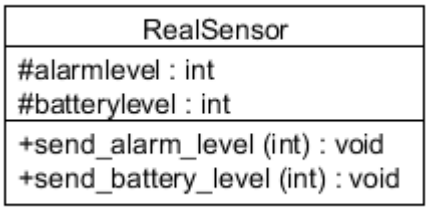

Figure 34: Class Diagram

It is a hardware device that detects and sends information about the target's position. In this project, we will use a few numbers of real seismic sensors to find the target's position correctly.

5.2.10.1. Processing narrative for Real Seismic Sensor Component

Real seismic sensors will be deployed on the field. It has functions that communicates with the system. It is main responsiblity is to send information about the target's position. A sensor is also responsible to send its battery level to the communication component.

5.2.10.2. Real Seismic Sensor Component interface description

The real seismic sensor component has interfaces with communication component. There are two functions in real seismic sensor component. They send information about battery level and position

of the target to the communication component. When battery level of a real seismic sensor drops below the critical level, a 'battery level low' signal is sent to the communication component.

5.2.10.3 Real Seismic Sensor Component processing detail

When there is a vibration because of the target motion, a real seismic sensor detects it. This sensor generates an alarm signal showing the strength of the vibration. Then it sends this signal to the communication component. Moreover, when battery level of a real seismic sensor drops below the critical level, a 'battery level low' warning message occurs at the sensor component. Then this s ensor generates a signal and sends this to the communication component.

5.2.10.4. Dynamic behavior Real Seismic Sensor Component

Trigger: Motion of target,

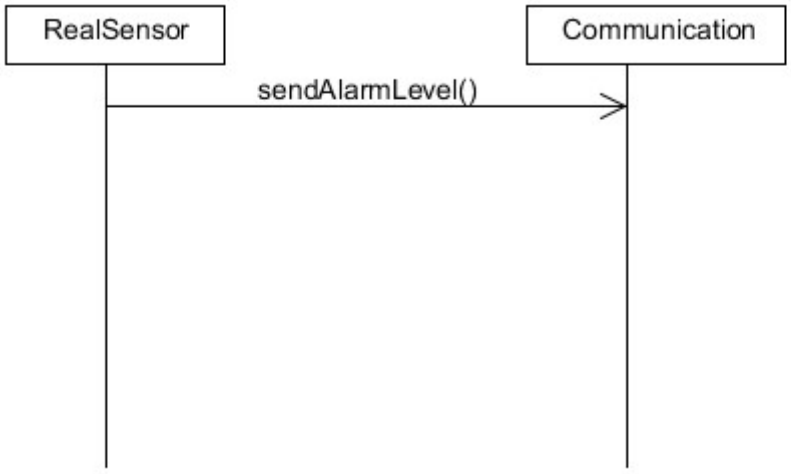

Figure 35: Sequence diagram

Trigger: Battery level drops,

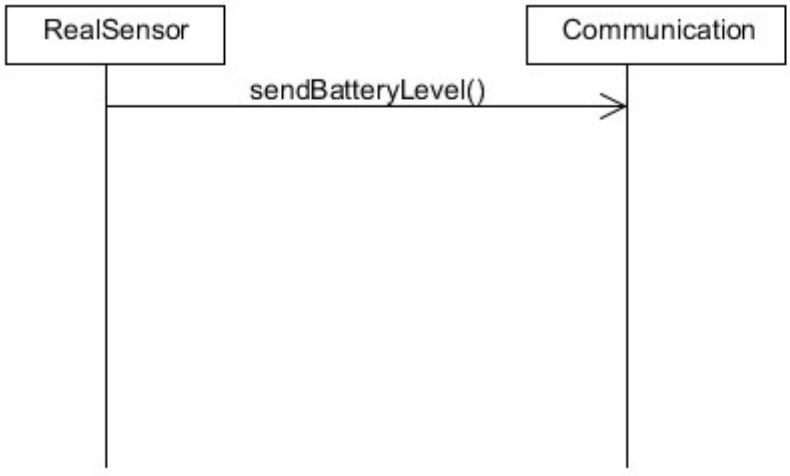

Figure 36: Sequence diagram

5.2.11 Software Sensor Component

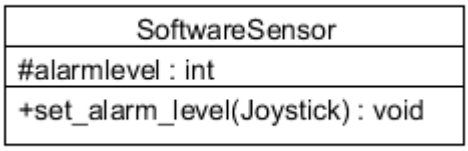

Figure 37: Class Diagram

It is a software that sends information about the position of inputs. When doing a simulation of target tracking, we need a software sensor which is a model of the real seismic sensor.

