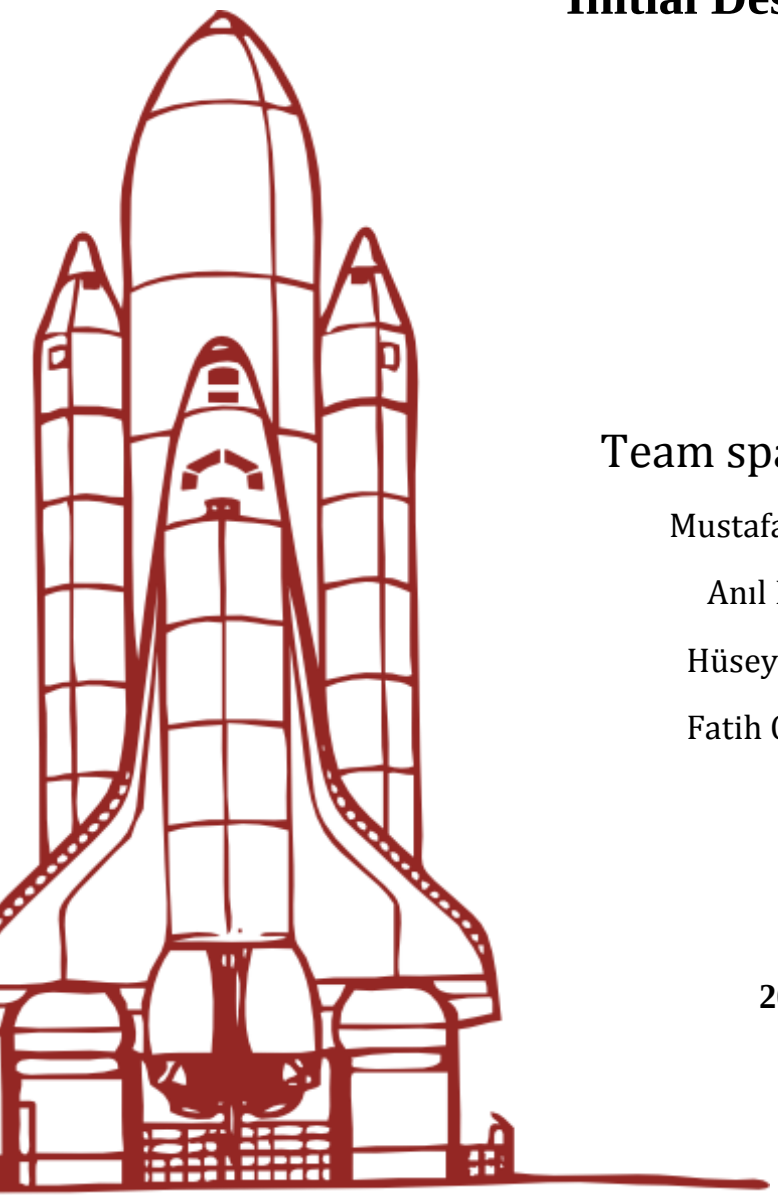
Middle East Technical University

Department of Computer Engineering

Target Tracking by Using Seismic Sensors

Initial Design Report



Team spaceShuttle

Mustafa MIZRAK

Anıl ERKOÇ

Hüseyin ÜNAL

Fatih GÜNDÜZ

2011

TABLE OF CONTENTS

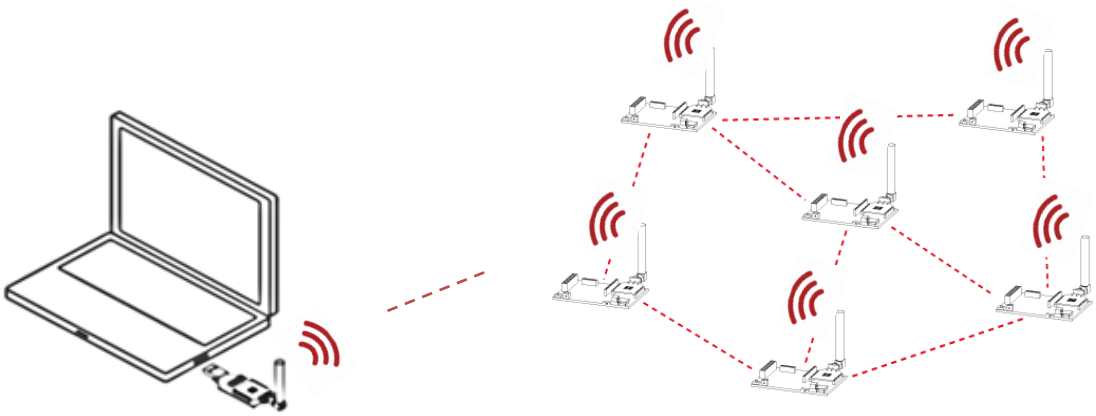
1 INTRODUCTION

This Inıtial Software Design Report document describes the conceptual design of "Target Tracking by Using Seismic Sensors" project, senior design project of *spaceShuttle* team at Department of Computer Engineering, Middle East Technical University and sponsored by ASELSAN Inc. according to the functionalities identified in the Software Requirements Specification (SRS) document and also guidelines presented in the IEEE-1016 1998 Recommended Practice for Software Design Descriptions (SDD).

1.1 PROBLEM DEFINITION

Seismic sensors detect vibrations occurred on the ground by determining pressure changing. They are very sensitive and can detect vibrations caused by the movement of animals or human beings. People or objects moving over ground generate a succession of impacts and these soil disturbances propagate away from the source as seismic waves. Hence, they can be used to monitor protected areas to restrict entry of unwanted persons or animals.

The problem is real time object tracking by analyzing the waves of seismic sensors that the object generates while it moves over the ground. In this project, *spaceShuttle* team will develop a target tracking algorithm such that it monitors route of moving objects/targets within an area using large-scale-deployed and spanned-hundreds-of-kilometers seismic sensors provided by Aselsan Inc. Moreover, team will implement target tracking simulator based on a Geographic Information System (GIS) to monitor deployed seismic sensors and route of the moving objects (real object and from others).



1.2 PURPOSE

This document is intended to specify initial design details of Target Tracking by Using Seismic Sensors project that implements the track of a moving target within an area where seismic sensors are previously deployed with satisfying the dependencies, functional and non-functional requirements, interfaces, design constraints, data model and descriptions and usage of the program specified in the Software Requirements Specification (SRS) document.

The main purpose is to give almost all information about the project to developers to code. It is prepared to cover all requirements and necessary steps including design considerations, data design, system architecture and user interface before detailed design of the project in order to not redesign, recode, and retest the project. Furthermore, this document aims to ensure misunderstandings and inconsistencies at the first stage/phase of the development design process of the project are minimized or eliminated. Also, this document provides a guide to potential developers who want to work on Target Tracking how the software requirements should be implemented.

1.3 SCOPE

This document shows how the project will be structured to satisfy the requirements identified in the Software Requirements Specification (SRS). This document covers architectural design, data design, procedural design, design constraints, development schedule, and hardware and software requirements of the project.

1.4 OVERVIEW

This Initial Design Document composes of nine parts. In the first part, named introduction part, the problem be intended to solve is defined, and the purpose of this document is explained. Furthermore, the scope of the project is explained, the contents of the document are summarized, and definitions, acronyms and abbreviations relevant to the project are mentioned.

In the second part, named system overview, a general description of the software system containing its functionality and matters related to the overall system and its design are explained in details. The intended goals, objectives and benefits of the project is briefly explained.

In the third part, named design considerations, special design issues which need to be addressed or resolved before attempting to devise a complete design solution are noted. This part is divided into two sub-parts: The first one is about design assumptions, dependencies and constraints. The second one is about design goals, guidelines and principles. Two sub-parts are detailed in this part.

In the fourth part, named data design, information about data design (data description) and data entities in the system is provided.

In the fifth part, the system architecture issues are explained in containing architectural design, description of the components, design rationale and traceability of requirements subsections. In architectural design, the subsystems of the project are represented, and the structure of the project is defined and also relationships between the sub modules to achieve the complete functionality of the system are expressed. UML diagrams are provided. Then, description of each component, input and outputs of the each component, interface of the components and algorithmic description of each component are detailed. In the design rationale subsection, the way of decomposing the system that our project group has chosen is explained. In the last subsection, use case of each component of the project is given.

In the sixth part, the outline user interfaces of the project are given. How user can use the system is explained with outlined screenshots. These screenshots contain all expected features.

In the seventh part, internal details of each component including attribute descriptions for identification, processing and data are mentioned. This part contains all the details that will be needed by the programmers for implementation. Data structure and algorithm details are also given.

In the eighth part, libraries, APIs and tools that our project is dependent on are explained.

In the ninth part, time planning of our project group is given. The Gantt Charts of fall and spring term are given, separately.

The last part covers a brief conclusion of Initial Design Document.

1.5 DEFINITIONS, ACRONYMS and ABBREVATIONS

IDR: Initial Design Report

SDD: Software Design Description

SRS: Software Requirements Specification

SS: Seismic Sensor

GIS: Geographic Information System

DB: Database (PostgreSQL Database)

KF: Kalman Filter, it is to use measurements observed over time, containing noise and other inaccuracies, and produce values that tend to be closer to the true values of the measurements and their associated calculated values.

IKF: Intelligence Kalman Filter is derived for tracking the maneuvering target model where the time varying variance of process noise is computed in an intelligent manner using a fuzzy system.

UML: Unified Modeling Language, a standardized general-purpose modeling language in the field of software engineering

GUI: Graphical User Interface, a type of user interface that allows users to interact with programs with images rather than text commands

PostgreSQL: An object-relational database management system (ORDBMS) based on POSTGRES developed at the University of California at Berkeley Computer Science Department. PostgreSQL is an open-source descendant of original Berkeley code. It supports a large part of the SQL standard and offers: complex queries, foreign keys, triggers, views, transactional integritya and multiversion concurrency control features.

NASA World Wind: Open-source, high-performance 3D Virtual globe API and SDK, provides geographic visualization to any application, huge collection of high-resolution imagery and terrain from NASA servers, and displays high-resolution imagery, terrain and geographic information

from any open-standard public or private source, open-standard interfaces to GIS services and databases, large collection of geometric and geographic shapes.

1.6 REFERENCES

[1] IEEE Std 1016-1998: IEEE Recommended Practice for Software Design Descriptions

[2] Schach, Stephen R. Object-Oriented and Classical Software Engineering, Fifth Edition, McGraw-Hill, 2002

[3] B.J. Lee, J.B. Park, Y.H. Joo and S.H. Jin, Intelligent Kalman filter for tracking a manoeuvring target, IEE Proceedings, 2004

[4] M.I. Ribeiro, Kalman and Extended Kalman Filters: Concept, Derivation and Properties, Institute for Systems and Robotics Instituto Superior Tecnico, Portugal

[5] J. P. lielferty, Improved Tracking of Maneuvering Targets: The Use of Turn-Rate Distributions for Acceleration Modeling, IEEE Transactions on Aerospace And Electronic Systems Vol. 32, NO.4 October 1996

[6] D. Li, K. D. Wong, Y. H. Hu and A. M. Sayeed, Detection, Classification and Tracking of Targets in Distributed Sensor Networks, Department of Electrical and Computer Engineering, University of Wisconsin-Madison, USA

2 SYSTEM OVERVIEW

Target tracking software is intended to detect intruder's path that is moving in a forbidden zone. Based on detection from the seismic sensors, the route of the target will be calculated and the system also will accept manually entered paths from users to check the performance of the tracking algorithm.

The system will consist modules and we will construct the system in a modular structure. Core two modules of the system will be target tracking algorithm and system simulator modules.

