# Middle East Technical University

Department of Computer Engineering

Real Time Decision Support System

Software Requirements Specification Document

**Arctic Donkeys**

Zeynep Miray Mazlumoğlu - 1819481

Arda Aslan - 1881010

Göksucan Akın - 1818905

Onur Yılmaz - 1819671

**30.11.2014**

# 1. Introduction

## 1.1. Problem Definition

The problem is to implement a real-time decision support system for earthquakes. It covers various functionalities like infrastructure management (on/off for gas pipes, electricity lines considering the current situation in the city) and unit distribution (optimization of distribution of recovery units among the field, distribution of various supplies efficiently). Data simulation will be based on realistic data. In our scenario, distribution model of damages to buildings / collapse probability considers some realistic parameters like distance to fault line, ground structure, building age. In order to visualize results of the model and optimization, a GIS interface will be provided. Actions and management of operations will be controlled on a map. Recommendation to user will be implemented via different AI and/or optimization algorithms

## 1.2. Purpose

This software requirements specification document is prepared to provide complete elucidation about the software in order to demonstrate the functionality of it and system attributes to clarify their role in the whole frame. This documentation also includes detailed information about system features that will be implemented. While defining interactions between attributes and functions, use case diagrams are applied. Furthermore, performance elements like speed, efficiency, availability, reliability, response time and throughput of the system under certain circumstances will be examined while considering its attributes such as security, maintainability, portability, dependability and testability. ER diagrams are used to show database design. This document is addressed to disaster relief organizations, emergency managers, civil administrators and people who live in seismic areas.

## 1.3. Scope

In Real-time Disasterous Support System(RDSS) there are two type of users which are supporters and clients. Clients can connect to the RDSS so as to get assistance from supporters in real-time to make correct decisions before disasterous situations occur. Supporters provide a decision to the client by analyzing the situational data with the help of information gathered about liability of the given location when disasterous event occurs. Clients have to enter some information such as location of the disaster, etc. in order to get reliable support. RDSS has two sub-systems which are SAP HANA and ArcGIS. RDSS is composed of three main parts. The first part is to gather data for chosen pilot location and put them into our database by using SAP HANA. The second part includes optimizing algorithms to give the best and the fastest decisions to take precautions before earthquakes which is also performed under the SAP HANA sub-system. The last part is based on visualization of the optimized results under ArcGIS software.

## 1.4. Definitions, Acronyms and Abbreviations

| Term | Description |
|---|---|
| **RDSS** | Name of the System |
| **GIS** | Geographic Information System |
| **ArcGIS** | GIS Software to be used |
| **SAP Hana Cloud Database Platform** | The in-memory Platform as a Service |
| **ER** | Entity Relationship Diagram |
| **SRS** | Software Requirement Specification |
| **SDD** | Software Design Description |
| **Java** | Programming Language to be used |
| **ArcCatalog** | Application organizes and manages all GIS Information |
| **Eclipse IDE** | Multi-language integrated development environment |
| **Admin** | Person who makes create, read, update and delete operations to SAP Hana Cloud Platform |
| **Client** | Person who sends query by using system |
| **Scrum** | Iterative and Incremental Agile Software Development Framework for Managing Product Development |

## 1.5. References

The resource listed below is references used in requirement analysis;

i. ArcGIS Desktop. (2010, July 23). Retrieved November 28, 2014, from http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//006m00000069000000.htm

ii. Mann, K. (2012, January 1). ArcUser. Retrieved November 28, 2014, from http://www.esri.com/news/arcuser/1012/a-workflow-for-creating-and-sharing-maps.html

iii. The In-Memory. (n.d.). Retrieved November 28, 2014, from http://hcp.sap.com/index.html

IEEE Standard Documents:

i. IEEE STD 830 - 1998, IEEE Recommended Practice for Software Requirements Specifications

## 1.6. Overview

The following section, Overall Description, portrays the conventional aspects of the system and the requirements of this system without getting definitive about these requirements.

In that section we will analyze Overall Description in six sections:

**Perspective of the System:**_A certain perspective will be gained by considering the relationship of this system with other systems.

**User Properties:**_Required user properties to be able to use the system to its full extent like technical capabilities are described.

