Software Requirement Specification
For
Target Tracking by Using Seismic Sensors

By:

Umut Erdal - 1745926

Edona Fasllija - 1645902

Serkan Yanar - 1746478

Erdoğan Cem Evin - 1678945

Table of Contents

1. Introduction

In this document, we will give software requirement specification of 'target tracking by using seismic sensors'. In this chapter we will provide purpose and scope of this document, then follow with an overall description of the system.

1.1. Problem Definition

In this project, we intend to provide an approach to the problem of motion tracking, which has a wide range of military applications. The study of this problem is based on signal processing and analysis, which suits our way to solve the problem of position determination over a field using by means of wireless 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 software requirement specification document we will describe the behavior of the software to be developed. This document basically explains what our software do and our purpose is to show functional and non-functional requirements, the dependencies and limitations of the project.

1.3. Scope

The software product we will introduce is target tracking by using seismic sensors. The software is intended to be target tracking system that will be used for tracking humans, animals etc. on large fields. The software will be able to simulate tracking path by using virtual seismic sensors hence the user will be able to simulate target tracking algorithm before locating the seismic sensors on to the field. The software will also provide an interface to analyze these data.

1.4. User and Literature Survey

There are a lot of publications about target tracking with wireless sensor network. But we could only find one product available in the market for tracking moving objects with seismic sensor network.

The producer company is Alpha Tech Inc. Alpha Tech also published an article about how the company used Kalman Filter algorithm for predicting path information . Our case is very similar to Alpha Tech's. But even though our software's algorithm will be based on Kalman Filter, we will use

pre-processing techniques on data before using Kalman Filter. The simulation of the target tracking is another difference between our and Alpha Tech's product.

1.5. References

[1] IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications

1.6. Overview

In the first chapter, we will explain our purpose and scope for writing this SRS document. We will also provide some information about this project like which problem we are trying to identify and who will use our project.

In the second chapter, first we will provide a diagram that explains the structure of our software. Then we will give general information about major functions, dependencies, limitations etc.

In third chapter, we will give detailed explanation of specific requirements of this project. First, we will explain user interface, then functional requirements and lastly non-functional requirements.

In fourth chapter, we will provide descriptions about data models that are managed by our software.

In fifth chapter, we will present a description of behavior of the software. We will also provide state transition diagrams.

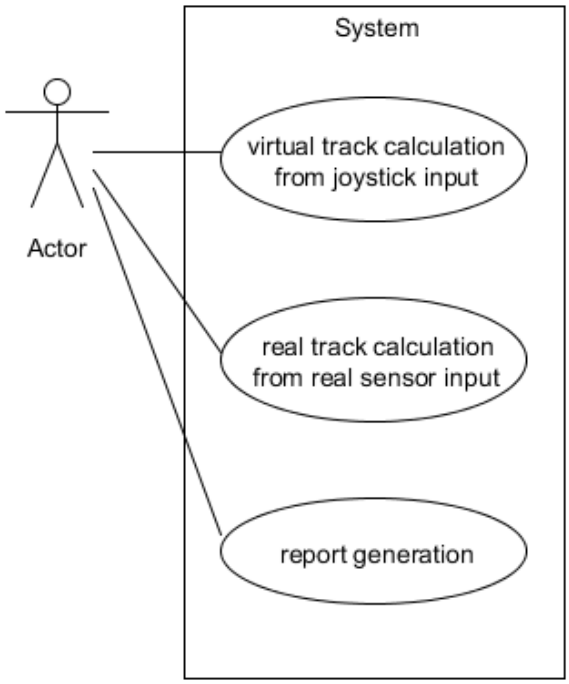
In sixth chapter, we will give information about our team structure and a general schedule for this project.

In seventh and last chapter, we will give conclusion of this SRS document.

2. Overall Description

2.1. Product Perspective

The product will be for the use of Turkish military company ASELSAN. The system will be integrated as a java program which will provide interfaces for the inputs of system. The system will have three external interfaces for users to operate system. The main interface will be user interface which allows user to manage simulation properties and analyze results. The other interface will be communication interface that will be used for communication with seismic sensors. The last interface will be joystick interface which allows user to send track data with joystick-like device to the system. The system will be standalone system and will not maintain an interface for other systems.



2.2. Product Functions

Our system will require specific sequence of inputs from the users to operate correctly. The inputs will be given by the users of the system with the user interface, and if necessary with a joystick-like device. As can be seen from figure below our system will serve to the users in three ways.

1. Selecting the mode of program.

This is the first phase of the system. The user can select between 'virtual track calculation' or 'real track calculation' modes of the system.

2. Starting simulation.

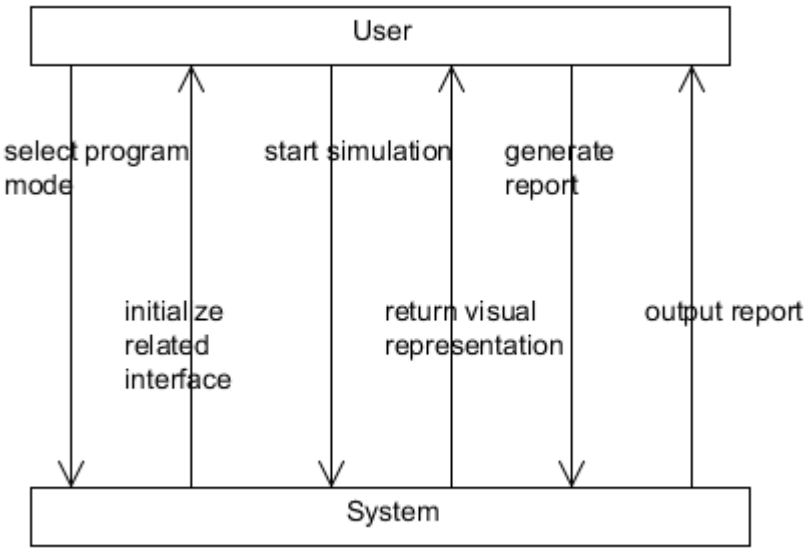
In this phase of program, the user starts to simulate the path of moving object. The simulation can depend on either joystick input from user or real seismic sensor data.

3. Printing report.

In this last phase the user can print out detailed report of the simulation. The report can be about specific information about the simulation which depends on user's choice.

2.3. Constraints, Assumptions and Dependencies

In virtual track calculating mode, our system shall support various types of joystick or keyboards. Also since our system will use Waspmote seismic sensor cards, it shall use the application programming interface of the sensor cards that is provided by Waspmote.