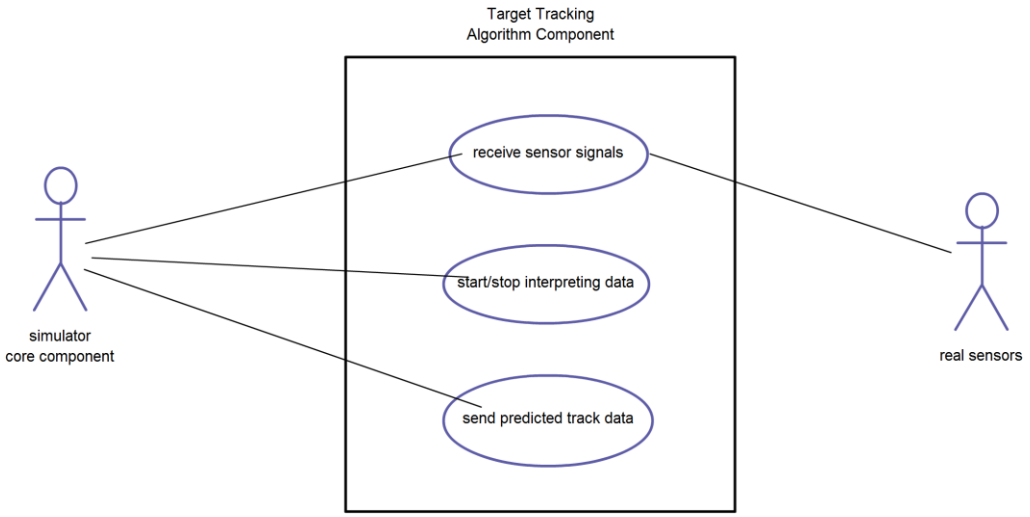
In more detail, target tracking algorithm component is to calculate the path of the intruder and a GIS based sensor system simulator will provide a GUI to display the track and predicted way points. The simulator will accept sensors' position values and target route manually. By using sensor inputs, the implemented target tracking algorithm will estimate the way points. So the system simulator will show both the calculated way point and manually entered track to evaluate the correctness of the algorithm.

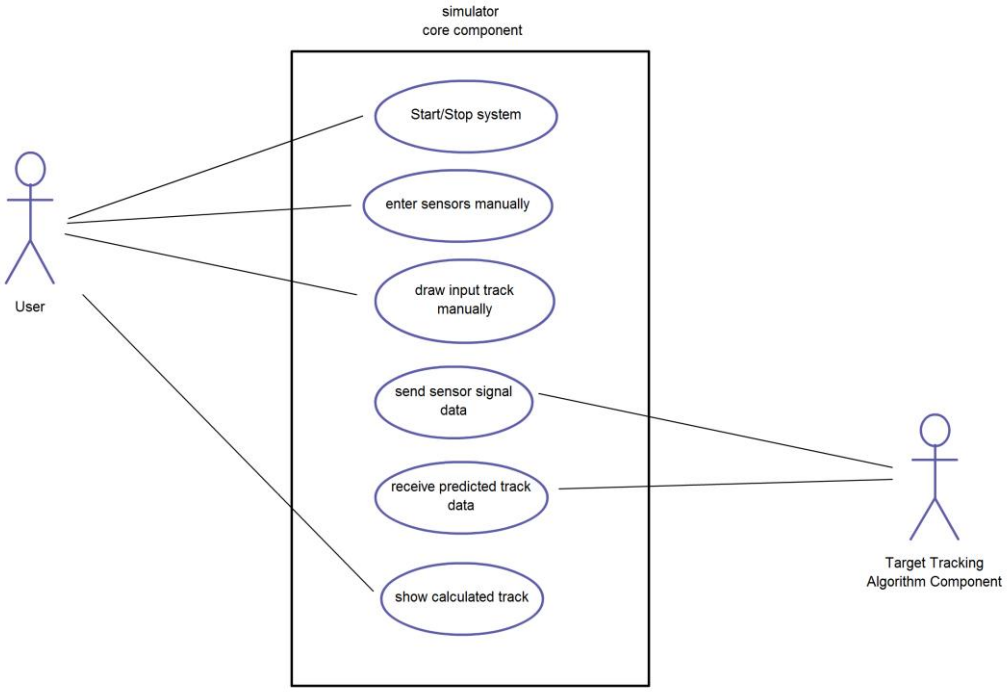
Users will be able to manually draw tracks by joysticks. Joystick component will get the input and direct it to track drawing related components.

Also the system will be integrated with the real sensor network. Real sensor will sent data via wireless communication. Communication component will get the data stream coming from sensors and after converting it to meaningful data, direct it to simulation component.

Simulation environment will be showing the calculated track based on coming information from real sensor through wireless network.

Finally, the users will be able to print and obtain hard copies of the information displayed. Report component will handle this task. The component will convert the information to PDF file format and the users will be able to print it.





Those components will be described in chapter 5 in details.

3 DESIGN CONSTRAINTS

In this section, assumptions, dependencies and constraints in the design of our project will be explained. Design goals and guidelines also will be explained.

3.1 DESIGN ASSUMPTIONS, DEPENDENCIES, AND CONSTRAINTS

3.1.1 DESIGN ASSUMPTIONS AND DEPENDENCIES

While designing this project, we had to make some assumptions about software working environment, hardware and end user.

We do not have any assumptions about operating system which runs our project. It has to be OS independent. Our project has to work on Linux OS or Windows OS. When user wants to change OS, user should not change anything about project.

We do not have any assumptions about environment. Because, this application works on various type of geographical formation. Because of it is unknown where terrorists can come into the country; our project has to work on highland or lowland.

There is assumption about end user. End user has to be informed about how this project can be used and where seismic sensors should be located.

3.1.2 DESIGN CONSTRAINTS

3.1.2.1 Time Constraints

This project started on June 2011, as senior design project. Our aim is to accomplish this project prototype end of this semester. We want to finish our project end of academic year without any bugs. For this purpose, project tasks are divided into parts for each member of our team. In Gantt chart section, which is section 8, we made detailed project schedule. Our team will try to not exceed these time constraints.

3.1.2.2 Performance Constraints

Our project works in real time, because it tracks real targets which run or walk on real geographic areas. For this reason, performance is so important. It has to handle route of targets in so efficient and fast way. We need to implement fastest and accurate target tracking algorithm. In the beginning we planned to use Kalman filter for performance. After first implementation we are planning to develop own algorithm in order to gain more performance.

3.1.2.3 Experience of Team Members

Each member of our team has different knowledge about parts of our project. However, we regarded these abilities little bit. Ali Fatih GÜNDÜZ and Hüseyin ÜNAL will be more related to user interface part, Anıl ERKOÇ and Mustafa MIZRAK will be more related to database part of our project. Implementing Kalman filter is product of all members of our team. These tasks distribution will not be strict. Each member is not broken off other parts of project. We want to this project be our joint product.

3.1.2.4 Financial Constraints

In order to accomplish, our project we will need to seismic sensors and APIs. Because of these simulations will be made on user's computer, after implementing this project, we do not need any simulator devices. Our project partner, Aselsan, supplied necessary seismic sensors and APIs. There is no other financial constraint. Other works related to our project does not need any financial support.

3.2 DESIGN GOALS

In this project, our main goals are accuracy and performance. Track of targets has to be drawn in accurate and fast way. In order to achieve best performance and accuracy, firstly we will implement Kalman filter then each member of our team offer own algorithm approach to this problem. Besides of performance and correct track, graphical user interface of our project has to be as simple as possible because of user profile of our project. In addition these goals, we also take the following features into considerations.

3.2.1 PORTABILITY

This project is designed to be platform independent. So it is independent from which operating system will be runs this project. Windows OS, Linux OS or MAC OS will be run our project without any changes. However we will not handle mobile operating systems such as Android, Symbian or IOS.

3.2.2 EXTENSIBLITY

This project designed to be extensible. It means any updates and changes have to be done easily. Our code implementation will be done with regard of this feature.

3.2.3 RELIABILITY

This project designed to be reliable. Because this project is real project, Turkish army forces will use this project to track terrorists. For this reason, it will run without any bugs or problems. To accomplish our project without any bugs, our team wants to be so fast in order to generate test application. After making demo of our project, we will spend all efforts on bug finding and bug correction.

4 DATA DESIGN

This section will give information about data objects, class attributes and data relations.

4.1 DATA DESCRIPTION

In our target tracking project, there will be two modes. One of them is target tracking by using real seismic sensors the other one is simulating by joystick part. Data collection type is different in each model. In the first mode, data is collected from seismic sensors. In the second model data is collected from joystick output. After gaining data from seismic sensor or joystick, these data is collected at database. In the database these data are stored with record times, strength level, and other relevant information. After storing data in database, basically Kalman filter calculates the instant path at that time and this particular track stored at different data table in our database. These data go to simulator core component. According to mode of our program, either program use real seismic sensor or program simulates sensor vector.

4.2 DATA DICTIONARY

4.2.1 SENSOR COMPONENT CLASS

isAlive: Type of this attribute is Boolean. This data holds information whether sensor is open or close. If sensor is open (gives signal) then this Boolean is true. If sensor is not available (does not give signal) then this Boolean is false.

sensorId: Type of this attribute is integer. This attribute holds information about sensor identification number. sensorId is settled by user or by default. Auto increment number is used to set sensorId.

signalStrength: Type of this attribute is integer. This holds signal level of sensor. Domain of data for this attribute is from zero to a hundred. This attribute is settled to zero if sensor is not alive.

sensorPosition: Type of this component is GISComponent. This attribute holds position of sensor in type of GISComponent.

4.2.2 GIS COMPONENT CLASS

locateToMap(int x,int y): This function locates any drawing into x,y coordinate. This method gets coordinate of point. There is no return value for this method

locateLineOnMap(int x1,int y1,int x2, int y2): This function locates line which is given with respect to two points. This line is located on to map by this function. There is no return value.

deleteFromMap(int x,int y): This function deletes a point from map. It has no return value. X and y coordinate is parameters for this function.

2DPosition: Type of this attribute is point. This point contains two double values; x position and y position.

4.2.3 JOYSTICK COMPONENT CLASS

pathString: Type of this attribute is string. To get input from joy stick, this attribute is used.

4.2.4 REPORT COMPONENT CLASS

data: This attribute is in string type. After final data is composed, final result is settled to data string.

4.2.5 DATABASE COMPONENT CLASS

addSensor(int id,int x, int y): This function gets id,x,y parameters in integer type. This function adds sensor to database.

addNewSignalStrength(int x, int y): This function get x,y parameters. Assigns a new signal strength to given position.

addTrack(path:Track): This function gets parameter in Track type. And this track is written to database by this method.

updateTrack(path:Track): This function makes updating on real map. There is no return value.

sensorInfo: This is in table type. This means; there is data table in database which holds sensor is alive or not, sensorId, signal strength of sensor. So all information about sensors is hold in this table.

dataTable: This is in table type. There is a table which holds sensor identification numbers and corresponding to time at which sensor is get signal.

trackInfo: This is in table type. Old sensor data, current sensor data and predicted sensor data are stored in this data table.

log: This is in data table type. User logs are stored in this data table.

4.2.6 TARGET TRACKING COMPONENT CLASS

sendToDataBase(path:Track): This method gets path in Track type as parameter. It sends this path to database.

calculatePath(sensorVector:vector<Sensor>): This method gets vector of sensors. By using Kalman filter, this method calculates path.

4.2.7 SIMULATOR COMPONENT CLASS

vectorOfSensors: this holds vector of sensorComponent so it is in type <vector: sensorComponent>.

4.2.8 STREAM CONVERTER COMPONENT CLASS

Data: this attribute is in <vector: byte> type. Acquired data is stored in this vector as byte vector.

4.2.9 COMMUNICATION COMPONENT CLASS

convertedData: This is in <vector: int> type. After manipulating data and converting data from byte vector to integer vector, this vector holds this converted vector.

4.2.10 TRACK COMPONENT CLASS

Track: Type of this attribute is lineString. This attribute holds tracks.

4.2.11 GIS COMPONENT CLASS

reporter: This attribute is in reporter type. This attribute is used in order to print out tracks in pdf format.

parse(theOutputTrack:Track): This function get outputTrack parameter. Parse this track

5 SYSTEM ARCHITECTURE

A general description of system architecture of the project is analyzed in this part of the document.

5.1 ARCHITECTURAL DESIGN

The architectural design of the project consists of 12 components namely. They are connected to each other in terms of data flow. Simulator Core Component, the most important part of the design, is the responsible showing tracks, positions of the sensors, information of the sensors and additional actions including adding new sensors, deleting the existing sensors and disabling the selected sensors and user interface. Data comes from Database component, Target Tracking component, and Communication component to this component. This component gets data, processes those data and sends the results to the Database component. Database component gets data (data comes from seismic sensors) from Communication component, and data (track information) from Target Tracking component. Joystick Component gets data via Joystick and send Track Drawing component That component draw the track and send it to the GIS component. Target Tracking component gets data from Database Component and Track component. That component is responsible for algorithmic of the track using Kalman Filter and other algorithms. Determined tracks in this components flows Track Creator component and Simulator Core component. Track Creator component is responsible for determining the tracks. The results of this component are sent the Report Component to report. Communication component is responsible for getting signals/information form seismic sensors and it sends this information to the Simulation Core component. Sensor Component keeps all sensor information including the positions, aliveness of the sensors.