**Assumptions and Dependencies:** Conditions that may lead to change in the requirements in a non-functional or functional way are depicted.

**Distribution of the Requirements over the System:** Distribution of requirements between current version and forthcoming adaptations.

In the third chapter namely Specification of Requirements, specifications of the system will be discussed in a very detailed (All specifications needed to implement the system in order to run the system correctly) and professional way. In this section we will analyze Specification of Requirements in seven sections**:**

**External Interfaces:** Represents accurate definitions of the all input-output combinations with definite characterization of interface.

**Functional Requirements:** Central operations starting from the process of inputs until the output is produced will be specified.

**Performance Requirements:** Statistical requirements concerning the performance of the system will be clarified.

**Database Requirements:** Requirements for any coherent information which will be included by database.

**Constraints on the Design:** Restrictions which affect the design of the system will be pointed out.

**Software System Attributes:** The attributes mentioned in section 1.2(Purpose) will be thoroughly analyzed.

**Organization of the Requirements:** For the sake of comprehension the requirements are organized in this subsection.

In the last section (Supporting Information), table of contents, index and appendixes will be provided in order to make the SRS more usable.

# 2. Overall Description

In this section, it will be focused on the conventional aspects of the system and brief introduction to requirements will be made. As it is mentioned in section 1.6 (Overview) the topic will be analyzed in five subsections.

## 2.1 Product Perspective

Real life systems are generally built with several related products and likely, in this system, SAP-HANA and ArcGIS are used as commercial off-the-shelf products. As a whole, it is a single system and it is not a part of any other system. All associated algorithms are going to be developed in SAP-HANA sub-system by the help of Java's HANA libraries. Also an interface will be provided for clients to ease the use of the system. This interface interacts with Eclipse IDE. Eclipse will be connected with both SAP HANA and ArcGIS. There will be data interaction between SAP-HANA and ArcGIS. By sending appropriate queries with ArcGIS to database, optimized outputs can be visualized. Database that is going to be used is SAP-HANA's own in memory database. End users will only deal with user interface. Admins are interacted with SAP-HANA's database.