5.2.11.1. Processing narrative for Software Sensor Component

When user wants to enter the track by using I/O, we need a component that models the real seismic sensors so that we can read this component's alarm level. We will find the simulated path and compare this path with input given by user.

5.2.11.2. Software Sensor Component interface description

The software sensor component has interface with simulator component. Simulator component gets the alarm level of the software sensor component when input is given.

5.2.11.3 Software Sensor Component processing detail

When the system is 'simulation mode' the user gives input to the system using I/O, core component of the system gets the location information from I/O device. Core component sends the location information to the software sensors. After getting that, every software sensor is responsible to set the alarm signal level. According to the distance between the location of the software sensor and given location, every software sensor sets its alarm level.

5.2.11.4. Dynamic behavior Software Sensor Component

Trigger: core component sends location of jinput,

Software Sensor component sets its alarm level

6. User Interface Design

6.1. Overview of User Interface

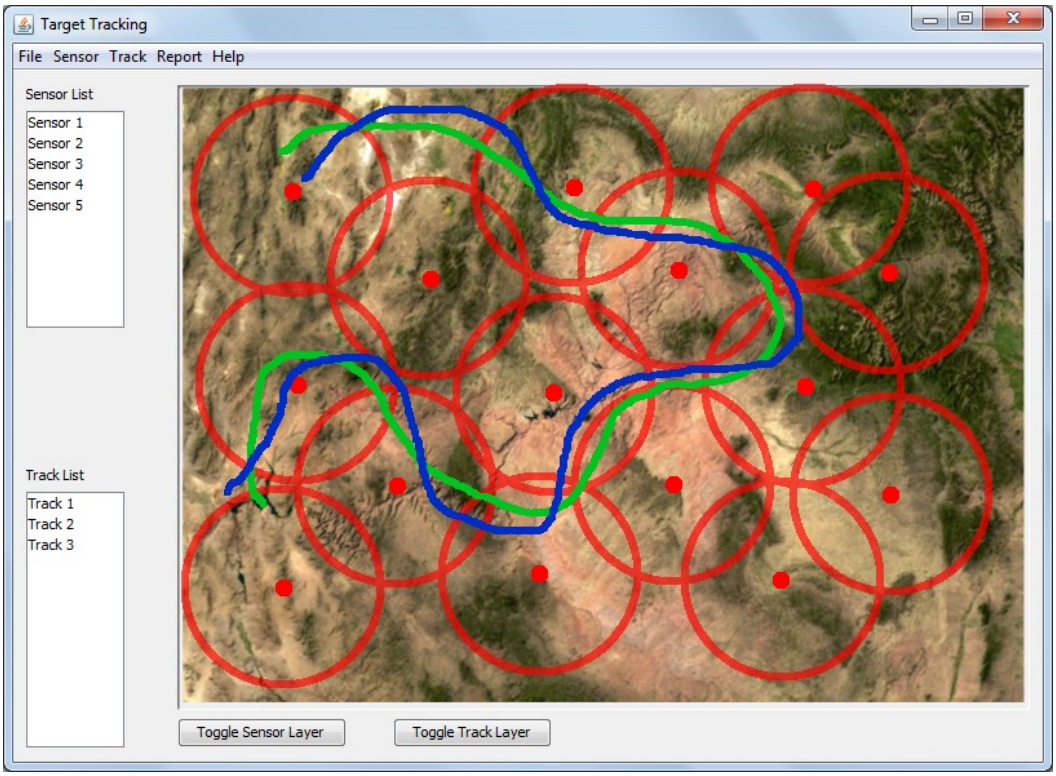

Figure 38: The GUI

The user interface contains the interactive menus, lists and map, which contains several layers.

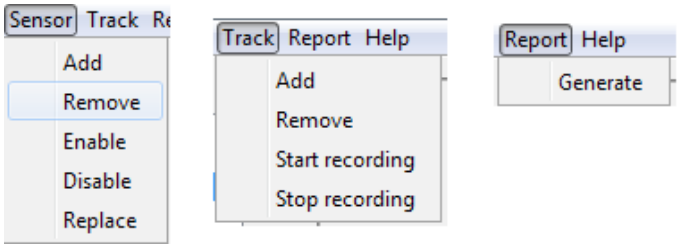

Figure 39: Menus

There are menus for sensor, track and report. The sensor menu allows the user to add a sensor, remove a sensor, enable a sensor, disable a sensor, replace a sensor. Similarly, the track menu allows the user to add a track, remove a track, start recording a track, stop recording a track. The report menu allows the user to generate a report based on the options provided to the user.