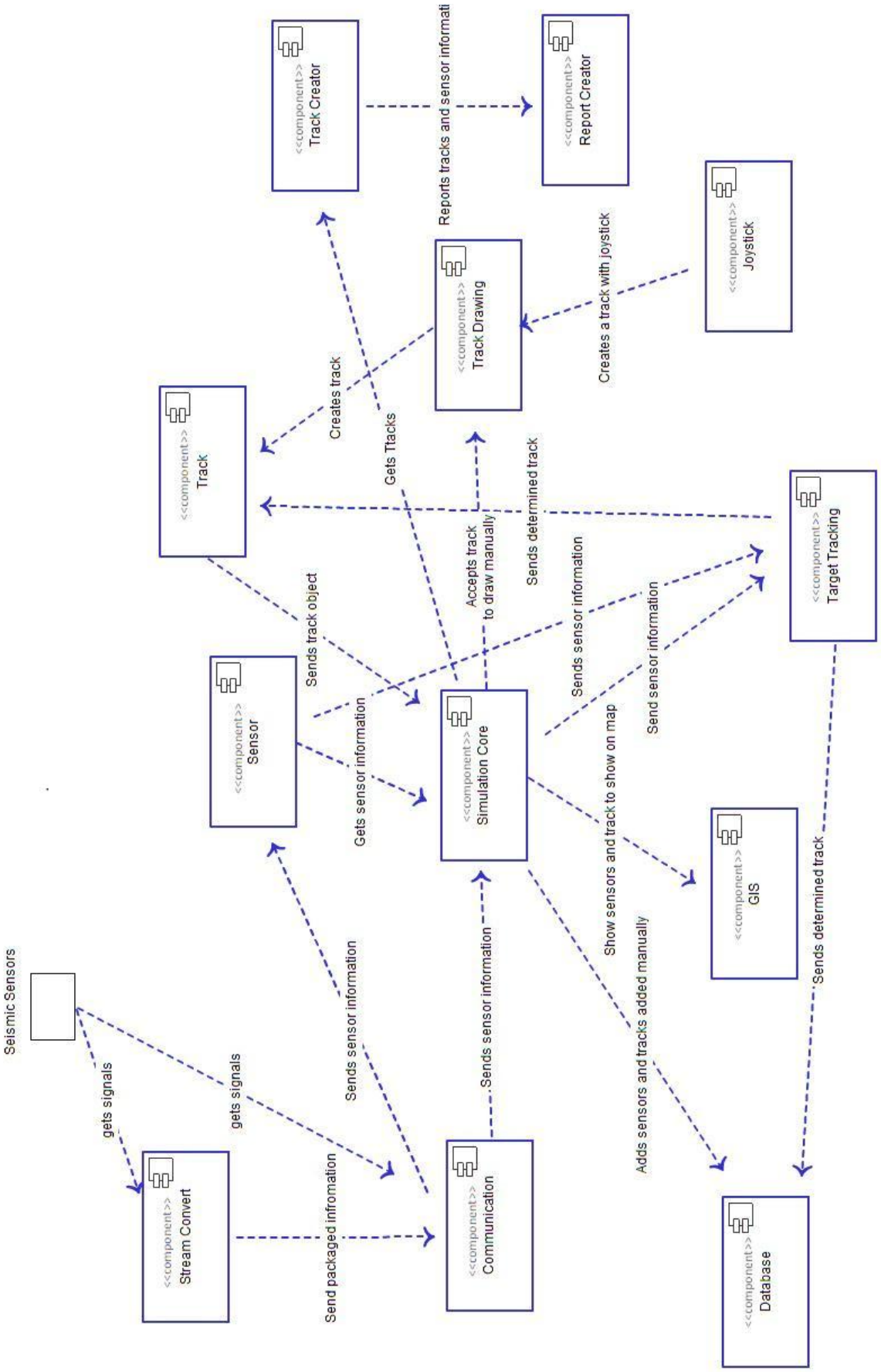


Figure. Component Diagram of the system

5.2 DESCRIPTION OF COMPONENTS

In this section decomposition of the subsystems in the architectural design is explained in all details.

5.2.1 GIS COMPONENT

5.2.1.1 Processing Narrative for GIS Component

The Geographic Information System (GIS) component is responsible for drawing routes and seismic sensor positions on the real map. When the information about sensorInfo, trackInfo, or data collected previously from seismic sensor is needed, the Database component is accessed via Simulator Core component and the desired information is gotten from there. Then, GIS component draw the required route and label the position of the sensors on the map. In this component, we will use the NASA World Wind open source, real map API.

5.2.1.2 GIS Component Interface Description

GIS component gets required information via Simulator Core component as an input, so Simulator Core component is an input interface. There is no an output interface of GIS component but the results (draws on the map) is shown in the map interface.

5.2.1.3 GIS Component Processing Detail

- SimulatorCore component sends the data to be shown on the map
- GIS compoent gets this as an input
- GIS component draw those on the map via NASA World Wind API

5.2.1.4 Dynamic Behavior of GIS Component

GIS component has only interaction Simulator Core component. Simulator Core component gets required data from Database component, and then those data is send GIS component to be drawn. GIS component draws and result is shown on the real 3D world map.

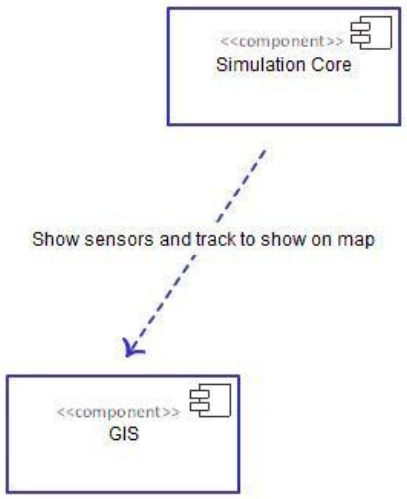


Figure. GIS component

5.2.2 DATABASE COMPONENT

5.2.2.1 Processing Narrative for Database Component

The Database component is responsible for accessing database. When the information about sensorInfo, trackInfo, or data collected previously from seismic sensor is needed, the Database component accesses the database, gets the desired information from database and sends it to target Tracking or Simulator Core component, in order for it to be processed. For Simulator Core component all tables in the database are needed. For Target Tracking component sensorInfo, and data are needed.

5.2.2.2 Database Component Interface Description

Database component gets data from Simulator Core component as an input, so Simulator Core component is an input interface. The output interface of Database component is Target Tracking component, information is sent to Target Tracking component to be determining the track.

5.2.2.3 Database Component Processing Detail

- SimulatorCore component sends the data (seisemic sensors information)
- Database compoent gets this as an input
- Database component keeps this data in the sensor table
- SimulatorCore component sends the data to the Target Tracking component
- Target Tracking component sends the calculated routes to the Database component

5.2.2.4 Dynamic Behavior of Database Component

Database component has interaction Target Tracking and Simulator Core components. When the Communication component gets data from seismic sensors, the data is shown in the Simulator Core component, same data is saved in the database as an input and when Target Tracking demands data, the Database component gets information and sends the results to Target Tracking component.

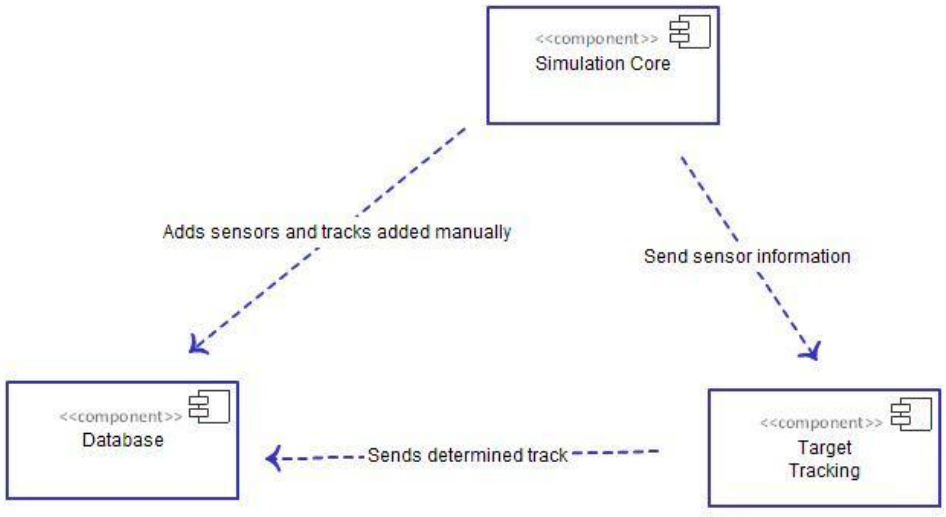


Figure. Database component

5.2.3 TARGET TRACKING COMPONENT

5.2.3.1 Processing Narrative for Target Tracking Component

The Target Tracking component is responsible for determining the track of the moving objects among seismic sensors. When the information about sensorInfo, trackInfo, or data collected previously from seismic sensor is needed, the Database component accesses the database, gets the desired information from database and sends it to Target Tracking. Target Tracking component determines the route of the object according to those sensor information using Kalman Filter algorithm in order to eliminate the noise to get more accurate results. The determined track is sent to Track Creator component to be report with Report component, and also it is saved in Database component in trackInfo table. Moreover, created track is send Simulator Core component to show track/route in the GUI.

5.2.3.2 Target Tracking Component Interface Description

Target Tracking component gets sensor information and data from Database component as an input, so Database component is an input interface. The output interface of Target Tracking component is Database, Simulator Core and Track Creator components. The calculated route of the object is determined with Target Tracking component, it is written with Track Creator component, and the route is saved Database component, and the same route is shown in the map via Simulator Core component.

5.2.3.3 Target Tracking Component Processing Detail

- Target Tracking component reads data from Database
- Target Tracking component uses the Kalman Filter algorithm and determines the route
- Determined route is created/drawn via TrackCreator compoent
- Determined route is saved Database, and sent SimulatorCore component

5.2.3.4 Dynamic Behavior of Target Tracking Component

Target Tracking component has interaction with Database, Track Creator and Simulator Core components. When data is gotten from Database, the data is proceed with Target Tracking component, result is sent to the database and Simulator Core component as an output and result is drawn with Track Creator component.

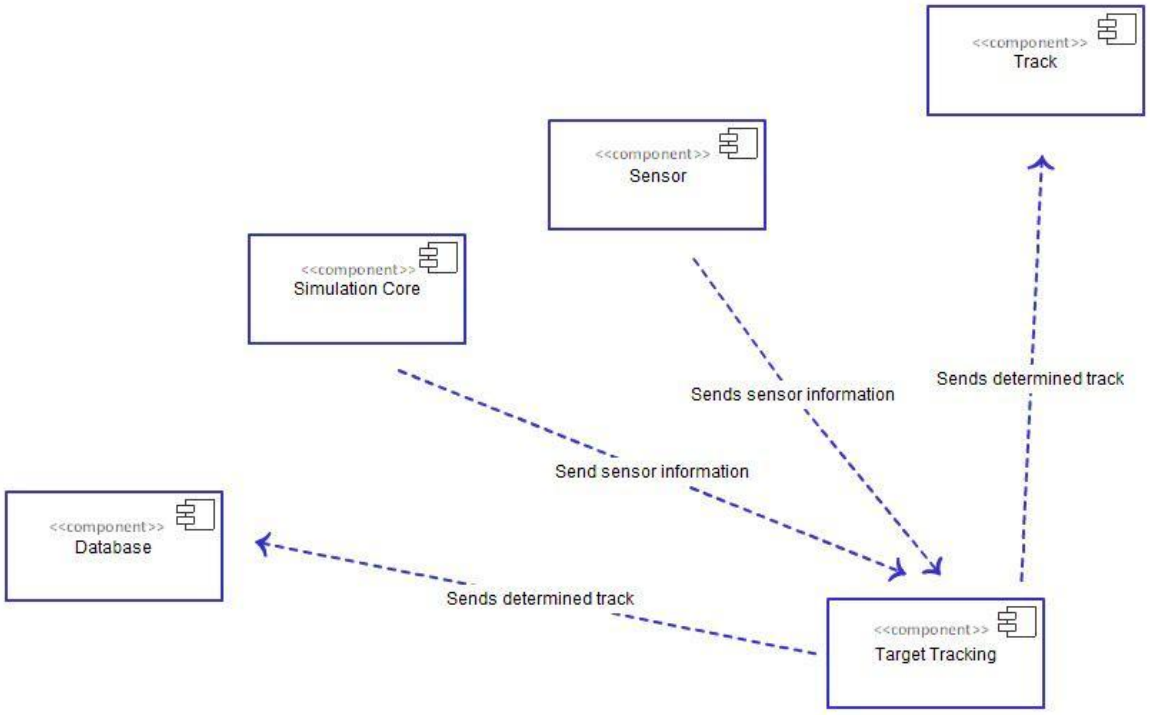


Figure. Target Tracking component

5.2.4 COMMUNICATION COMPONENT

5.2.4.1 Processing Narrative for Communication Component

The Database component is responsible for getting new data from seismic sensors. When new data from seismic sensor is needed, the Communication component gets data from seismic sensors and separates the input string and sends separated parts to Simulator Core component, in order for those to be shown on the map. Communication component gets sensorID, date, sensorPosition, and signalStrength from seismic sensors as a string.