3. Specific Requirements

3.1. Interface Requirements

In the interface requirements part, we will explain the interfaces between the external systems and product components. This part is organized into two section, namely user interface and communication interface. In the user interface section, we will explain individual layers of the user interface. In the communication interface, we will explain the communication interface of the system with the hardware.

3.1.1. User Interface

In this section, we will explain the layers and usage of the user interface.

The user interface shall be composed of two main sections: Map and menu. Map shall show the user the current geographic map, along with sensors and track layers, whereas menu shall allow the user to choose among various options, such as add a sensor, or show a track (will be explained in detail below).

3.1.1.1. Map/GIS Layer

The UI shall have a map layer. This layer shall show the user the current geographic map according to the given coordinate inputs. This layer shall work parallel with GIS library. This layer shall be a base for other layers which shall be added on top of this one.

3.1.1.2. Sensor Layer

The sensor layer shall show the sensors placed on the map visually to the user. This layer shall be visible on top of the map layer. It shall be switched on/off via the user.

In case of an alarm with some sensors, the relevant sensors shall be indicated as alarming on the map.

This layer shall also allow the user to add a sensor to the system visually, by clicking on a place on the map, when the user chooses to add a sensor from the menu.

3.1.1.3. Track Layer

The track layer shall show tracks on the map. This layer shall show the real track, which is given by the user via an input device, such as joystick. It shall also show the track generated by the algorithm. The tracks on the map shall be updated real-time.

When the user selects a track from the menu, the user shall be able to see that specific track on this layer.

3.1.1.4. Menu

The menu shall let the user toggle visibility of individual layers. It shall also include sensor and track list. From those lists, the user shall be able to enable/disable each sensor and toggle visibility of each track.

The menu shall also let the user add sensors to the map. The user also shall be able to start the input of a new track, end and replay a track from the menu.

3.1.2. Communication Interface

The communication interface shall serve as an interface layer between the hardware and the software. It shall deal with hardware – api related issues and render the information taken from the hardware to a more human-friendly format. It shall also handle the information flow from the system to the hardware.

3.1.2.1. Sensor interface

The sensor interface shall provide an abstraction layer between the system and the sensors. It shall convert the input from the sensors into a more human-friendly version. It shall also convert the information from the system to a version that the sensors can interpret.

3.1.2.2. User input interface

The user input interface shall provide an abstraction layer between the system and possible input devices that user can use, such as joystick, keyboard or mouse. It shall convert all inputs into a common format, which will be interpreted by the system.

3.2. Functional Requirements

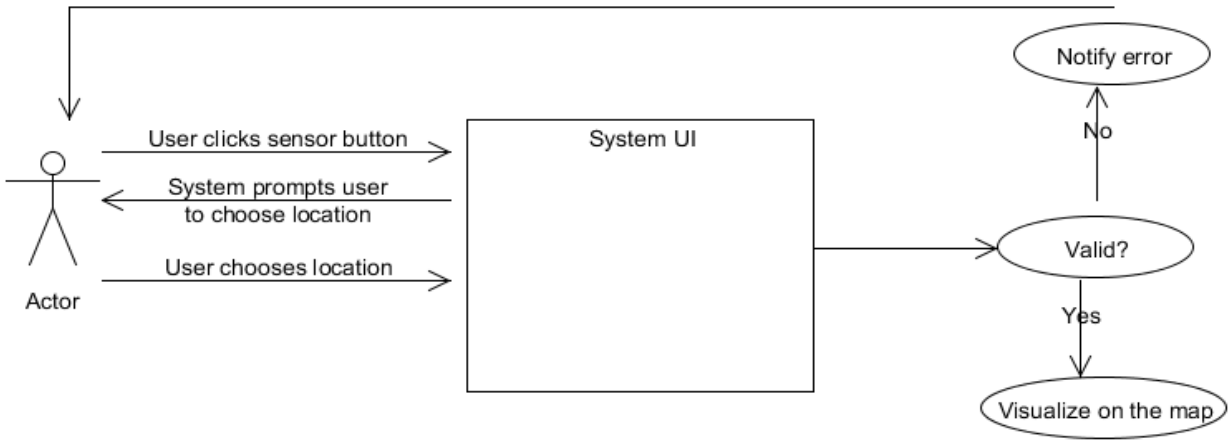
3.2.1. Adding a sensor

3.2.1.1. Background Information

The user shall choose to add a sensor through menu in the UI of the system, as explained in 3.1.1. When the user clicks the button, the user shall be prompted to click on a place in the map, to visually place the sensor. The user shall be able to cancel the operation.

3.2.1.2. Stimulus/Response Sequences

3.2.1.2.1. Diagram



3.2.1.2.2. Description

Primary Actor	User of the system
Goal In Context	To add a sensor on the map, which will ready for simulation realtime.
Trigger	User clicks add sensor button on the menu

3.2.1.2.3. Normal Flow of Events

1. User clicks add sensor button on the menu.

2. User is prompted to choose where to place the sensor.

3. User clicks on a place on the map.

4. The sensor is placed on the map and in the sensors list.

3.2.1.2.4. Alternative Event Flows

3. User clicks cancel before placing the sensor.

4. The operation is canceled.

3.2.1.3. Functional Requirements

1. The system shall check the validity of the place which the user has clicked on the map.

2. The system shall handle cancellation properly.

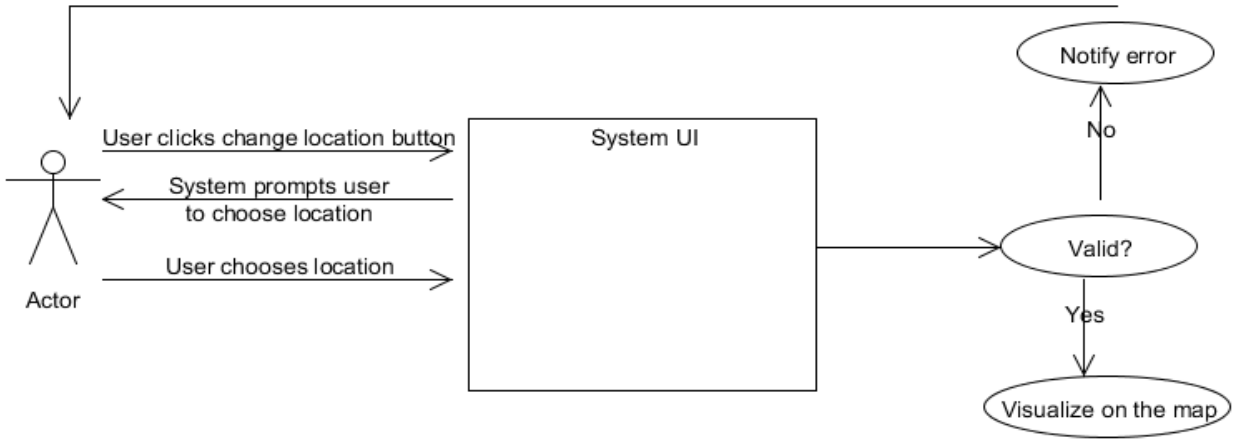
3.2.2. Changing the location of a sensor

3.2.2.1. Background Information

The user shall choose to change the location of a sensor through menu in the UI of the system. When the user clicks the Change Sensor location button, the user shall be prompted to drag and drop the sensor on a place in the map, to visually relocate the sensor. The user shall be able to cancel the operation.