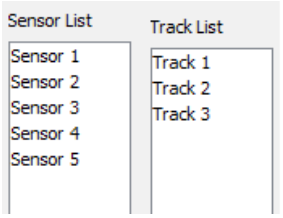

Figure 40: Lists

The track list allows the user to select a track and make it visible/invisible. Similarly, the sensor list allows the user to select a sensor and make it enabled/disabled for simulation.

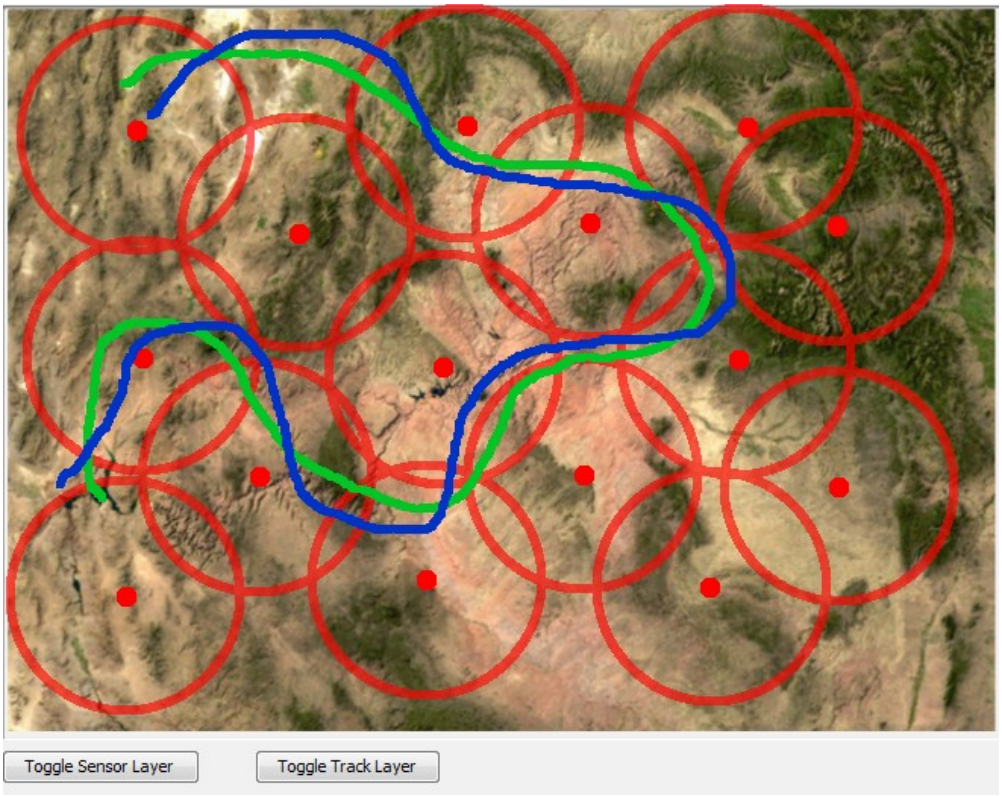

Figure 41: Map and toggle buttons

The map contains geographic map layer, sensor layer, track layer. It also has buttons for each layer, which is used for toggling the visibility of the layer. The geographic map layer shows the current chosen location data, as GPS photo data. The sensor layer shows the sensors placed on the map. This layer also shows when there is an alarm from the sensor. The track layer draws the currently generated track. If it is a simulation, it also draws the real track, given from the input.

The toggle buttons are used for toggling the visibility of the layers.