5.2.4.2 Communication Component Interface Description

Communication component gets demand from seismic sensors as an input, so sensor hardware component is an input interface. The output interface of Communication component is Simulator Core component, information is sent to Simulator Core component to be shown on a GUI.

5.2.4.3 Communication Component Processing Detail

- Seismic sensors gets signals from ground as a string containing sensorID, date, sensorPosition, and signalStrength
- Communication component gets these signals as an input
- Communication component seperates those inputs
- Communication sends those inputs to SimulatorCore component

5.2.4.4 Dynamic Behavior of Communication Component

Communication component has interaction seismic sensors (sensor hardware) and Simulator Core components. When the Communication component gets data from seismic sensors demands data, the data is separated and the required part are send to the Simulator Core component to be shown on the map.

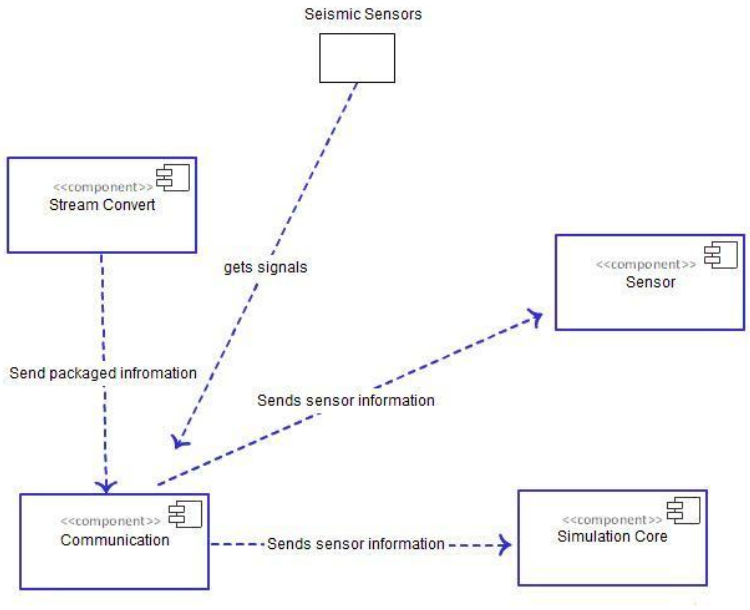


Figure. Communication component

5.2.5 SENSOR COMPONENT

5.2.5.1 Processing Narrative for Sensor Component

The Sensor component is responsible for getting sensor information from Communication component. When new data from seismic sensor is needed, the Communication component gets data from seismic sensors and separates the input string and sends separated parts to Sensor component. Then, from Sensor component, these sensor information is sent Simulator Core component in order for those to be shown on the map. Furthermore, they are also sent to Target Tracking component to determine the route of the objects.

5.2.5.2 Sensor Component Interface Description

Sensor component gets seismic sensors information as an input from Communication component so Communication component is an input interface. The output interfaces of Sensor component are Simulator Core and Target Tracking components. Required information is sent to Simulator Core component to be shown on a GUI, and also these information is sent the Target Tracking component to determine the route.

5.2.5.3 Sensor Component Processing Detail

- Seismic sensor signals are gotten via Communication component
- Communication component sent signal information the Sensor component
- Sensor component converts these signal to Sensor object.
- Sensor component sends these objects to
    - SimulationCore component
    - Target Tracking component

5.2.5.4 Dynamic Behavior of Sensor Component

Sensor component has interaction Communication, Simulator Core and Target Tracking components. When the Communication component gets data from seismic sensors data, the data is separated and the required part are send to the Simulator Core component to be shown on the map. Also, these data are sent to the Target Tracking component to calculate the routes of the objects.

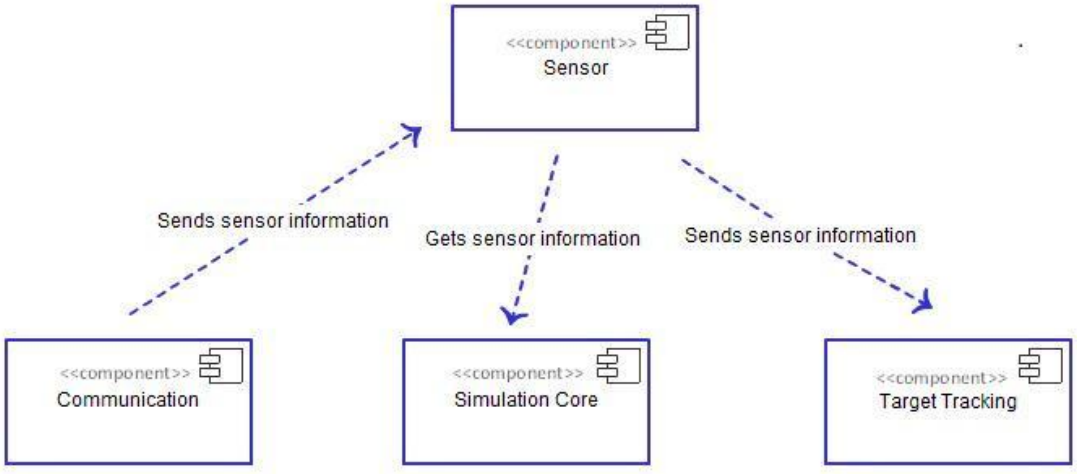


Figure. Sensor component

5.2.6 TRACK DRAWING COMPONENT

5.2.6.1 Processing Narrative for Track Drawing Component

Track Drawing component is responsible for drawing tracks manually by getting commands from Joystick component. When user needs to add tracks manually to the system, the system get the commands from the joystick, and according to these commands, Track Drawing component creates track object, then this track object is sent to Simulator Core component.

5.2.6.2 Track Drawing Component Interface Description

Track Drawing component has only one input interface which is Joystick component, and it has only one output interface which is Track component.

5.2.6.3 Track Drawing Component Processing Detail

- User uses joystick to add tracks manually to the system
- Track Drawing component uses commands from Joystick component to create a track object via Track component
- Track componentsend this track to the Simulator Core component

5.2.6.4 Dynamic Behavior of Track Drawing Component

Track Drawing component has interaction Joystick and Target components. When user draw track using the joystick the commands of the joystick are sent to the Track Drawing component, and then Track component determines and it sends the result to the Track component.

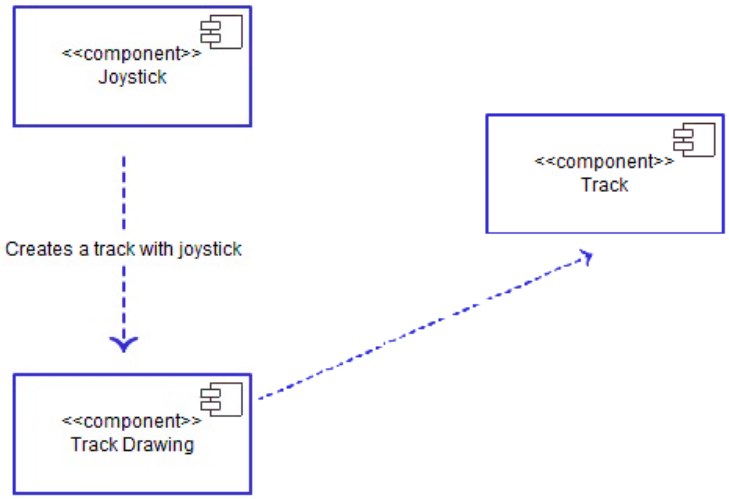


Figure. Track Drawing component

5.2.7 JOYSTICK COMPONENT

5.2.7.1 Processing Narrative for Joystick Component

Joystick component is responsible for joystick. When user needs to add tracks manually to the system, it uses the joystick and Joystick component creates commands according to the joystick usage. According to these commands, Track Drawing creates draws the track, then this track object is sent to Simulation Core component.

5.2.7.2 Joystick Component Interface Description

Joystick component has only one input interface which is joystick. It gets data form usage of the joystick, one output interface which is Target Drawing component.

5.2.7.3 Joystick Component Processing Detail

- User uses joystick to add tracks manually to the system
- Joystick component creates commands from joystick usage
- Resultant commands are sent to Target Drawing component.

5.2.7.4 Dynamic Behavior of Joystick Component

Track Drawing component has interaction joystick and Target Drawing component. When user draw track using the joystick the commands of the joystick are sent to the Track Drawing component, and then Track component determines and it sends the result to the Track component.

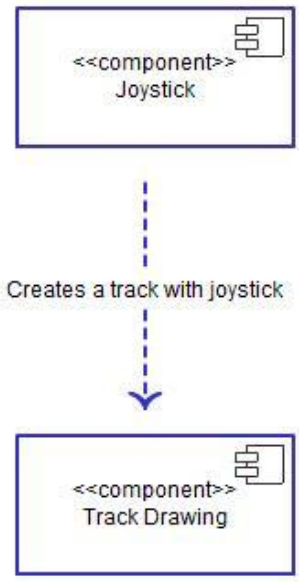


Figure. Joystick component

5.2.8 STREAM CONVERT COMPONENT

5.2.8.1 Processing Narrative for Stream Convert Component

The Stream Convert component is responsible for getting sensor information from seismic sensors to Communication component. When new data from seismic sensor is needed, the Stream Convert component gets data from seismic sensors and according to needed data, it created a package that contains all required information rather than keeping a string that keeps all required information. Then, from Stream Convert component, these sensor information is sent Communication component in order for those to be shown on the map. Furthermore, they are also sent to Target Tracking component to determine the route of the objects.

5.2.8.2 Stream Convert Component Interface Description

Stream Convert component gets seismic sensors information as a package. So, seismic sensors are input interface. Then, these packages are sent to Communication component, an output interface. Required information is sent from Communication component to Simulator Core component to be shown on a GUI, and also this information is sent the Target Tracking component to determine the route.

5.2.8.3 Stream Convert Component Processing Detail

- Signals are gotten from seismic sensors
- Stream Convert component process these signals and packages a class/struct that keeps all required information
- Packaged information/data is sent to the Communication component

5.2.8.4 Dynamic Behavior of Stream Convert Component

Stream Convert component has interaction seismic sensors and Communication component. When new signals are generated, the signals are sent to the Stream Convert component, and data is packaged into a class/struct, and then this struct is sent to the Communication component.

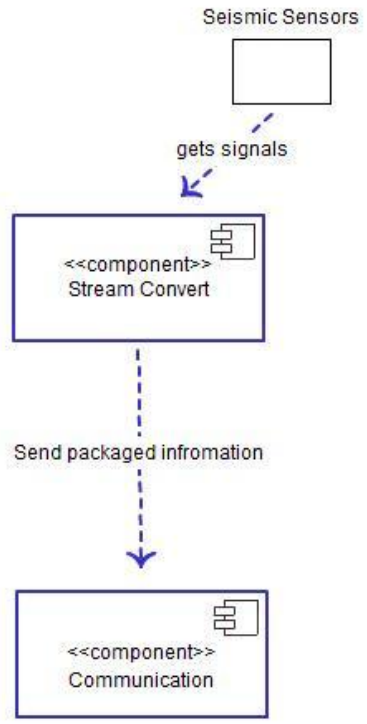


Figure. Stream Convert component

5.2.9 REPORT COMPONENT

5.2.9.1 Processing Narrative for Report Component

The Report component is responsible for reporting all or selected sensor positions, sensor aliveness, strength of the signals, and tracks/routes of the object for Command Control Unit. When user wants to report, the user must use the report button from the main interface of the system, and then a report will be created in PDF or DOC format containing all information up requested time according to choices.

5.2.9.2 Report Component Interface Description

Track Creator component is an input interface for Report component. There is no output interface of Report component. Sensor information and tracks are sent via Track Creator component to Report component.

5.2.9.3 Report Component Processing Detail

- User selects “Report” button in the main interface of the system
- Track Creator determines the tracks, and gets previous tracks and sensor information from Database component according to choices
- Result of the Track Creator are sent to Report component
- Report component saves information coming from Track Creator component in a PDF or Word format which is alredy defined

5.2.9.4 Dynamic Behavior of Report Component

Report component has only interaction with Track Creator component. When user select the report button from the main interface of the system, and then a report will be created in PDF or DOC format containing all information up requested time according to choices.