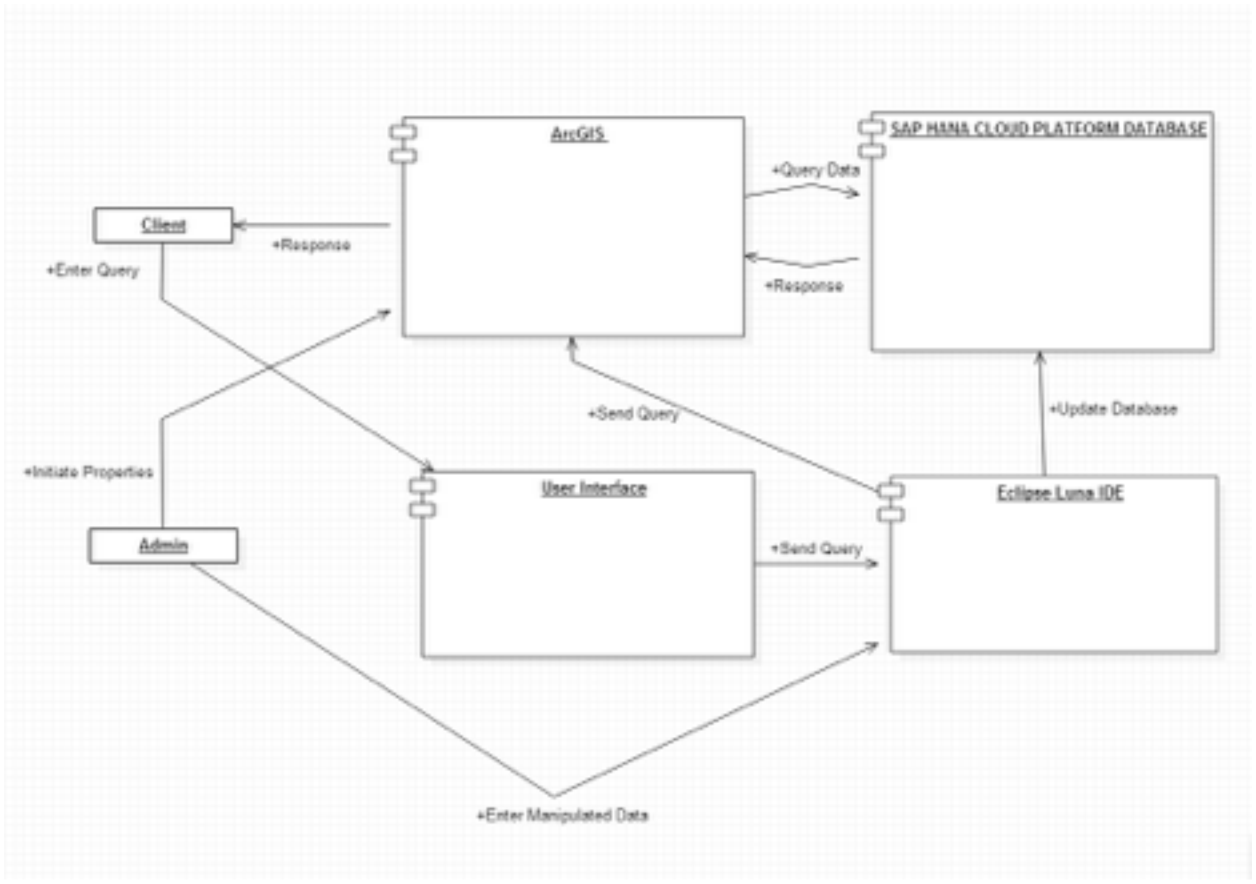


**Figure 2.1 - Block Diagram**

### 2.1.1. System Interfaces

The In-Memory states that ArcCatalog is used to connect ArcGIS to SAP HANA using the HANA odbc interface (which is a layer connecting DBMS and the application)[3]. The paper of ArcGIS Desktop implies that ArcCatalog is an application provides a catalog window that is used to organize and manage various types of geographic information for ArcGIS Desktop[1]. ArcCatalog is capable of organizing GIS contents and managing GIS servers. Other than that, connection between SAP HANA and Eclipse IDE is provided by Java libraries.

### 2.1.2. User Interfaces

There are two user interfaces for RDSS. One is a basic user interface with four text boxes where each of which is associated with its own title (Latitude, Longitude, Magnitude, Focal Depth). For

Latitude text box, valid entry should be between 40.28-41.33. If user entered invalid data, error message will be shown saying “Valid entry should be between 40.28-41.33”. For Longitude text box, valid entry should be between 28.01-29.55. If user entered invalid data, error message will be shown saying “Valid entry should be between 28.01-29.55”.For Magnitude text box, valid entry should be between 4-10. If user entered invalid data, error message will be shown saying “Valid entry should be between 4-10”. For Focal Depth text box, valid entry should be between 0-200. If user entered invalid data, error message will be shown saying “Valid entry should be between 0-200”. It has one button which is for sending data to the system. Other then these, there is a combo box allows users to choose a district. If there is no selected district area , error message will be shown saying “Please select a valid district area”. Interface elements contain texts in English. Other user interface is the default user interface provided by ArcGIS as an output to the user. The related map and relevant settings (Zoom in, zoom out, close, minimize) are provided in that interface.

### 2.1.3. Hardware Interfaces

RDSS should operate on the Microsoft platform which requires Intel and AMD chips. Other than these hardware constraints there is no need for additional hardware.

### 2.1.4. Software Interfaces

SAP HANA is a platform which provides its users multiple functionalities to use. SAP HANA has a cloud platform where a user can deploy his/her application to be executed. Another functionality is that SAP HANA has a database which has both column and row based storage for any data in-memory so that the query results can be much faster . Also, SAP HANA provides a development environment for its users in many languages which results in with a great flexibility. SAP HANA is a low cost product with a higher performance comparing it to its peers. SAP HANA is also an open platform. Therefore , any third-party application can be run on this platform.

Java will be used as the programming language for this project. The last versions of Java Standard Edition 7 was released on July 28, 2011 and Java Enterprise Edition 7 was released on May 28, 2013. Java is very popular programming language and it serves, many opportunities regarding connecting SAP HANA platform. In detail, there are built-in database connection protocols for SAP HANA in Java . This is the most important reason why we choose Java to implement the project. Another reason is Java's performance. Java platform achieves superior performance by adopting a scheme by which the interpreter can run at full speed without needing to check the run-time environment.