6.2. Screen Images

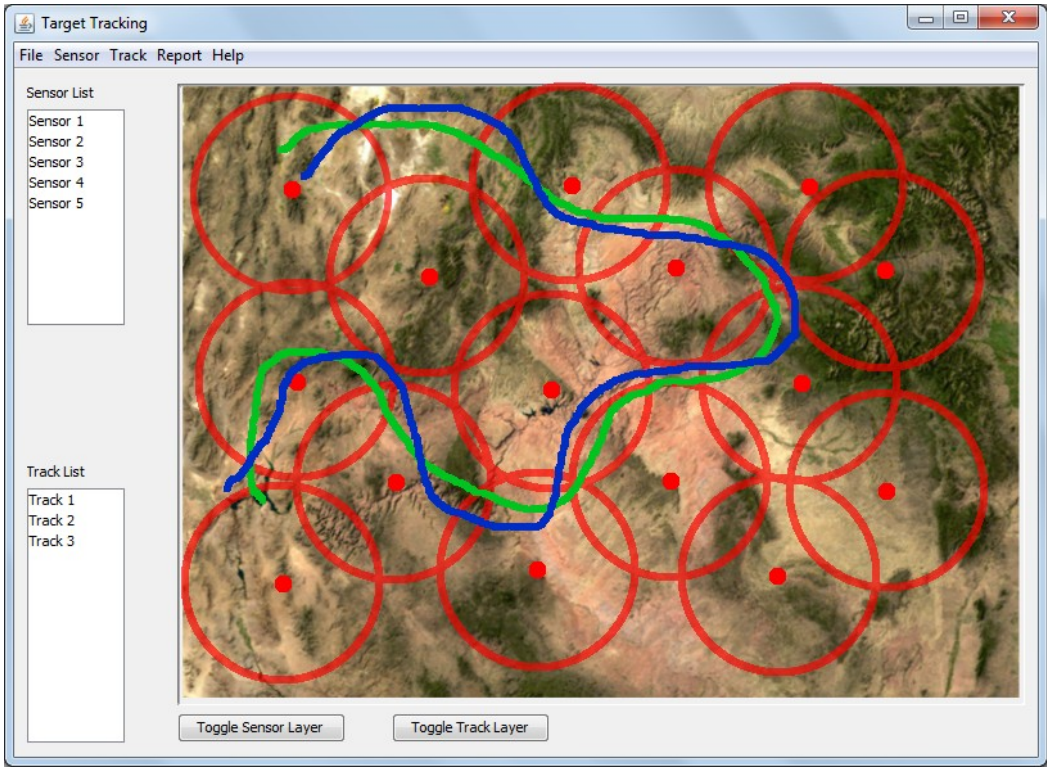

Figure 42: The GUI of the software.

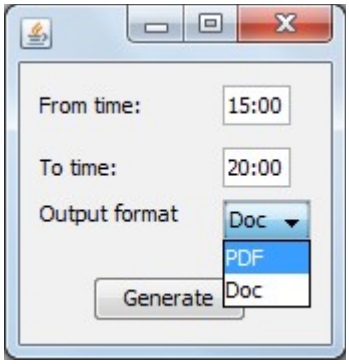

Figure 43: Report menu

6.3. Screen Objects and Actions

Menu Bar: The menu bar contains file, sensor, track, report and help menus. The file menu has the menu item “exit” and help menu has the menu items “help, about”. The sensor menu has menu items, “add, remove, enable, disable, replace”. The track menu has menu items, “add, remove”. The report menu has menu item “generate”.

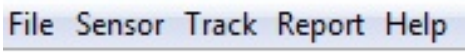

Figure 44: Menu Bar

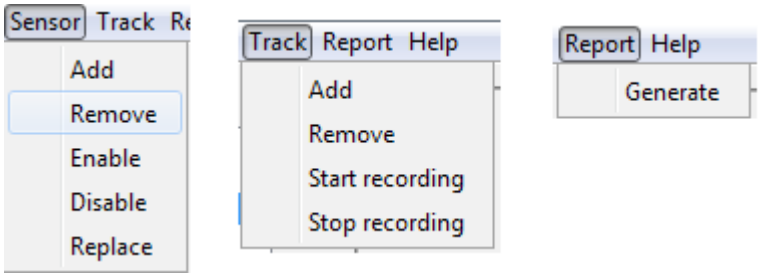

Figure 45: Menus

The sensor and track lists are interactive lists. The list items are selectable. The selection is used along with the related menu items. For example, user selects a sensor, then clicks “remove” from the sensor menu.

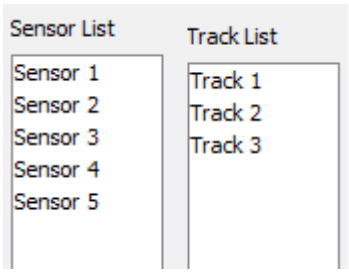

Figure 46: Lists

“Toggle Sensor Layer” and “Toggle Track Layer” are toggle buttons.

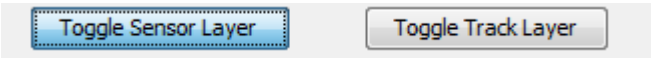

Figure 47: Toggle button. (Left one is on, right one is off)

7. Libraries and Tools

In this project, the sponsor company, ASELSAN, has provided us Waspmote Seismic Sensor Card, Waspmote Gateway as our hardware devices. We wil implement our 'target tracking by using seismic sensors system' using Java Eclipse RCP. We are also provided some libraries like lowagie-itext, sun-pdfview, jinput-windows libraries. We will use PostgreSQL as our database managment system. We will use PostgreSQL because it is an object-relational database managment system (ORDBMS) available for many platforms including Linux, FreeBSD, Solaris, MS Windows and Mac OS X.