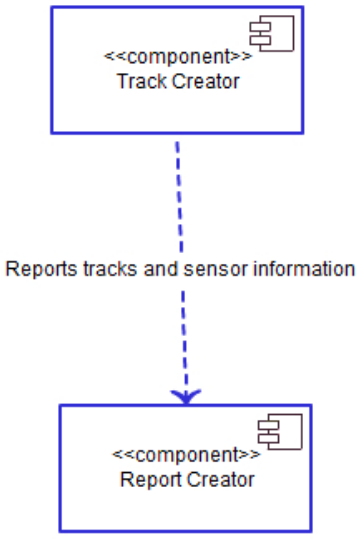


Figure. Report component

5.2.10 TRACK CREATOR COMPONENT

5.2.10.1 Processing Narrative for Track Creator Component

Track Creator component is responsible for detection of tracks or routes of the object according to signals coming from the seismic sensors. The signals are provided via Simulation Core component. In other words, signal coming from seismic sensors are sent to the Track Creator component to create the routes. In this component, we use our algorithm to detect the track. We will extend the Kalman Filter, to use measurements observed over time, containing noise and other inaccuracies, and produce values that tend to be closer to the true values of the measurements and their associated calculated value, in order to reduce noise margin. Then, the determined tracks are sent to the Report component in order to report the information up to requested time.

5.2.10.2 Track Creator Component Interface Description

Track Creator component gets seismic sensors information from Simulation Core component as an input interface. Then, determined tracks and sensor information are sent to Report component as an output interface.

5.2.10.3 Track Creator Component Processing Detail

- Track Creator determines the tracks, and gets previous tracks and sensor information from Database component according to choices via Simulation Core component
- Track Creator determintes the tracks
- Result of the Track Creator are sent to Report component to report

5.2.10.4 Dynamic Behavior of Track Creator Component

Track Creator component has interactions with Simulation Core and Report Creator components. Required sensor information is gotten from Database component via Simulation Core component. Tracks created in Track component are sent to Report component.

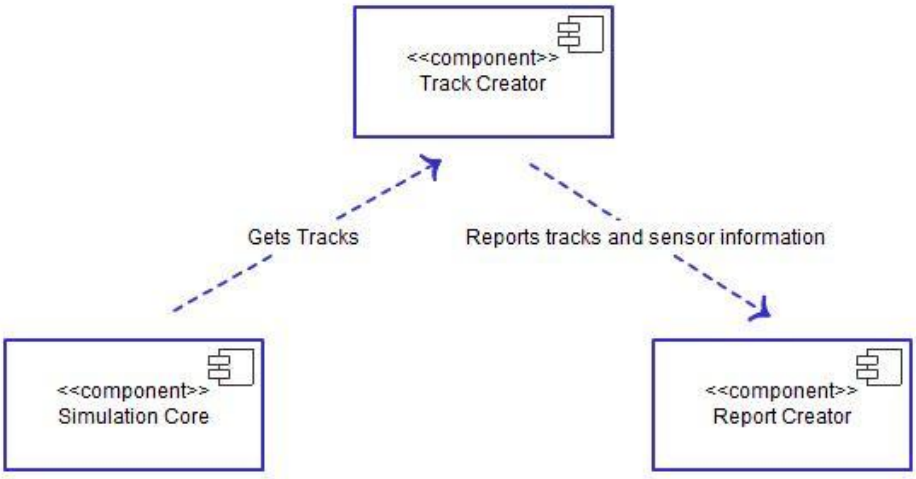


Figure. Track Creator component

5.2.11 TRACK COMPONENT

5.2.11.1 Processing Narrative for Track Component

Track component is responsible for creation of tracks or routes of the object. Track objects are created by result of Target Tracking component or Track Drawing component which uses the joystick. Then, track objects are sent to Simulation Core component to be shown on real 3D map.

5.2.11.2 Track Component Interface Description

Track component gets data from Target Tracking and Track Drawings components as input interfaces. Then, determined track objects are sent to Simulation Core component as an output interface.

5.2.11.3 Track Component Processing Detail

- Target Tracking component sends determined track or Track Drwaing component sends manually created tarck to Track component
- Track component creates track objects
- Created objects are sent to Simulation Core component

5.2.11.4 Dynamic Behavior of Track Component

Track component has interactions with Simulation Core, Target Tracking and Track Drawing components. Determined Tracks are sent from Target Tracking component to Track component, and created tracks from Track Drawing component to Track component. Track object are sent from Track component to Simulation Core component.

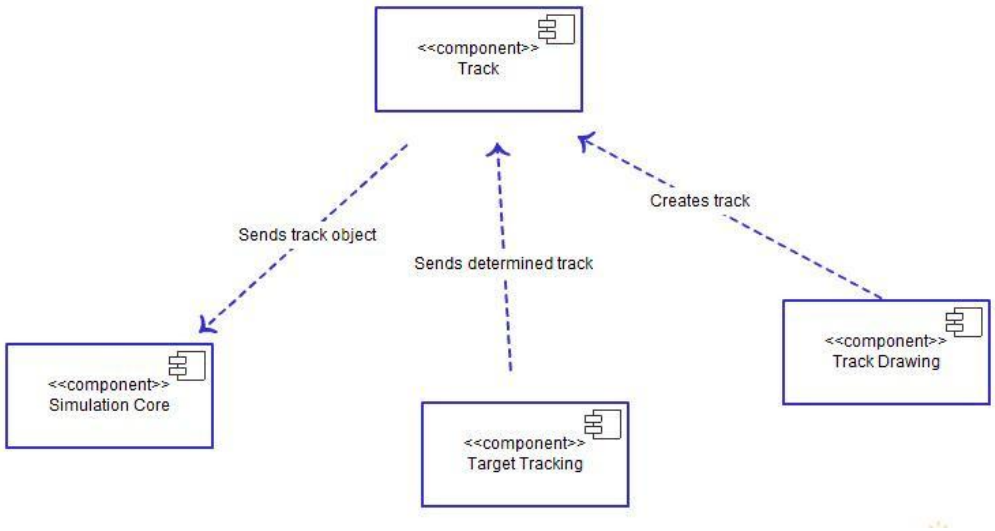


Figure. Track component

5.2.12 SIMULATOR CORE COMPONENT

5.2.12.1 Processing Narrative for Simulator Core Component

Simulation Core component is responsible for showing all information including routes and seismic sensor positions and aliveness on the real map. It is main component of the system. When the information about sensorInfo, trackInfo, or data collected previously from seismic sensor is needed, the Database component is accessed and the desired information is gotten from there. All tracks from either created manually or determined according to seismic sensor signals and signals from seismic sensors are shown in Simulation Core component.

5.2.12.2 Simulator Core Component Interface Description

Simulation Core component gets data from Communication, Sensor and Track components as input interfaces. Database, GIS, Target Tracking, Track Creator, and Target Drawing are output interfaces of Simulation Core component.

5.2.12.3 Simulator Core Component Processing Detail

- Communication component sends sensor information
- Sensor component sends sensor information
- Track component sends track object
- Database component saves sensors and tracks added manually from Simulation Core component
- GIS component shows sensors and tracks on the map
- Target Tracking component determines the tracks
- Track Creator component creates the tracks
- Track Drawing componet creates tracks manually (vith joystick)

5.2.12.4 Dynamic Behavior of Simulator Core Component

Track component has interactions with Communication, Sensor, Track, Target Tracking and Track Drawing, Target Creator, Database and GIS components.

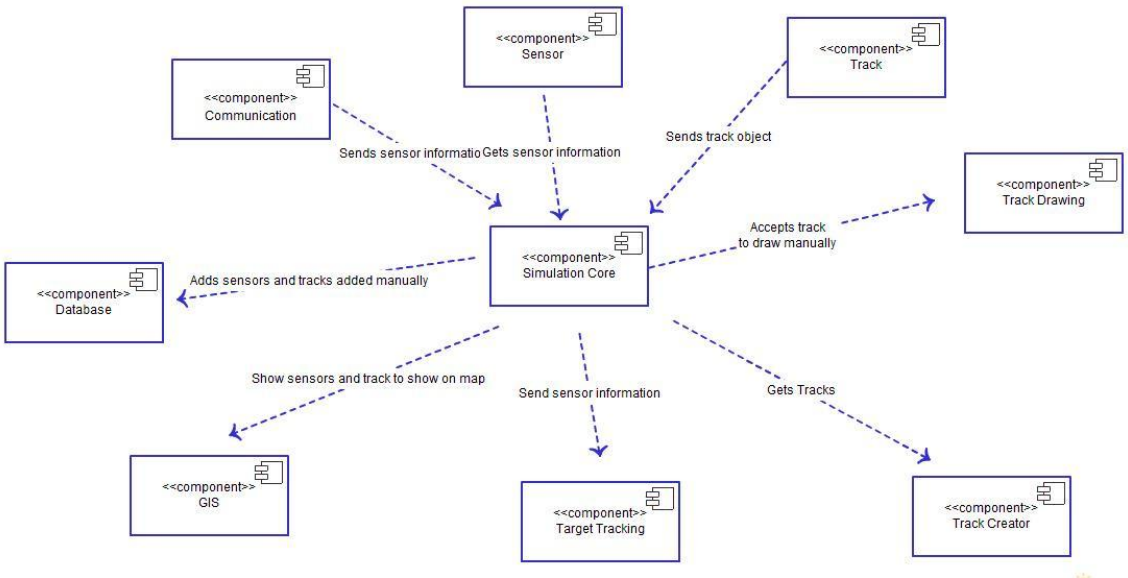


Figure. Simulator Core component

5.3 DESIGN RATIONALE

Our design is composed of 12 main components namely Track, Track Creator, Track Drawing, Sensor, Target Tracking, Report, Joystick, Simulation Core, Communication, Stream Convert, Database and GIS components. Main criterion used in decomposition of the system is to manage each subsystem easily; each subsystem has no common action with others. Furthermore, This type of design gives an opportunity to add additional component easily to the system after building whole system. With this design, we can easily add and remove or change behavior of components.

6 USER DESIGN INTERFACE

6.1 OVERVIEW OF USER INTERFACE

One of the main goals of our team is creating user friendly graphical user interface. Each member of our team offered his idea about GUI. Because of this tool is used by soldiers, GUI of our project will be able to so understandable. We are planning to accomplish GUI by using Java Swing library. Application mainly includes several Desktop forms which are provides to user to choose mode. Some GUI forms will make connection with database. In the beginning this database will be in our local hosts. After first implementation, our GUI will be able to have web-based interface plug-in for remote observation purpose.

Although we wanted to create understandable GUI, our user interface will not be funny because of we are trying to implement our project for Aselsan for a serious purpose. We keep project interface as simple as possible. Here is some snapshots of our project.

6.2 SCREEN SNAPHOTS

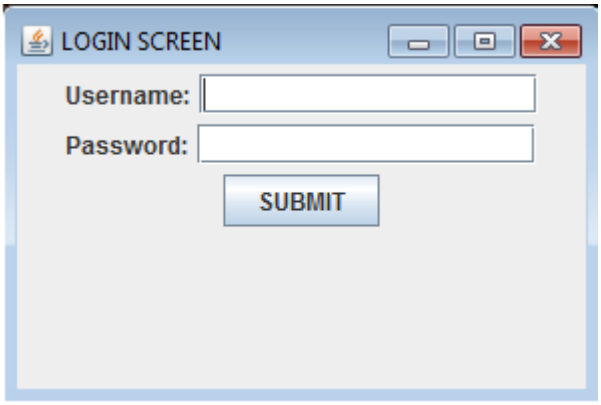


Figure. Snapshot of login screen

Figure. Selection mode panel

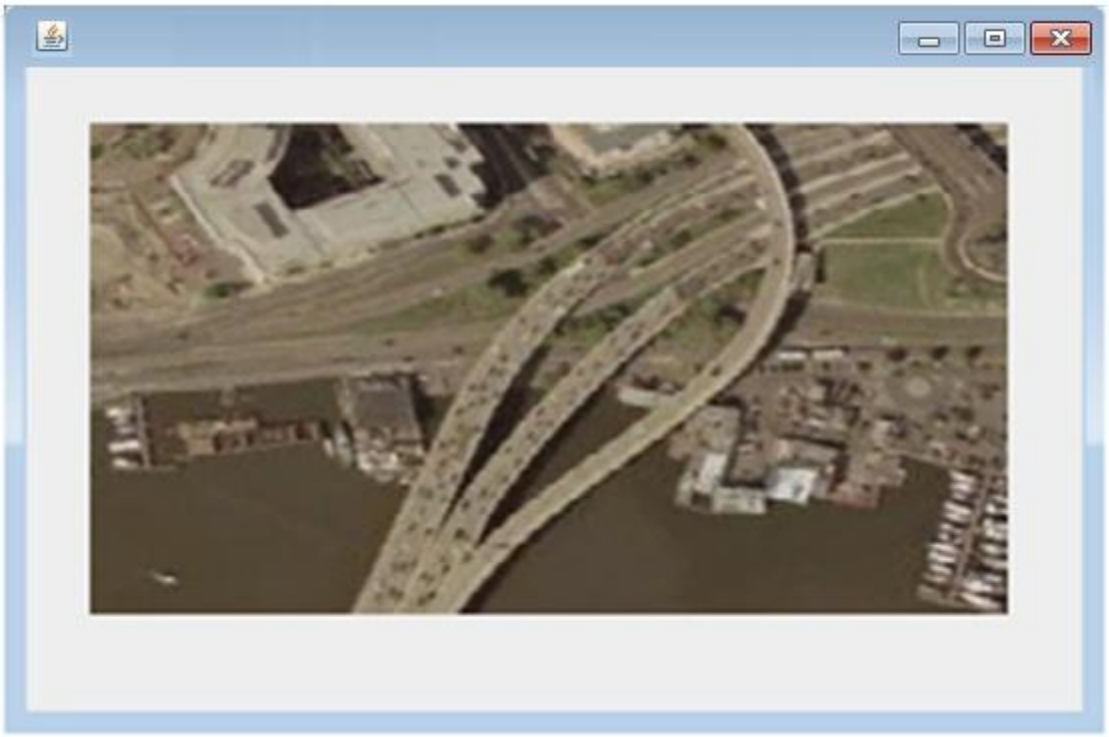


Figure. Image view brings map

6.3 SCREEN OBJECTS AND ACTIONS

In order to talk about user interface of our user design interface, it is know that in order to log into a system there should be user. This user should be initially authenticated in system. After opening our project, first screen is login panel. User will enter his password and user name then click the login button. If user is authenticated by system, he or she can enter our target tracking system. If his information denied he or she can't into the system. User and password information will be defined in the beginning. We can assign authentication level to user later.