Another software product which will be used is ArcGIS. ArcGIS is a geographic information system (GIS) cooperating with maps and geographical data. It is used for; creating and using maps, processing geographical data, analyzing mapped information and managing geographic information in a database. In February 2014, ArcGIS 10.2.2 was released and in this project, desktop version of ArcGIS 10.2.2 is used.

The system will have four software interfaces; one between the SAP-HANA and the ArcGIS, one between Eclipse and SAP-HANA, one between Eclipse and ArcGIS the last one between these components and the operating system. Interfacing with the operating system is outside the scope of the project because softwares handle the interfacing by themselves.

### 2.1.5 Communications Interfaces

RDSS will communicate with SAP HANA using jdbc protocol. Communication between ArcGIS and SAP HANA is held by ArcGIS (Esri and SAP are partners so by default, this feature is enabled). Lastly, communication between ArcGIS and Java will be performed by appropriate Java libraries.

### 2.1.6. Memory Constraints

Since SAP-HANA is cloud based database, clients are connected to a host and process their data. On most SAP HANA hosts, memory size ranges from 256 gigabytes to 2 terabytes. However, clients can connect to these host machines by using their credentials without requiring that much memory on their personal computer. In addition to that, the system does not limit on the secondary memory.

### 2.1.7. Operations

Some basic operations will be examined in section 3.2(Functional Requirements)in a detailed way. This section will describe and summarize the operations of the system. Operations visible to users are the ones executed by the help of user interface. User can send queries to the system. After that, invisible operations take place between the subsystems, database and the operating system.

### 2.1.8. Site Adaptation Requirements

This system is based mostly on data process. Therefore, admins provide initial data to the database in order to make the system operable. In addition, these data shall be realistic since the system will be real time decision support system. There is no specific modification needed to adapt the system to the containing softwares.

## 2.2. Product Functions

In this section, the main functions that are served in subsystems will be given. These functions will be shown with the help of Use-Case diagrams. Later in the section 3.2 (Functional Requirements) functions will be examined in a more detailed manner.