8. Time Planning

The Gannt chart for both terms is below.

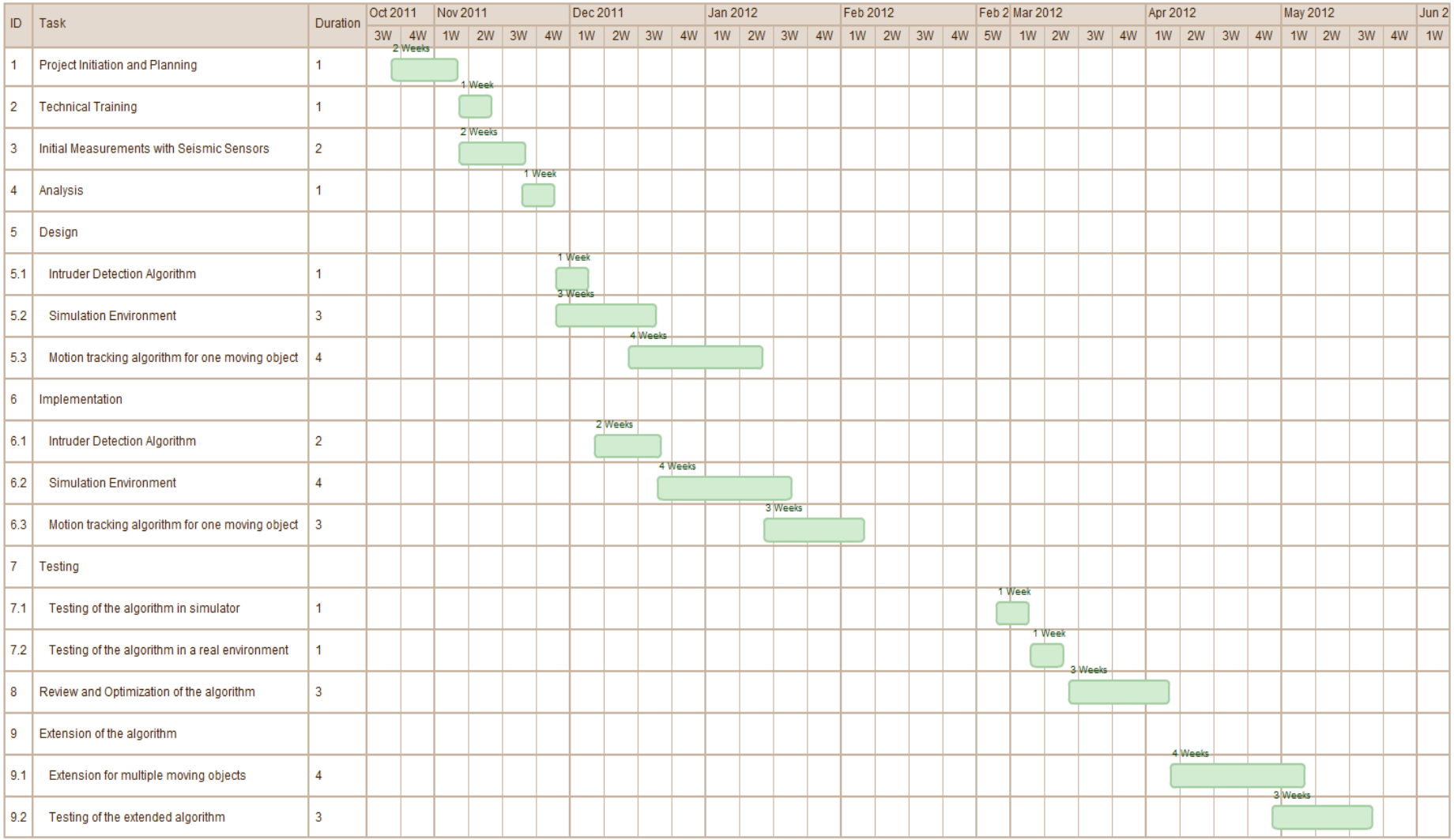

Figure 48: Gannt chart

9. Conclusion

In this document, we have talked about the initial design specifications of our project. We have discussed design considerations, data design, system architecture, libraries and tools.

After the completion of this project, the end product will be used by Turkish Company ASELSAN. Possible scenarios for the usage of this product are border control, battle field surveillance, traffic flow measuring, or animal monitoring, etc.