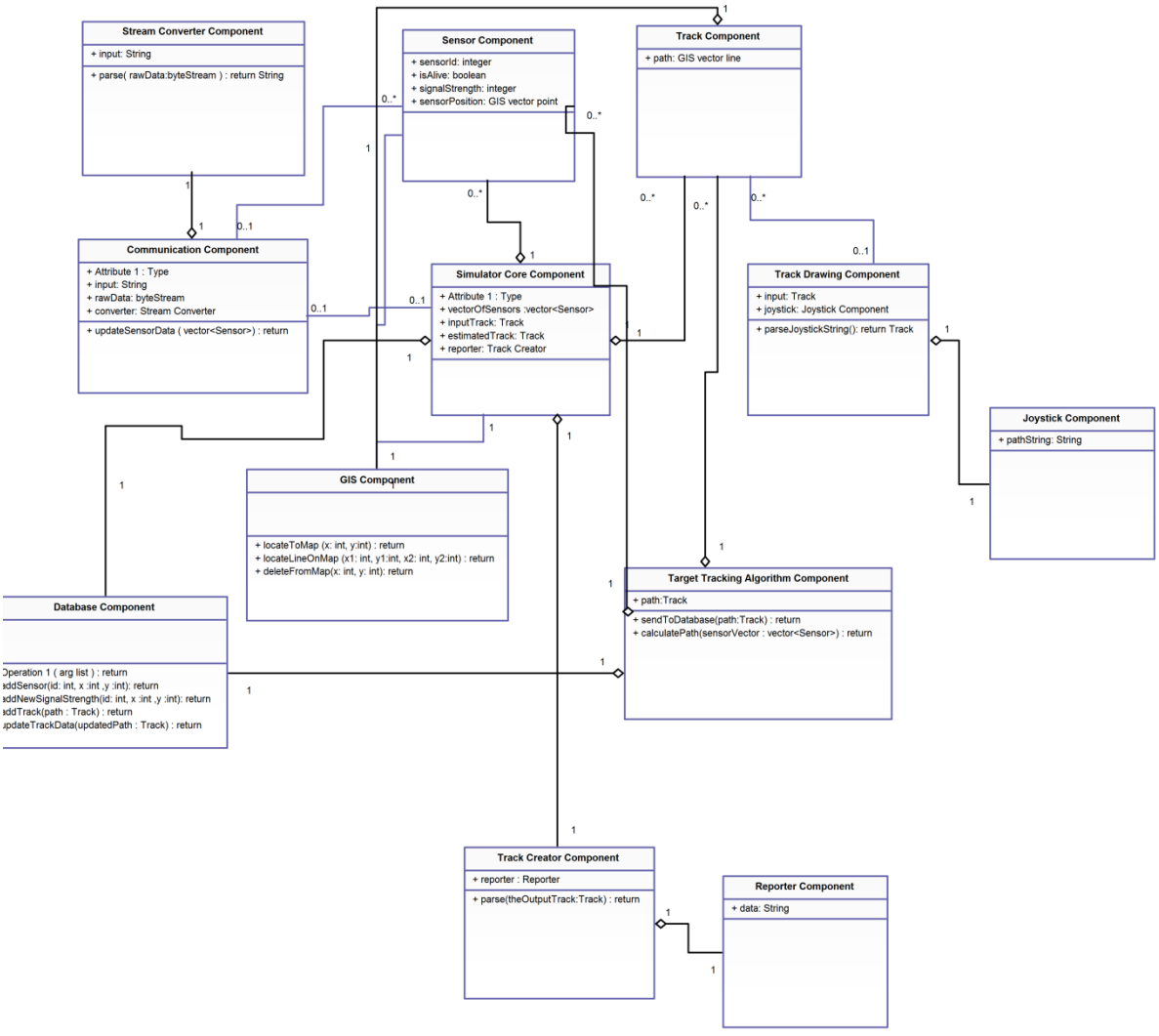
After entering system, there are two modes. One is tracking simulation mode, the other mode is real target tricking mode. User will chose one of two modes by clicking "Simulation by Joystick" and "Real target tracking" buttons. Each button will open a new window.

In first mode, application will simulate a virtual target tracking algorithm. That is first part of our project. User will chose seismic sensor icon and locate these sensors into map. Location seismic sensors will be dynamic. Drag and drop method can be used in this mode. After locating sensors, user will click "start target moving". This button activates joystick and user will move virtual target like a real target. Movement of the target which is detected by virtual seismic sensors will be drawn on the map.

In the second mode, application will draw real path for real target. Seismic sensors are used in this part. After soldiers or users locate these sensors at real area, user will click to “Observe” button. This button opens a new screen. Real map in which these seismic sensors are located will be shown in this window. If there is activation in this area, application will start to draw tracks of real targets. Tracking will be real time and as fast as possible. At anytime while application tracks target user will be able to clear map by clicking “clear map” button.

After each mode, user will be able to save these tracks to database. For this purpose, there is another button; save track. This button saves track, track record date and track record area. If user wants to print out tracks, our project will print out recorded tracks in pdf format. In order to get pdf of tracks, only thing user should do is clicking “print” button.

7 DETAILED DESIGN



7.1 STREAM CONVERTER COMPONENT

Classification:

Stream Converter Component is a module of the system.

Definition:

Name of this module comes from its duty. This module converts the byte stream which is coming from hardware sensors to meaningful integers.

Responsibilities:

This module's responsibility is to be a translator between hardware sensors and Communication Component. Sensors send the data signals as byte streams. Among those signals, there exist being alive of the sensor and seismic wave strength of a possible intruder. The seismic wave strength of the intruder is an integer in range [0, 1024]. For example if a sensor sending 0 strength seismic wave signals, this means this sensor does not detect an intrusion.

In brief, this component receives the data coming from wireless network over gateway and then parses it to send to Communication Component.

Constraints:

This module needs sensors and gateway. In order to be able work properly, the sensors should be sending signals in suitable frequency. This is from the real time working part of the system and if this module cannot reach the speed of the sensors then the signals coming from the sensors are lost.

Composition:

None, this is at the lowest part of the system modules.

Uses/Interactions:

This module will work only if the system is receiving signals from seismic sensors. If the system is working in simulation mode, then this module will not be initialized by the system.

This module does not use any other module but it is used by Communication Component. For detail, please look at the class diagram.

Resources:

This module needs sensors and gateway, since the duty of this component is to parse the coming data form sensors. As mentioned in the Constraints part, this module requires the sensors be sending data in a suitable frequency. If this component cannot reach the speed of the sensors then the some of the data coming from the sensors are lost.

Processing:

This component receives data as byte stream. After receiving the data, it divides the data in subparts by a brute force parsing algorithm. To do this, Stream Converter Component has a parse method.

The coming data after interpretation are stored in a structure where id of sensor, signal strength and being alive values are separated from each other.

If this component receives NULL/no byte stream, it simply throws an NO_BYTE_SREAM exception.

Interface/Exports:

This module has a parse subroutine and vector of a special structure in which heart beat/being alive of the sensor and signal strength is send to Communication Component. The structure has Being alive as a boolean and the signal strength as an integer in interval [0, 1024]. Any further sensor related data can be added here in the future if it is needed, such as temperature.

7.2 COMMUNICATION COMPONENT

Classification:

Communication Component is a module of the system.

Definition:

This component provides the communication between the simulation core component and the sensors as its name implies. Actually there is Stream Converter Component between seismic sensors and this component but this component is the main responsible of the communication.

Responsibilities:

This component's responsibility is to send the meaningful sensor data to Simulator Core Component. After the byte stream coming from the seismic sensors converted to interpretable values by Stream Converter Component, this module transport them to Simulator Component where signal strength is an integer in interval [0, 1024].

Constraints:

This module assumes that the sensors are sending data and Stream Converter Component works correctly. This component is from the real time part of the system like Stream Converter Component and this component needs that the frequency of the sensors' signal sending is catchable.

Composition:

This module contains an instance of Stream Converter module and a vector of Sensor Component instances.

Uses/Interactions:

This components use its Stream Converter Module instance' parse method to interpret the byte stream. This module is also cooperates with Sensor Component. After interpreting the data, this module updates the Sensor instances which are gathered in vector and after update operation they are sent to Simulator Core Component.

Those Sensor module instances belong to Simulator Core Component and they are created by users manually. After creation of each sensor instance, they get a unique id and the mentioned update process is done by checking those ids.

Resources:

This component works in real time and requires a suitable signal frequency since it works by using Stream Converter Component instance. Those requirements are explained in Resources part of the Stream Converter Component. Please look there for more detail.

Processing:

This module receives the data coming from sensors. After accepting the data, that byte stream is processed by Stream Converter Component instance in this part. Stream Converter module has a parse method to separate the data into meaningful portions. This method sends the converted data back to this component. After taking the data back from the parse method, this module checks the sensor id of the sender seismic sensor. This module fetches a vector of sensors from Simulator Core Component. By checking the sensor ids, the module finds the sender sensor

among the other sensors in the vector. After finding the sensor, this module updates the data values of the sensor and finally sends back all the sensors to Simulator Core Component.

If NO_BYTE_SREAM exception is thrown by the instance of Stream Converter module's parse method, this exception is caught in this module and no update is done on the Sensor instances.

Interface/Exports:

As mentioned above, this module uses parse method of Stream Converter Component via an instance. Also this module receives instances of Sensor Component.

This module provides a subroutine as updateSensorData. This method takes a vector of Sensor objects as parameter. All the updates mentioned above are done in this subroutine. After this subroutine finishes, as mentioned above, the Sensor instances' data values are updated.

7.3 SENSOR COMPONENT

Classification:

Sensor Component is a class of the system.

Definition:

This module represents the sensors in software environment. Because of this fact, the name of the component is Sensor. Basic information related to sensors are obtained from this class such as signal strength, world coordinates etc.

Responsibilities:

This component represents sensors. They will hold the core information of sensors. The necessary data like world coordinates of a sensor, signal strength and being alive condition are accessed via this class in software environment.

Constraints:

This class will be hold in a vector in Simulator Core Component. In the vector, there will be pointers to the sensor instances. Other components will reach them by taking that vector. They will not be accessed other than the methods explained in this 7.3 part.

Composition:

None, this component is at the lowest level of the system.

Uses/Interactions:

The sensor objects will be hold in a vector in Simulator Core Component. Other modules will reach them via referencing. Simulator Core Component will be displaying the changes in the simulation environment depending upon the sensor objects it has.

When hardware sensors send data to Communication Component, it will reach the sensor vector of the Simulator Core Component. Update will be done on them. Simulator Core Component will send the vector to Target Tracking Component. Target Tracking Component will interpret the differences and predict a new track.

If the system is not working with hardware sensors, the users should have entered imaginary sensors and a track manually on the simulation environment. Simulation Core Component similar to above explanation, will create Sensor objects and the system will work depending on the imaginary data just to check the efficiency of the Target Tracking Algorithm Component.