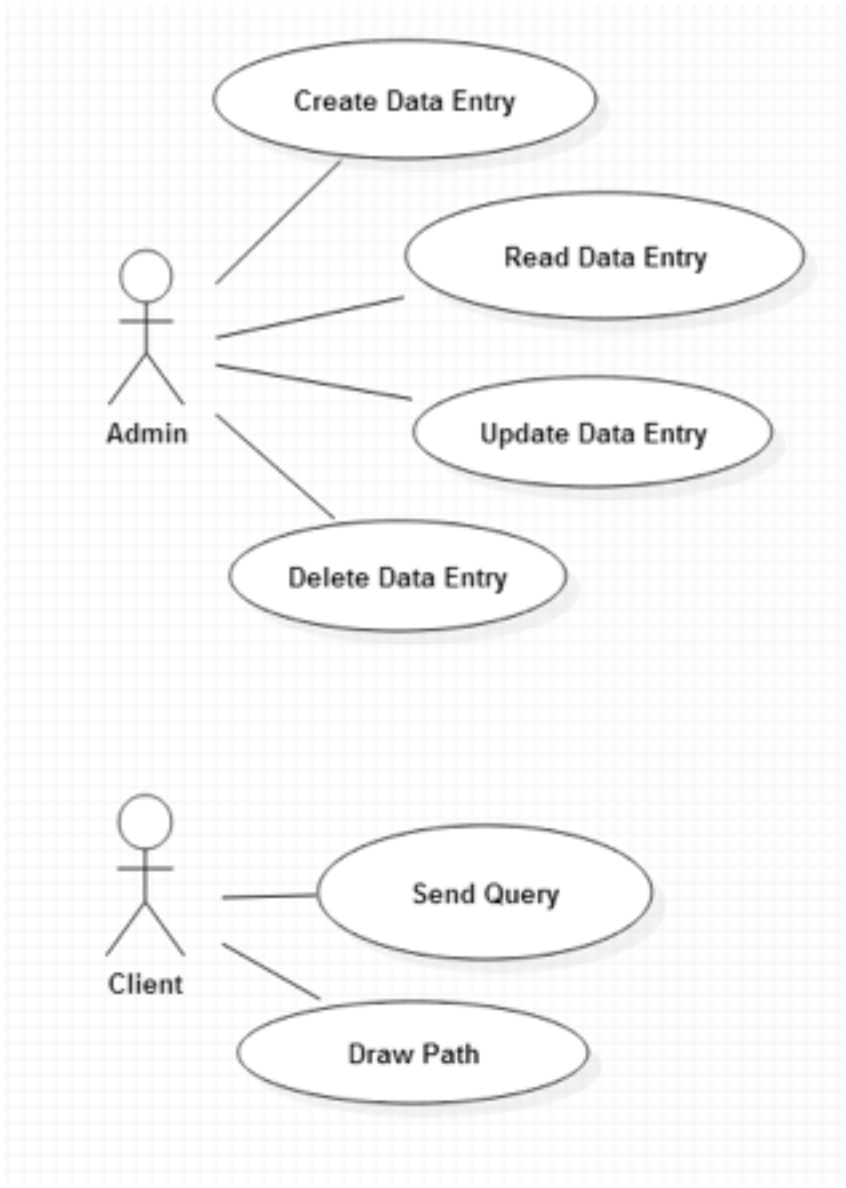


**Figure 2.2.1 Use case diagram for RDSS**

In RDSS, there are two types of users. One of them is Clients. Clients are composed of rescue agents who take advantage of the end product. Admins are the creators of the RDSS software and they are capable of manipulating/creating earthquake and building data. Compared to the Admins, Clients have less privilege. They cannot manipulate any database entity. They can only query data by using existing training data and get model's output.

1) Send Query: Client sends query to the system. This query includes magnitude, latitude, longitude, focal depth and district information. System will try to find predicted outcome by comparing input end existing models.

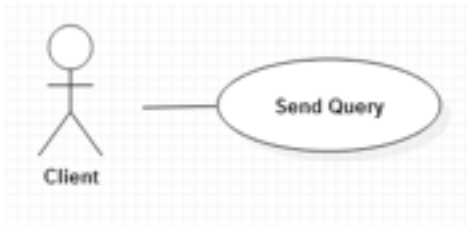


**Figure 2.2.2 Send Query**

| | |
|---|---|
| Use Case ID | RDSS_UC1 |
| Use Case Name | Send_Query |
| Description | Send_Query use case defines the user action that user can fill the criteria and send it to the system to get model on a map |
| Actors | Client |
| Preconditions | ■ Client should open the RDSS to make a query.<br>■ All empty text boxes should be filled<br>■ District area should be selected from dropdown list<br>■ All searching criteria should be filled with valid inputs which are discussed in section 2.1.2. |
| Trigger | This use case is activated when user fills all criteria and clicks "Send" button. |
| Basic Flow | Step 1: Client opens RDSS system<br>Step 2: Client fills all query criteria by considering valid input ranges.<br>Step 3: Client presses "Send" button.<br>Step 4: RDSS models the given query using training data.<br>Step 5: Result model is illustrated via ArcGIS interface. |

| | |
|---|---|
| Alternate Flow | There is no alternate flow this use case. |
| Exception Flow | ■ If Client does not select any district and tries to send the query, system gives “Please select a valid district area” error.<br>■ If Client does not fill Latitude box or entered Latitude value is out of range, system gives “Valid entry should be between 40.28-41.33” message.<br>■ If Client does not fill Longitude box or entered Longitude value is out of range, system gives “Valid entry should be between 28.01-29.55” message.<br>■ If Client does not fill Magnitude box or entered value is not valid, system gives “Valid entry should be between 4-10” message.<br>■ If Client does not fill Focal Depth box or entered value is not valid, system gives “Valid entry should be between 0-200” message.<br>■ If multiple areas are not filled and/or filled with invalid data error message is given regarding upper left located entry. |

2) Draw Path: If a user wants to draw a path to the zone s/he wants, s/he shall click to the appropriate zone on the output map provided by ArcGIS. Then, the shortest path from the nearest rescue agent to that zone will be drawn.