3.2.2.2. Stimulus/Response Sequences

3.2.2.2.1. Diagram



3.2.2.2.2. Description

Primary Actor	User of the system
Goal In Context	To change the location of a sensor in the map.
Trigger	User clicks change sensor location button on the menu

3.2.2.2.3. Normal Flow of Events

1. User clicks change sensor location button on the menu.

2. User is prompted to choose a sensor, drag it to where to place the sensor.

3. User drops the sensor on a place on the map.

4. The sensor location is updated on the map, the sensor list and the database .

3.2.2.2.4. Alternative Event Flows

Alternative Event Flow 1

3. User clicks cancel before placing the sensor.

4. The operation is canceled.

3.2.2.3. Functional Requirements

1. The system shall check the validity of the place which the user has clicked on the map.

2. The system shall handle cancellation properly.

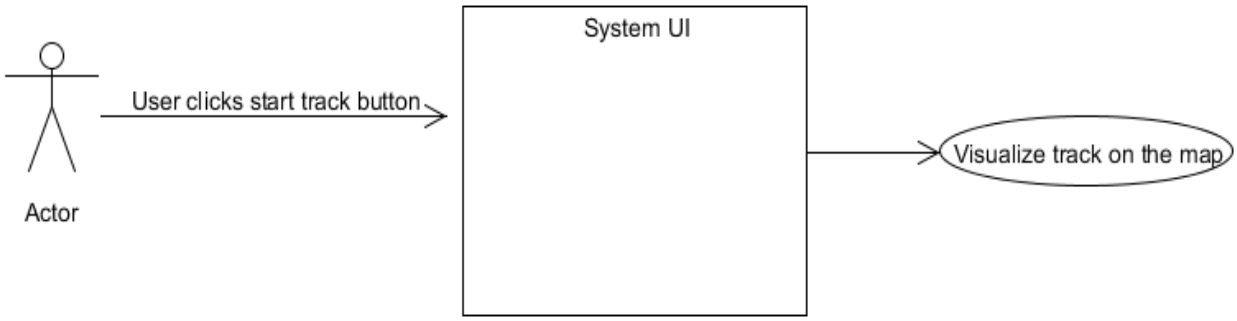
3.2.3. Starting a track

3.2.3.1. Background Information

The user shall choose to start a track through the menu. When the user clicks on the button, the system shall start listening to the input given by the user. The system shall also start drawing the real track and algorithm-generated track to the track layer of the UI map in real-time. The track shall be added to the track list.

3.2.3.2. Stimulus/Response Sequences

3.2.3.2.1. Diagram



3.2.3.2.2. Description

Primary Actor	User of the system
Goal In Context	To start giving input for a track
Trigger	User clicks start track button on the menu

3.2.3.2.3. Normal Flow of Events

1. User clicks start track button on the menu.

2. The system starts drawing tracks on the map in real-time.

3.2.3.3. Functional Requirements

1. The system shall notify the relevant components on the incoming input and beginning of the new track.

2. The system shall draw both the input track and the algorithm-generated track on the track layer of the map.

3. The system shall add the new track to the tracks list on the menu.

4. The system shall disable the start track button and enable the end track button on the menu of the UI, since the user shall be able to give only one input at a time.

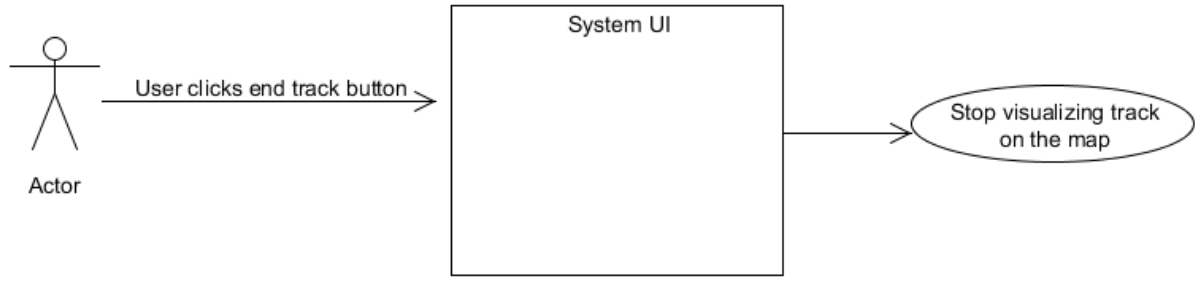
3.2.4. Ending a track

3.2.4.1. Background Information

The user shall choose to end the active track. The UI shall end the drawing of both real and calculated tracks.

3.2.4.2. Stimulus/Response Sequences

3.2.4.2.1. Diagram



3.2.4.2.2. Description

Primary Actor	User of the system
Goal In Context	To stop giving input for a track
Trigger	User clicks stop track button on the menu

3.2.4.2.3. Normal Flow of Events

1. User clicks stop track button on the menu.

2. The system stops drawing tracks on the map in real-time.

3.2.4.3. Functional Requirements

1. The system shall notify the relevant components that the input of the new track has stopped.

2. The system shall disable the end track button and enable the start track button on the menu of the UI.

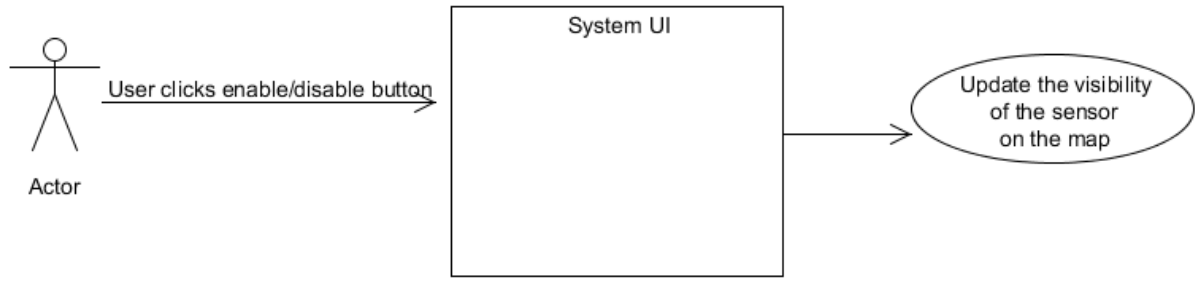
3.2.5. Enable/disable a sensor

3.2.5.1. Background Information

The user shall be able to enable or disable a sensor from the sensors list in the menu in the UI. When the sensor is enabled/disabled, it shall be indicated so in the map, in the sensors layer. The user shall be able to fulfill this operation by clicking on a check box next to the sensor in the sensors list.