Resources:

Only necessary resource for this component is memory.

Processing:

This class is a structure to hold the necessary sensor information. All the necessary information is hold in this class related to sensors. This class will make the system able to reach that data easily.

Interface/Exports:

sensorId: integer

Explanation: This id will be unique to each sensor and they will be differentiate from each other by checking this id.

isAlive: boolean

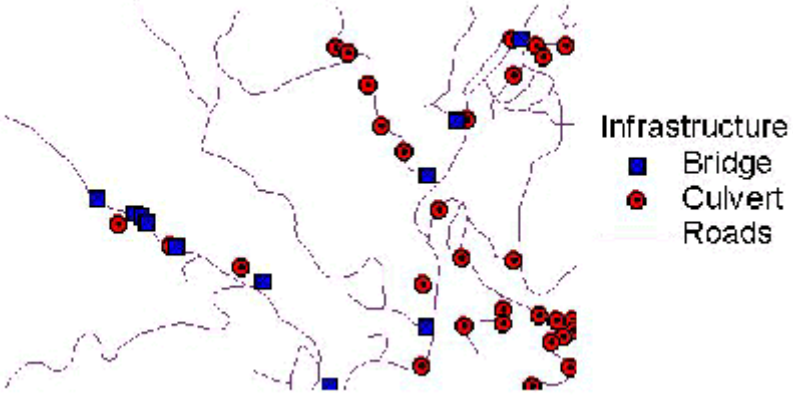
Explanation: This boolean will hold the condition of the sensor. If a sensor is thought to be sending wrong information, then the user will be able to kill this sensor. If this value of a sensor is set to False, this means that this sensor is not alive and the data it sends should discarded.

signalStrength: integer

Explanation: This integer will hold the strength of the signal. If a seismic wave is detected by sensor, it will send values which are bigger than 0 and smaller than 1024. The interval of this integer [0,1024]. 0 means no seismic detection and 1024 means maximum strength.

sensorPosition: GIS Vector Point

Explanation: It will hold x and y coordinates of the sensor. We will not use z coordinate, simply work in 2D. Each sensor will be detected and drawn on the map by GIS coordinates.



A simple vector map, GIS point data showing the location of bridges and culverts.

7.4 TRACK COMPONENT

Classification:

This will be a class of the system.

Definition:

As its name implies, this class represents tracks in the simulation environment. This class will be needed to hold coordinate like information of tracks.

Responsibilities:

This class will hold the core information of tracks. Track objects will be used to see the path of tracks, i.e. starting point, ending point etc.

Constraints:

There will be at most two instances of this class. This fact will be described later in this section. They will be accessed only as described in this section as well. There is not any other constraint.

Composition:

None, this class is at the lowest level of the design.

Uses/Interactions:

There will be two instances of the Track Component:

One of them will be created if the user uses joystick to draw a track. According the input coming from the joystick and Joystick Component, Track Drawing Component will create a Track object. This will be the input track. The system will use this track to simulate sensors' sent data. In this case, there won't be hardware communication. The system will assume that the formerly located sensors signal strength values as if an intruder passes through the input track.

The other track instance will hold the predicted path information. As sensors send data or they are simulated to send data, the Simulator Core Component will transform the data to Target Tracking Algorithm Component so that Target Tracking Algorithm will interpret this data and calculate the route. The predicted track will be extended after this process and it will be shown by Simulator Core Component on the simulation environment.

There will be pointers to both Track Component instances in Simulator Core Component. They will have a life cycle depending on life cycle of Simulator Core Component instance.

Resources:

Only necessary resource for this component is memory.

Processing:

This class simply will be a structure to hold the necessary track information in together. This class will make the system able to reach that data easily.

Interface/Exports:

Path: GIS vector line

Explanation: In 2D, this attribute will hold the path of a track. Linear features will be displayed at a small scale by this attribute. This will be used in measuring distance as well.

GIS Vector line example: Streams are shown as dashed blue lines and roads as solid black lines in this example.

7.5 JOYSTICK COMPONENT

Classification:

This is a module of the system.

Definition:

This module is to be a translater among joystick hardware and the system. Its name implies this.

Responsibilities:

The responsibility of this module is to accept data from joystick and convert the data into meaningful-to-system data. According to that data Track Drawing Component draws the input track.

Constraints:

This component requires joystick hardware. If the system is accepting input track from users, then this module will work.

Composition:

None, this module does not have any sub module.

Uses/Interactions:

This module will be used by Track Drawing Component. This component accept the data and send it as string to Track Drawing. Joystick hardware is required as mentioned above. Main use of this part is to be an interface between joystick and Track Drawing Component.

Resources:

Instance of this class will be hold in memory and this module also will need joystick. Also here we are going to use the API, given to us by ASELSAN Inc.

Processing:

This module will accept data from joystick. After converting it into string of integers, this string will be sent to Track Drawing.

Interface/Exports:

pathString: String

Explanation: This module will use the API, and store the information in this string.

7.6 TRACK DRAWING COMPONENT

Classification:

This is a module of the system.

Definition:

This module takes the data string which comes from Joystick Component's instance. The users if want to simulate the system and to check the efficiency of the Target Tracking Algorithm, then they enter an imaginary intruder's track by joystick.

As explained above, Joystick Component accepts the input of the hardware. After converting it to string form, this module receives. Interpreting the data, intruder's track is created.

Responsibilities:

This module is responsible from creating imaginary intruder's track. Data coming from Joystick Component is interpreted and a Track Component's instance is created by this component. This Track instance is created at heap memory. And then it is sent to Simulator Core Component from reference since the main user of this track is Simulator Core Component.

Constraints:

This component requires joystick hardware. If the system is accepting input track from users, then this module will work. Also this component depends on the Joystick Component. Joystick Component should be working properly.

Composition:

This module uses Joystick Component; there will be an instance of that class in this module. Data will be coming as string from the Joystick Component and the only duty of this one, is to parse the data and separate it into meaningful parts.

Uses/Interactions:

This component is used by Simulator Core Component. The Simulator Core Component is the main customer of this class since the input track goes there over this component. The Simulator Core Component will have an instance of this component and it will use the methods of this component by that instance.

Resources:

Only memory.

Processing:

This module will use the methods of Joystick Component. There will be an instance of that Component in this one. When the hardware joystick sends the data, Joystick Component will be dealing with it. In real time, Joystick Component will receive the data and obtain coordinate x and y values. Those values are needed by Track Drawing Component. After Track Drawing Component parse the string and extract those coordinate values, it will create a Track object in heap memory. After building the Track, a reference to it will be given to the Simulator Core Componet.

Interface/Exports:

parseJoystickString: method

Explanation: This method will parse the string coming from Joystick Component.

inputTrack: Track

Explanation: This object will be created by interpreting the coordinate values obtained after parsing.

7.7 TARGET TRACKING COMPONENT

Classification:

This is a module of the whole system.

Definition:

This module calculates the track of the intruder depending upon data coming from sensors. As its name implies, this module is the place where target tracking algorithm is used. This is one of two main modules of the system.

Responsibilities:

This module uses target tracking algorithm and calculates the path of the intruder.

Constraints:

There are two ways to use this component. If the system is accepting data from real hardware sensors, this data is sent to Target Tracking Algorithm Component by Simulator Core Component and the resulting track is received by Simulator Core Component again. In order to work in this way, the sensors should be sending data successively. On the other hand, if the system is working by simulating sensors, Simulator Core Component will be sending imaginary intruder's sensor related data by using imputed track by joystick.

Composition:

This module has instances of Database component, Track component and Sensor component.

Uses/Interactions:

This module will be used by Simulator Core Component. Sensor related data will be sent to this component over Simulator Core Component. This module will predict a track for the intruder and this predicted track will be sent back to the Simulator Core Component. In order to achieve this task, this component will be working with instances of Track and Sensor classes.

Moreover this component will be sending the predicted track information to Database Component to store it in the database of the system.

Resources:

This module needs database and memory.

Processing:

As explained above, Simulator Core Component will invoke this component. The necessary sensor related data will be coming from Simulator Core Component. At each step, this module will calculate the track of the intruder. The formerly predicted path will be elongated and sent to database to store. Also this path will be sent to Simulator Core Component to be displayed at simulation environment.

Sensor data will be reached a vector of Sensor class instances which will be given by simulator Core Component and predicted path will be an instance of Track class.

Interface/Exports:

calculatePath: method

Explanation: This method will calculate the path of the intruder and return a Track instance as explained above.

sendToDatabase: method

Explanation: This method will send the necessary information of a track to the database in the necessary format.

7.8 DATABASE COMPONENT

Classification:

This is a module of the system.

Definition:

This module arranges the accesses to database of the system. Each module reaches database over this module.

Responsibilities:

This module provides database connection and storing data in the database. Aim of this module is to ease the database access.

Constraints:

The user entered the system should be registered. Registered users of the system will be given root privileges to access database. i.e.: with root password and root name.

Composition:

None.

Uses/Interactions:

This module is used by Target Tracking Algorithm Component and Simulator Core Component of the system.

Resources:

This module should have database. For our system, it will be PostgreSQL.

Processing:

Track information and sensor information will be hold in the database. This module will enable Simulator Core Component and Target Tracking Algorithm Component to send their data to database.

Interface/Exports:

addSensor: method

Explanation: This method will add a new sensor to the database.

addNewSignalStrength: method

Explanation: This method will add new signal strength value of a sensor by finding it from its id.

addTrack: method

Explanation: This method will add information of a track to the database.

updateTrackData: method

Explanation: This method will update the information of a track in the database.

7.9 GIS COMPONENT

Classification:

This is a module of the system.

Definition:

This module will manipulate the GIS API.

Responsibilities:

Duty of this module is to enable system to use GIS API. In other words, this module allows us to view, understand, question, interpret, and visualize data in many ways that reveal relationships and patterns in the form of GIS map.

Constraints:

GIS API should be working on the system.

Composition:

None.

Uses/Interactions:

This component is used by Simulator Core Component. Simulator Core Component will display Sensors and Tracks on simulation environment by using this component.

Also Sensor and Track Components will use this component to hold their 2D coordinations.

Resources:

GIS API is required for this component.

Processing:

Sensor class will hold a GIS vector point to designate its 2D coordinates. Also Track Component will hold a GIS vector line for same purpose. Simulation Core Component will be the one which operates with those information.

Interface/Exports:

locateToMap: method

Explanation: This method gives a 2D coordinate to an object.

locateLineOnMap: method

Explanation: This method will create a Track object's GIS vector line data. This will be used to detect track over 2D map.

deleteFromMap: method

Explanation: This method will delete any object from the map.

7.10 SIMULATOR CORE COMPONENT

Classification:

This is a module of the system.

Definition:

This module will manipulate the simulation environment. This module is one of the two main modules of the system.

Responsibilities:

This module will be providing GUI. The users will be entering sensor positions, joystick used tracks and report commands via this module. Sensor objects and tracks will be hold here. Target Tracking Algorithm Component will be called from this module. After each data acceptance from imaginary or real sensors, those data will be sent to Database over Simulator Core Component. The updated track will also be shown to the users from here as well.

Constraints:

This module assumes that all other parts of the system works correctly.

Composition:

This module contains a vector of Sensors, Track instances for input and output. There will also be a Track Creator instance to take pdf format output from the system.

Uses/Interactions:

Coming data from hardware sensors will be gathered by this component. If the system is not using hardware sensors, the Simulation Core Component will simulate it. Sensor Component instances will be activated and a data flow is created by this component. After this step, the data will be directed to Target Tracking Algorithm Component to interpret it and predict the intruder's path. Generated sensor data will be sent to Database Component to save as well. The predicted track will be sent to Simulation Core Component to display on the simulation environment. Simulation Core Component will use GIS Component to get 2D coordinate values and NASA World Wind API to create a simulation environment.

When users want to take pdf output from the system, they will again use this component. This component will provide an interface for this task. The Simulation Core Component will have an instance of Track Creator module and by using this module, it will generate pdf format output data.

Resources:

This component will be using the every resource that is needed by the system.

Processing:

When the system is started, users first will see a GUI which is created by this component. This GUI will provide two options to users: They may use hardware sensors or they may simulate it. If they want to simulate it, they will enter a track by joystick. This component will simulate sensors to be sending data regularly. Otherwise the system will work in real time. This time the

component will be gathering data from hardware sensors. Those data will be stored in database as well.

The data will be directed to Target Tracking Algorithm Component in order to predict a track. This is the most crucial part of our project. After creating the track, Simulation Core Component will display it in simulation environment too.

When users want to take pdf output from the system this component will provide an interface for this task. Simulation Core Component has an instance of Track Creator Component. By invoking this module, the output will be given to users in pdf format.

Interface/Exports:

This part is thought to be core of main system. Every service of the system is reached from here by users.

7.11 TRACK CREATOR COMPONENT

Classification:

This is a module of the system.

Definition:

This module creates logical data as a track of an intruder to print.

Responsibilities:

This module arranges the output pdf file format and invokes Reporter Component to output pdf file.

Constraints:

Reporter Component is to be working correctly.

Composition:

This module contains Reporter Component.

Uses/Interactions:

This module is used by Simulation Core Component, the output pdf files are given to users by using this module.

Resources:

This module requires printer and memory. Also this module will take its data form database.

Processing:

This module will read data form database and print the developed track in a predefined format.

Interface/Exports:

parse: method

Explanation: This method will get data from database and put it into predefined format to be printed.

7.12 REPORT COMPONENT

Classification:

This is a module of the whole system.

Definition:

This module outputs pdf files as reports. Because of this fact, its name is Reporter Component.

Responsibilities:

This module converts its string attribute into pdf file. This module will work under Track Creator Component. Its only responsibility is to creating pdf files by using APIs which are given to us by ASELSAN Inc.

Constraints:

The format of the pdf file is arranged by Track Creator Component. This module assumes that the format is correct and user friendly.

Composition:

None.

Uses/Interactions:

This module uses APIs and it is used by Track Creator Component.

Resources:

Printer and memory are required.

Processing:

This module will be invoked by Track Creator Component. First, that component will deploy the expected data into data String of this component. Then this module will start to create a pdf file from its String attribute.

Interface/Exports:

There is only data string of this module. This attribute will be assigned to expected data by calling clients of this module.

REFERENCES

1. http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/
2. http://www.agilemodeling.com/artifacts/classDiagram.htm
3. http://gislounge.com/geodatabases-explored-vector-and-raster-data/

8 LIBRARIES AND TOOLS

8.1 ECLIPSE

It is an IDE that we are going to use for design. The Eclipse community has a well earned reputation for providing quality software in a reliable and predictable fashion. This is due to the commitment of the committers and organizations contributing to the open source projects. The Eclipse Foundation also provides services and support for the projects to help them achieve these goals. It is really compatible with Java programming language which is the language that are we are going to use for our design. Therefore we have chosen this IDE to the design the system.

8.2 POSTGRESQL

PostgreSQL is an open source object-relational database system. . It has native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, among others.This software has been designed and created to have much lower maintenance and tuning requirements than the leading proprietary databases, yet still retain all of the features, stability, and performance.In addition to this, this training programs are generally regarded as being far more cost effective, manageable, and practical in the real world than that of the leading proprietary database vendors. According to the things that i mentioned above we have chosen this database system.

8.3 JAVA

Because of the libraries provided of use, we are going to use Java.

8.3.1 JXINPUT

JXinput library is used for input devices like joysticks,gamepads for Java. For computer without a phyisical joystick attached, JXInput offers the possibility to emulate 'virtual' joystick axes with help from a set of buttons. So it is very easy to use a 'virtual' joystick by using the e.g. the cursor keys.Since we will use this joystick during target tracking at our GIS based map, we will use this library.

8.3.2 NASA WORLD WIND

The WWJ (World Wind Java) SDK is a 3D graphics globe built on top of the Java OpenGL (JOGL) extensions. At the core of the WWJ class hierarchy is the WorldWindowGLCanvas, which is a subclass of GLCanvas. GLCanvas is in turn an Abstract Window Toolkit (AWT) component. WWJ's dependence on AWT is an obstacle for GIS developers who want to use WWJ in their Eclipse applications. Eclipse uses the Standard Widget Toolkit (SWT), which is incompatible with AWT. Furthermore, AWT and JOGL are tightly integrated, making a port of the AWT interfaces to SWT difficult. These builts presents a solution that enables us to use the WWJ SDK with our Eclipse applications.

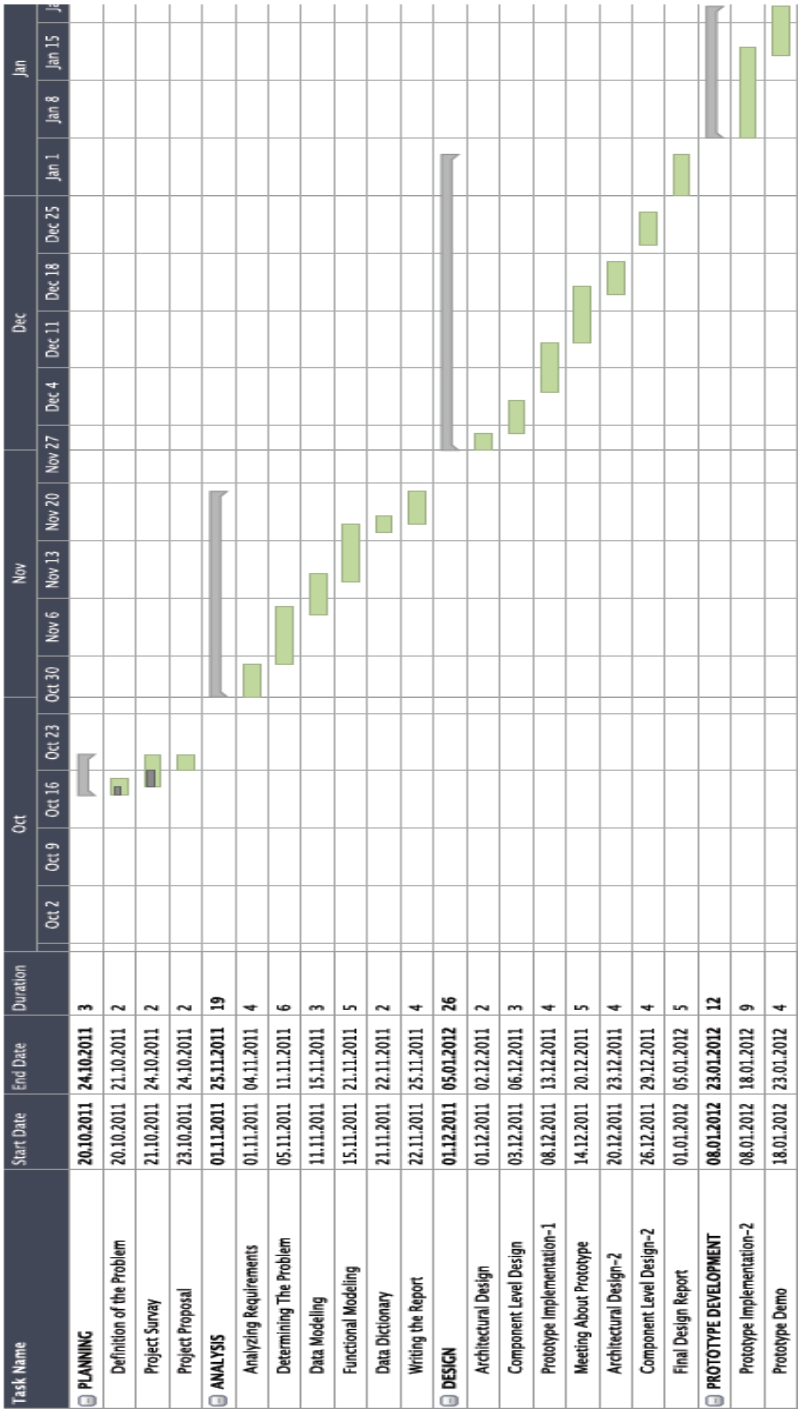
8.3.3 JOYSTICK COMPONENT

JReporter is a Java class library to ease the development of Java applications which provide printing capabilities. It is designed to provide the following features:

- **Interface for Data:** Provides an interface for printing data from various data sources. This relieves the need to copy data into new data structures just for printing. This is in much the same spirit as JTable/TableModel.
- **Interfaces for printing:** Intelligent layout agents which allow complex layouts on a page.
- **Default Implementations:** Default implementations are being written to provide flexible printing formats. The DefaultDataItem knows how to draw data on a report, in addition to drawing borders around the data, padding data cells with margins, and performing justification.
- **Report Classes:** Classes will be provided which allows a programmer to put various report items together and print them out as a group. Report are themselves ReportItems, so it is quite easy to construct a large report which is itself a set of sub-reports.
- **Dialog Framework:** A framework is provided for extensible report setup dialogs, including page setup information. Programmers can write classes that will easily work with the existing dialog.
- **Basic Report:** Basic reports which can be used directly without learning much about the underlying structure of the system.

9 TIME PLANNING

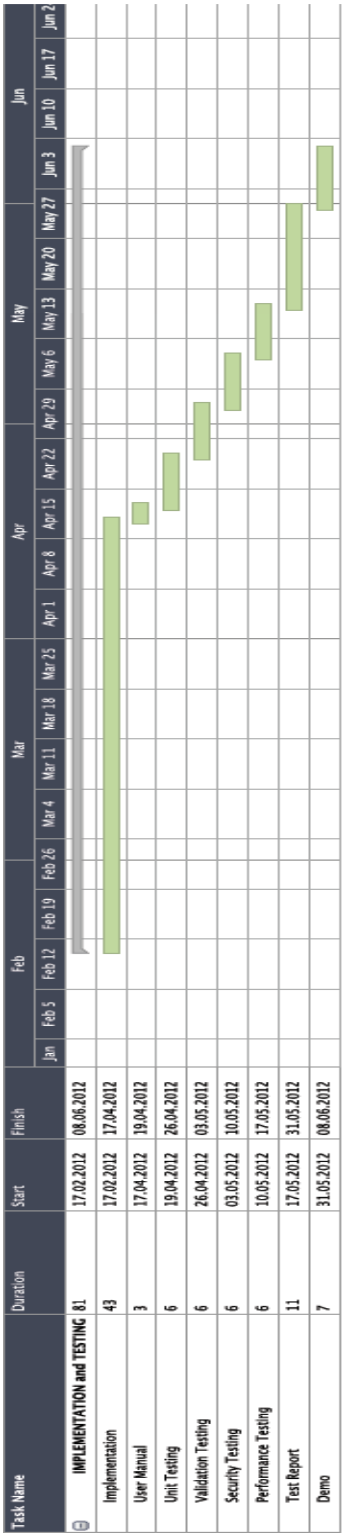
9.1 TERM I GANNT CHART



Task Name	Start Date	End Date	Duration
PLANNING	20.10.2011	24.10.2011	3
Definiton of the Problem	20.10.2011	21.10.2011	2
Project Survay	21.10.2011	24.10.2011	2
Project Proposal	23.10.2011	24.10.2011	2
ANALYSIS	01.11.2011	25.11.2011	19
Analyzing Requirements	01.11.2011	04.11.2011	4
Determining The Problem	05.11.2011	11.11.2011	6
Data Modeling	11.11.2011	15.11.2011	3
Functional Modeling	15.11.2011	21.11.2011	5
Data Dictionary	21.11.2011	22.11.2011	2
Writing the Report	22.11.2011	25.11.2011	4
DESIGN	01.12.2011	05.01.2012	26
Architectural Design	01.12.2011	02.12.2011	2
Component Level Design	03.12.2011	06.12.2011	3
Prototype Implementation-1	08.12.2011	13.12.2011	4
Meeting About Prototype	14.12.2011	20.12.2011	5
Architectural Design-2	20.12.2011	23.12.2011	4
Component Level Design-2	26.12.2011	29.12.2011	4
Final Design Report	01.01.2012	05.01.2012	5
PROTOTYPE DEVELOPMENT	08.01.2012	23.01.2012	12
Prototype Implementation-2	08.01.2012	18.01.2012	9
Prototype Demo	18.01.2012	23.01.2012	4

Term I Gannt Chart

9.2 TERM II GANNT CHART



Task Name	Duration	Start	Finish
IMPLEMENTATION and TESTING	81	17.02.2012	08.06.2012
Implementation	43	17.02.2012	17.04.2012
User Manual	3	17.04.2012	19.04.2012
Unit Testing	6	19.04.2012	26.04.2012
Validation Testing	6	26.04.2012	03.05.2012
Security Testing	6	03.05.2012	10.05.2012
Performance Testing	6	10.05.2012	17.05.2012
Test Report	11	17.05.2012	31.05.2012
Demo	7	31.05.2012	08.06.2012

Term II Gannt Chart

10 CONCLUSION

In conclusion, Initial Design Report gives the definition, purpose and scope of the project. The possible design and other constraints that can be faced are explained. The tools and the libraries that will be used during developing the project are decided. Data flow models, class diagrams, interface features, entity relationship diagrams, possible use cases are given within the document. We have explained the works that we have done so far and within the schedule we give the future work to be done.