3.2.5.2. Stimulus/Response Sequences

3.2.5.2.1. Diagram



3.2.5.2.2. Description

Primary Actor	User of the system
Goal In Context	To enable/disable the sensor
Trigger	User clicks a check box in the sensors list

3.2.5.2.3. Normal Flow of Events

1. User clicks the check box of a sensor in the sensors list.

2. The system enables or disables the sensor. The on/off state of the sensor on the map is updated.

3.2.5.3. Functional Requirements

1. The system shall notify the relevant components that the sensor is switched on/off.

2. The system shall handle the visual changes of the sensor in the sensor layer of the map accordingly.

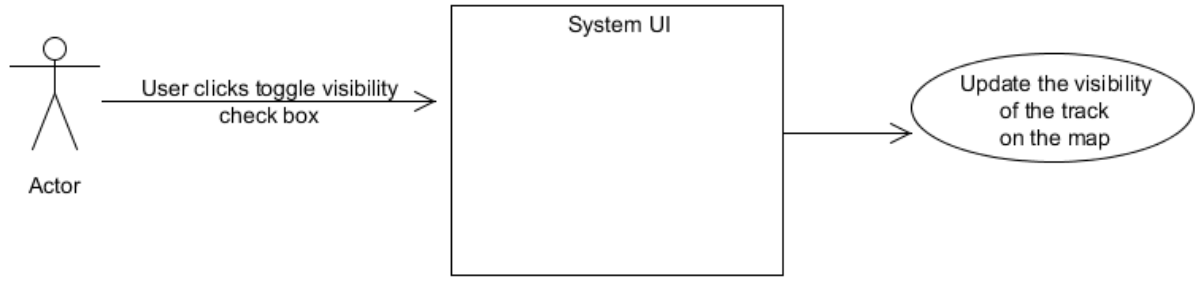
3.2.6. Toggle visibility of a track

3.2.6.1. Background Information

Similar to enabling/disabling a sensor, the user shall be able to toggle visibility of each track from tracks list in the menu of the UI, via a check box next to the track in the list.

3.2.6.2. Stimulus/Response Sequences

3.2.6.2.1. Diagram



3.2.6.2.2. Description

Primary Actor	User of the system
Goal In Context	To toggle visibility of the track
Trigger	User clicks a check box in the tracks list

3.2.6.2.3. Normal Flow of Events

1. User clicks the check box of a track in the tracks list.

2. The system makes the track visible/invisible on the map, in the tracks layer.

3.2.6.3. Functional Requirements

1. The system shall handle the visual changes of the track in the track layer of the map accordingly.

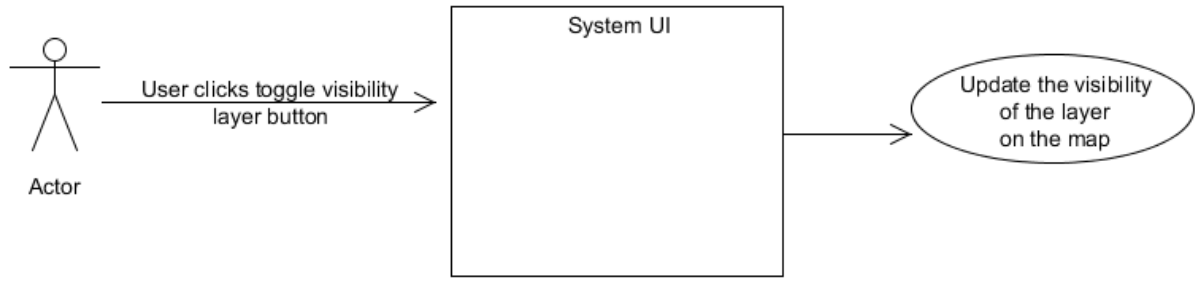
3.2.7. Toggle visibility of a layer

3.2.7.1. Background Information

The user shall be able to toggle visibility of sensor and track layers by clicking on buttons for each layer.

3.2.7.2. Stimulus/Response Sequences

3.2.7.2.1. Diagram



3.2.7.2.2. Description

Primary Actor	User of the system
Goal In Context	To toggle visibility of the sensor or track layer.
Trigger	User clicks a button for making a layer visible/invisible

3.2.7.2.3. Normal Flow of Events

1. User clicks the show/hide button of sensor or track layer.

2. The system makes the layer visible/invisible on the map.

3.2.7.3. Functional Requirements

1. The system shall handle the visual changes of the layer in the map accordingly.

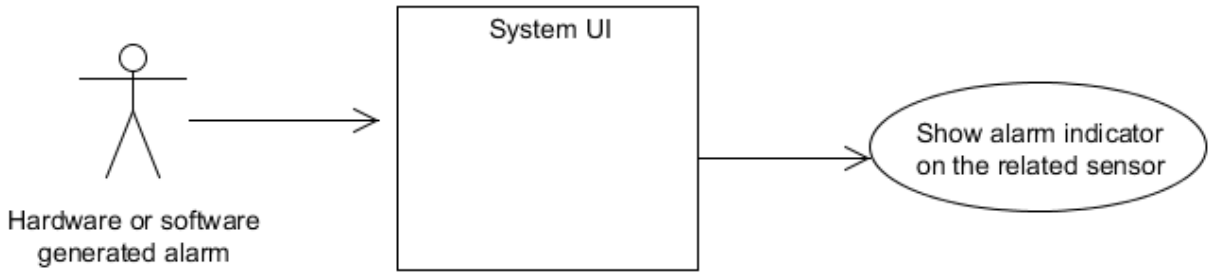
3.2.8. Show sensor alarm

3.2.8.1. Background Information

The system shall indicate an incoming alarm from the sensors, on the map, in the sensor layer, in the UI. The alarm shall be like a flashing circular indication on the sensor, also showing the strength of the alarm.

3.2.8.2. Stimulus/Response Sequences

3.2.8.2.1. Diagram



3.2.8.2.2. Description

Primary Actor	A real/software simulated sensor
Goal In Context	To display the incoming alarm from a sensor visible on the map
Trigger	An alarm arises from a sensor and is sent to the system

3.2.8.2.3. Normal Flow of Events

1. A software alarm is generated for a sensor due to the given track input from the user.

2. The alarm is shown on the sensor in the map for a few seconds.

3.2.8.2.4. Alternative Flow of Events

1. An alarm is generated by a real sensor, due to a real input in the field where the sensor is located.

2. The alarm is shown on the sensor in the map for a few seconds.

3.2.8.3. Functional Requirements

1. The system shall show a red flashing circular alarm on the sensor which has generated input.

2. The system shall show the input only if the sensor layer is visible.

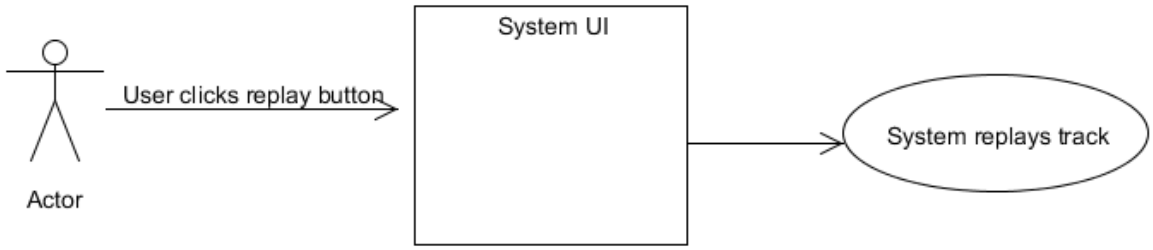
3.2.9. Replay a track

3.2.9.1. Background Information

The system shall be able to choose a track from the track list and click on the replay button to replay the track. The replay shall be as a step-by-step animation, showing the progress of the real track and algorithm-generated track.

3.2.9.2. Stimulus/Response Sequences

3.2.9.2.1. Diagram



3.2.9.2.2. Description

Primary Actor	The user of the system
Goal In Context	To replay a previously processed track
Trigger	The user clicks on the replay button

3.2.9.2.3. Normal Flow of Events

1. The user clicks on a track in the tracks list in the menu.

2. The system replays the progress of the real and computed track.

3.2.9.2.4. Alternative Flow of Events

3. The user cancels the replay before it is finished.

4. The system returns the UI to its state before the replay.

3.2.9.3. Functional Requirements

1. The system shall check whether the user has selected any tracks before pressing the replay button. If not, the system shall give a warning to a user, indicating to choose a track.

2. The system shall handle the stopping of a replay properly. If the replay is aborted, the final versions of the real and computed tracks should be visible again on the map.

3. If the track, on which the user has clicked to replay, is invisible, the system shall make the layer visible, replay, and make it invisible after the replay or abortion of the replay.

3.2.10. Setting the states of software sensors

3.2.10.1. Background Information

When user wants to enter the track by 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.

3.2.10.2. Stimulus/Response Sequences

3.2.10.2.1. Description

Primary Actor	A software seismic sensor
Goal In Context	To inform the system about their states.
Trigger	An input is given by a user.

3.2.10.2.2. Normal Flow of Events

1. I/O device sends an input from user.

2. Every software sensor compares its location and given location from user.

3. According to the distance between the location of the software sensor and given location, this software sensor sets its alarm level.

3.2.10.2.3. Alternative Flow of Events

1. I/O device sends an input from user.

2. Every software sensor compares its location and given location from user.

3. If the software sensor is too far away from the motion it doesn't generate an alarm signal or set its alarm level to zero.

3.2.11. Getting the states of the software sensors

3.2.11.1. Background Information

The simulator component is responsible to get the states of the software sensors. When input is given from I/O, every software sensor will generate an alarm signal. The strength of the alarm is reversely proportional to distance with the software sensor location and location of the simulated target.

3.2.11.2. Stimulus/Response Sequences

3.2.11.2.1. Description

Primary Actor	The simulator component of the system
Goal In Context	To get information about the states of the software sensors
Trigger	Simulator component gets a signal from its internal clock.

3.2.11.2.2. Normal Flow of Events

1. A clock signal comes.

2. Software sensors generate alarm signals.

3. Simulator receives the new states of the sensors by checking every software sensor's alarm strength.

4. Simulator writes the new states to Database and sends them to algorithm.

3.2.11.2.3. Alternative Flow of Events

1. A clock signal comes.

2. Simulator receives no signal since there is no input from I/O.

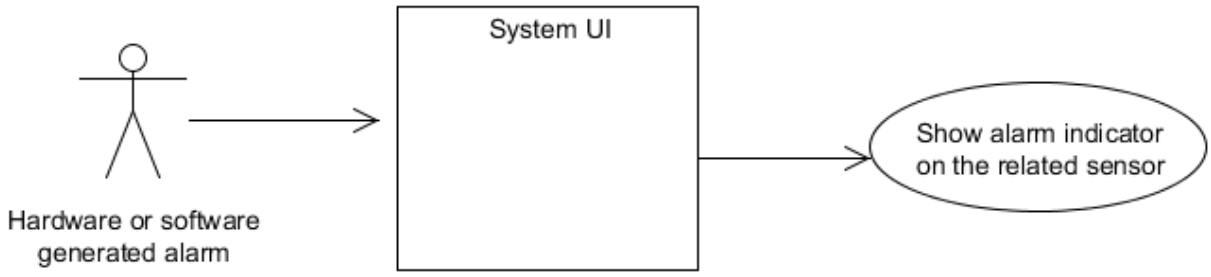
3.2.12. Sending an alarm signal

3.2.12.1. Background Information

If there is a vibration, 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. Communication component holds this information with sensor id at its memory.

3.2.12.2. Stimulus/Response Sequences

3.2.12.2.1. Diagram



3.2.12.2.2. Description

Primary Actor	A real seismic sensor
Goal In Context	To inform the system about the vibrations.
Trigger	The seismic sensor detects a vibration.

3.2.12.2.3. Normal Flow of Events

1. A vibration occurs due to targets motion.

2. A real seismic sensor detects it and generates an alarm signal.

3. This sensor sends this alarm to the communication component.

3.2.12.2.4. Alternative Flow of Events

1. A vibration occurs due to targets motion.

2. If the sensor is too far away from the motion it doesn't generate an alarm signal.

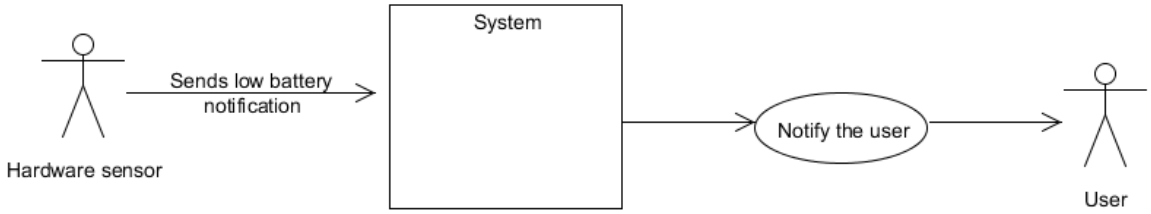
3.2.13. Sending a battery level signal

3.2.13.1. Background Information

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 sensor generates an signal and sends this to the communication component.

3.2.13.2. Stimulus/Response Sequences

3.2.13.2.1. Diagram



3.2.13.2.2. Description

Primary Actor	A real seismic sensor
Goal In Context	To inform the system about the battery level of the sensor.
Trigger	The seismic sensor gets a 'battery level low' warning from its components.

3.2.13.2.3. Normal Flow of Events

1. Battery level of a seismic sensor drops below the critical level.

2. This sensor generates a battery low signal.

3. This sensor sends this signal to the communication component.

3.2.13.2.4. Alternative Flow of Events

1. Battery level of a seismic sensor drops but not below the critical level.

2. This sensor doesn't generate a battery low signal.

3.2.14. Getting the alarm and battery level of the real sensors

3.2.14.1. Background Information

The simulator component is responsible to get and send the states of the real sensors from communication component to algorithm component of the system. Simulator has its own clock. This clock generates a signal in every few miliseconds. When a clock signal comes, simulator gets and sends the alarm and battery level of the real sensors from communication component of the system.

3.2.14.2. Stimulus/Response Sequences

3.2.14.2.1. Description

Primary Actor	Simulator component of the system
Goal In Context	To get information about the battery and alarm level of the real sensors
Trigger	Simulator component gets a signal from its internal clock.

3.2.14.2.2. Normal Flow of Events

1. A clock signal comes.

2. Simulator receives information about the battery and alarm level of the real sensors.

3.2.14.2.3. Alternative Flow of Events

1. A clock signal comes.

2. If there is no target on the field, simulator doesn't need to get the alarm level of the real sensors.

3.2.15. Send the states of the real sensors

3.2.15.1. Background Information

The simulator component is responsible to get and send the states of the real sensors from communication component. Simulator has its own clock. This clock generates a signal in every few milliseconds. When a clock signal comes, simulator gets and sends the alarm and battery level of the real sensors from communication component of the system.

3.2.15.2. Stimulus/Response Sequences

3.2.15.2.1. Description

Primary Actor	The simulator component of the system
Goal In Context	To send the states of the real sensors
Trigger	Simulator component gets a signal from its internal clock.

3.2.15.2.2. Normal Flow of Events

1. A clock signal comes.

2. Simulator receives information about the battery and alarm level of the real sensors.

3. Simulator sends information about the battery and alarm level of the real sensors to the algorithm.

3.2.15.2.3. Alternative Flow of Events

1. A clock signal comes.

2. If there is no target on the field, simulator doesn't need to send the alarm level of the real sensors.

3.2.16. Generating Location

3.2.16.1. Background Information

The algorithm component is responsible to generate location of the target by using the states of the sensors. Simulator component sends the states of the real sensors to algorithm. From only this information algorithm generates the current location of the target.

3.2.16.2. Stimulus/Response Sequences

3.2.16.2.1. Description

Primary Actor	The algorithm component of the system
Goal In Context	To generate location of the target from the currrent state of the real sensors
Trigger	Algoritm component gets the state information of the sensors.

3.2.16.2.2. Normal Flow of Events

1. Simulator sends information about alarm level(states) of the real sensors to the algorithm

2. Algorithm generates the current location of the target.

3.2.16.2.3. Alternative Flow of Events

1. Simulator sends information about alarm level(states) of the real sensors to the algorithm

2. If there is no target on the field, algortihm doesn't generate any location information.

3.2.17. Generating track of a target

3.2.17.1. Background Information

Algorithm component of the system 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. Simulator component sends current the states of the real sensors to algorithm. Then, algorithm generates the location and the track of the target's motion using kalman filter. It also stores the previous location of the target.

3.2.17.2. Stimulus/Response Sequences

3.2.17.2.1. Description

Primary Actor	The algorithm component of the system
Goal In Context	To generate track of the target from previous locations and the currrent state of the real sensors.
Trigger	Algorithm component gets the state information of the sensors.

3.2.17.2.2. Normal Flow of Events

1. Simulator sends information about alarm level(states) of the real sensors to the algorithm

2. Algorithm generates track of the target's motion.

3.2.16.2.3. Alternative Flow of Events

1. Simulator sends information about alarm level(states) of the real sensors to the algorithm

2. If there is no target at the field, algorithm doesn't generate any location information.

3.2.18. Generating track of a simulation

3.2.18.1. Background Information

Algorithm component of the system generates the track of a simulation by using kalman filter. The previous track information and the current location of the target are two necessary data to generate track. Simulator component sends current the states of the software sensors to algorithm. Then, algorithm generates the track by using kalman filter.

3.2.18.2. Stimulus/Response Sequences

3.2.18.2.1. Description

Primary Actor	The algorithm component of the system
Goal In Context	To generate track of the target from previous locations and the currrent state of the software sensors.
Trigger	Algorithm component gets the state information of the sensors.

3.2.18.2.2. Normal Flow of Events

1. Simulator sends information about alarm level(states) of the software sensors to the algorithm

2. Algorithm generates track of the target's motion.

3.2.18.2.3. Alternative Flow of Events

1. Simulator sends information about alarm level(states) of the software sensors to the algorithm

2. The simulated target gets out of bounds of the field, algortihm doesn't generate any location information.

3.2.19. Getting generated location

3.2.19.1. Background Information

The simulator component gets the states of the software sensors from algorithm component. Simulator component learns the states of the sensors from algorithm component and holds this information at its memory.

3.2.19.2. Stimulus/Response Sequences

3.2.19.2.1. Description

Primary Actor	The simulator component of the system
Goal In Context	To get the generated location of the target from algorithm.
Trigger	Simulator component gets a signal from its internal clock.

3.2.19.2.2. Normal Flow of Events

1. Algorithm generates the location of the target.

2. A clock signal comes.

3. Simulator receives the new location of the target from algorithm components.

3.2.19.2.3. Alternative Flow of Events

1. Simulator sends information about alarm level(states) of the real sensors to the algorithm

2. The target gets out of bounds of the field, algorithm doesn't generate any location information.

3.2.20. Getting generated track

3.2.20.1. Background Information

The simulator component sends the states of the real/software sensors to algorithm component. Algorithm stores the previous and current location of the target. So, it generates the track of the target by using kalman filter. It stores track information at its own memory.

3.2.20.2. Stimulus/Response Sequences

3.2.20.2.1. Description

Primary Actor	The simulator component of the system
Goal In Context	To get the generated track of the target's motion.
Trigger	Simulator component gets a signal from its internal clock.

3.2.20.2.2. Normal Flow of Events

1. Algorithm generates the track of the target's motion.

2. A clock signal comes.

3. Simulator receives the track information from algorithm components.

3.2.20.2.3. Alternative Flow of Events

1. Simulator sends information about alarm level(states) of the real sensors to the algorithm

2. The target gets out of bounds of the field, algorithm doesn't generate any location information.

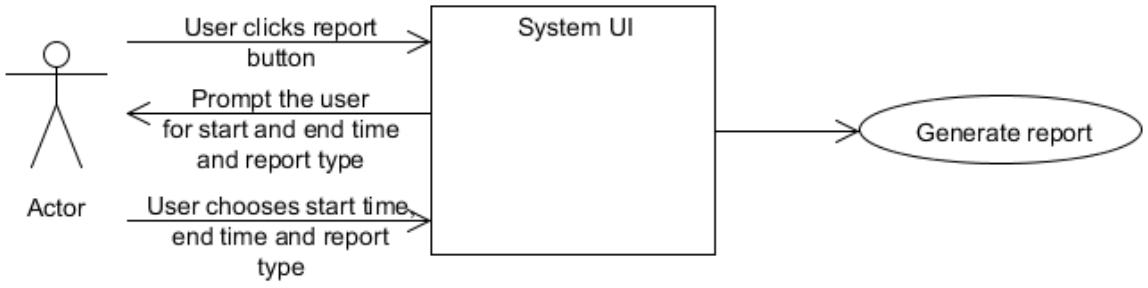
3.2.21. Report Generating

3.2.21.1. Background Information

This feature enables the user to view all relevant information of a session. The user shall select a start and end time for the session he/she wants to generate a report for. The use shall click at the Generate Report button from the menu and shall be prompted with a form asking for specific information about the report to be generated. The user shall provide the start and end time of the time interval he/she wants to generate the report for, and the format of the document. A document containing all of the relevant information such as map,sensor locations,alarm logs, plotted real and estimated tracks retrieved by the database will be generated.

3.2.21.2. Stimulus/Response Sequences

3.2.21.2.1. Diagram



3.2.21.2.2. Description

Primary Actor	User of the system
Goal In Context	To generate a report about a session.
Trigger	User clicks Generate Report button on the menu

3.2.21.2.3. Normal Flow of Events

1. User clicks generate report button on the menu.

2. User is prompted to choose a start and end time and the desired format.

3. The relevant information is retrieved by the database and shown in the desired format of the document.

4. User is prompted to save the report.

3.2.21.2.4. Alternative Event Flows

Alternative Event Flow 1

4.User discards the report.

3.2.21.3. Functional Requirements

1. The system shall check the validity of the time interval the user has selected to view a report for.

3.2.22. Managing Input

3.2.22.1. Background Information

The system shall listen inputs from a joystick device or another joystick-like device when received a notification from core module of the system and convert this raw inputs into human understandable location data form for User Interface.

3.2.22.2. Stimulus/Response Sequences

3.2.9.2.1. Description

Primary Actor	User input interface of the system
Goal In Context	To convert user input into understandable form
Trigger	The user interface sends listen request

3.2.22.2.2. Normal Flow of Events

1. The user interface sends listen request.

2. The system starts listening from input device.

3. The system converts the received data into understandable location data form.

3.2.22.2.3. Alternative Flow of Events

1. The user interface sends stop request.

2. The system stops listening from input device.

3.2.22.3. Functional Requirements

1. The system shall check whether a device is connected to system and this device is configured properly. If not, the system shall give a warning to user interface.

2. The system shall wait for listen requests from user interface of the system.

3. Once the system starts listening to input device, the system shall wait for stop requests from user interface of the system.

3.2.23. Logging

3.2.23.1. Background Information

As a necessity of aspect-oriented programming, the system shall log every important action occured in the event flow. Logging shall be done in two types: Error logging and event logging. All logs created by this function shall include exact date and time of event, type of log (error or event), the caller of this function and detailed definition of the event.

3.2.23.2. Stimulus/Response Sequences

3.2.23.2.1. Description

Primary Actor	The modules of the system
Goal In Context	To log an error or event
Trigger	A module of the system sends log request

3.2.23.2.2. Normal Flow of Events

1. A module of system sends event log request with necessary information.

2. The system logs event information into event log files.

3.2.23.2.3. Alternative Flow of Events

1. A module of system sends error log request with neccessary informations.

2. The system logs event information into error log files.

3.2.23.3. Functional Requirements

1. The system shall wait log requests from other modules of system and handle them properly.

3.2.24. Managing Subsystem Communication

3.2.24.1. Background Information

Since the system is composed of many subsystems, the system shall be able to manage communication between these subsystems. Basicaly the aim of this function is to redirect requests and messages to target modules. Also if necessary the system shall write related information to database or log necessary event information.

3.2.24.2. Stimulus/Response Sequences

3.2.24.2.1. Description

Primary Actor	The modules of the system
Goal In Context	To send incoming message to related target
Trigger	A module of the system sends message

3.2.24.2.2. Normal Flow of Events

1. A message come from a module.

2. The system logs sender and target information.

3. The system forwards the message to the target.

3.2.24.3. Functional Requirements

1. The system shall check whether the receiver received the message. If not, the system shall return an error message indicating the problem.

2. The system shall connect to database management system and write related data to database if necessary. If there is an error occurred during this process, the system shall return a warning message.

3.3. Non-Functional Requirements

In this section we will present performance requirements and design constraints of this project.

3.3.1. Performance Requirements

Since real time simulation is the main purpose of this project, to make visualization be fluent, we need a high performance graphics card.

3.3.2. Design Constraints

In this section we will present hardware and software requirements separately.

3.3.2.1. Hardware Requirements

The main hardware requirements of this project is Waspmote Seismic Sensor Cards and o Gateway. Alarm messages delivered from seismic sensor cards to gateway allows the system to gather location information.

3.3.2.2. Software Requirements

The project will be coded in JAVA Programming Language. We will use Eclipse IDE because of the Rich Client Platform support of Eclipse. We will use GIS libraries that is provided by NASA for building map layer.

The database management system that will be used in this project is PostegreSQL. PostgreSQL provides a useful data types that makes storing location information easier.

4. Data Model and Description

This section describes information domain for the target tracking system.

4.1. Data Description

Data objects that will be managed/manipulated by the target tracking software are described in this section. We will mainly have Alarm, Sensor, Location, RealTrack, GeneratedTrack and Session objects, which data descriptions and logical structure are explained below.

4.1.1. Data objects

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	

Sensor Location 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	

4.1.2. 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.

4.1.3. Complete data model

A simple ERD Target Tracking Database is provided below:

4.1.4. Data dictionary

A reference to the data dictionary is provided. The dictionary is maintained in electronic form.

5 . Behavioral Model and Description

In this section we will state the behavioral properties of our software. First we will give a description for software behavior and then we will present state transition diagram for our software.

5.1. Description for software behavior

Our system will require specific sequence of inputs from the users to operate correctly. The inputs will be given by the users of the system with the user interface.

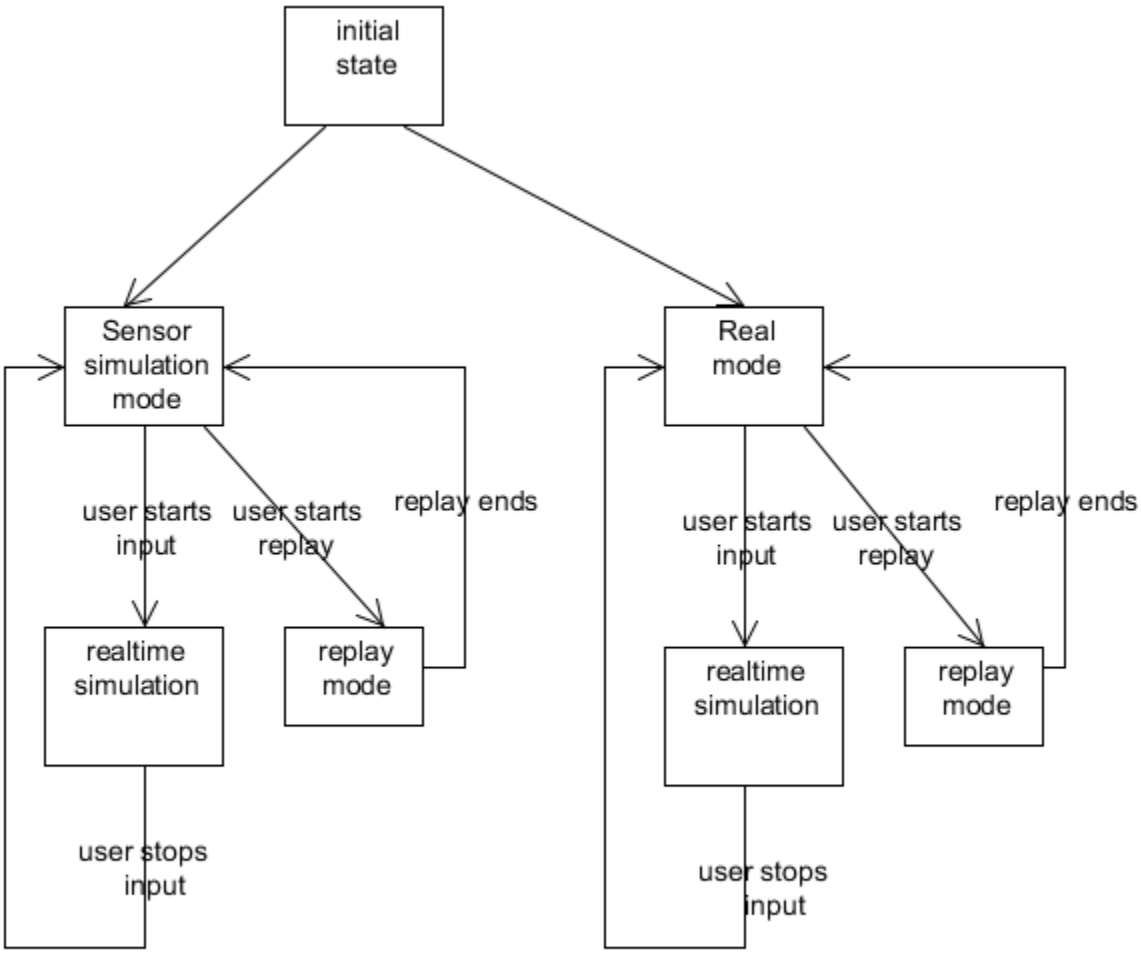
First our software will need user to select the mode of the program. Then depending on the user's choice the program can either switch to sensor simulation mode or switch to real mode.

If the user selects sensor simulation mode, our software will provide user two choices which are 'start simulation' and 'replay previous track'. Again depending on the user's choice our software will start a new simulation or play a previous simulation.

If the user selects real mode, like sensor simulation mode, our software will provide user two choices which are 'start simulation' and 'replay previous track'. Again depending on the user's choice our software will start a new simulation or play a previous simulation.

All states will be returned to their parent state.

5.2 . State Transition Diagrams



6 . Planning

6.1 . Team Structure

We have divided this project into components, every member of our group will implement some parts of project. Members and these components which they are responsible for implementing are below.

- Erdoğan Cem Evin User Interface
- Edona Fasllija Report, Database
- Umut Erdal Core, I/O, Log
- Serkan Yanar Simulation, Communication, Algorithm, Sensors

6.2 . Estimation (Basic Schedule)

For this project we have made a weekly progress plan. We will fallow every step of the plan for one or more weeks. The work plan of the project is described below.

- Intruder Detection Algorithm
- Simulation Environment
- Motion tracking algorithm for one moving object
- Implementation
- Testing
- Testing of the algorithm in simulator
- Testing of the algorithm in a real environment
- Review and Optimization of the algorithm
- Extension of the algorithm
- Extension for multiple moving objects
- Testing of the extended algorithm

6.3 . Process Model

We will take waterfall model as our process model while developing our project.

7 . Conclusion

At the end of the project, the end product will be software that can be used to predict and simulate the path of a moving object by processing the data coming from seismic sensors. If the project is successfully completed, it 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.