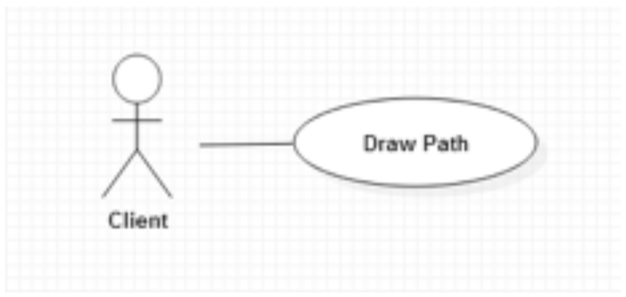


**Figure 2.2.3 Draw Path**

| | |
|---|---|
| Use Case ID | RDSS_UC2 |
| Use Case Name | Draw_Path |
| Description | Draw_Path use case defines the user action that user can select a special area on map to see shortest path to the that destination from the nearerst rescue agent. |
| Actors | Client |
| Preconditions | In order to do action on created map, Client should send query to the RDSS to create a map. |
| Trigger | Mouse clicking on map |
| Basic Flow | Step 1: Client selects a desired area on map by mouse click.<br>Step 2: RDSS handles this request and uses Graph Search Algorithm to draw path from the nearest rescue agent to the desired location<br>Step 3: Result path is visualised through ArcGIS |
| Alternate Flow | There is no alternate flow for this action |
| Exception Flow | Since map is not created from the RDSS unless a valid query search performed, there is no exceptional flow. |

3)Create Data: Admin creates data for whole optimization and visualisation process. Main algorithm uses created entries to produce a trained model.

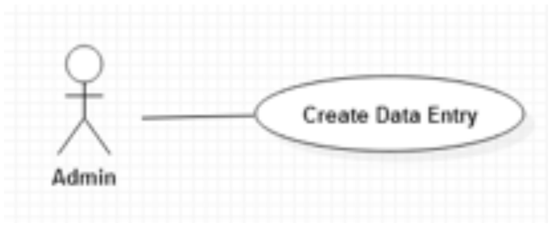


**Figure 2.2.4 Create Data Entry**

| | |
|---|---|
| Use Case ID | RDSS_UC3 |
| Use Case Name | Create_Data_Entry |
| Description | Create_Data_Entry use case defines the user action that user can fill the entities and add it to the system to create data for training action. |
| Actors | Admin |
| Preconditions | Connecting the database shall be established |
| Trigger | Since these actions are handled by java functions automatically, there is no special trigger button for it. |
| Basic Flow | Step 1: Admin connects database<br>Step 2: Admin fills entries to add them to the database<br>Step 3: By function call, data are added to database. |
| Alternate Flow | There is no alternate flow. |
| Exception Flow | If there is any insufficient data entity, Java functions throw exception which is handled by Admin. |

4)Read Data: Admin can fetch data from database to observe and process data.

Admin
Read Data Entry

**Figure 2.2.5 Read Data Entry**

| | |
|---|---|
| Use Case ID | RDSS_UC4 |

| | |
|---|---|
| Use Case Name | Read_Data_Entry |
| Description | Read_Data_Entry use case defines the user action that user can fetch the entities and investigate them for any errors. |
| Actors | Admin |
| Preconditions | Connecting the database shall be established |
| Trigger | Since these actions are handled by java functions automatically, there is no special trigger button for it. |
| Basic Flow | Step 1: Admin connects database<br>Step 2: Admin fetches entries to investigate them |
| Alternate Flow | There is no alternate flow. |
| Exception Flow | If the database does not contain the queried data, Java functions throw exception which is handled by Admin. |

5)Update Data: Admin can update the data in the database.

Update Data Entry
Admin

**Figure 2.2.6 Update Data Entry**

| | |
|---|---|
| Use Case ID | RDSS_UC5 |
| Use Case Name | Update_Data_Entry |
| Description | Update_Data_Entry use case defines the user action that user can change the entities in database |

| | |
|---|---|
| Actors | Admin |
| Preconditions | Connecting the database shall be established |
| Trigger | Since these actions are handled by java functions automatically, there is no special trigger button for it. |
| Basic Flow | Step 1: Admin connects database<br>Step 2: By function call, updates are reflected to the database. |
| Alternate Flow | There is no alternate flow. |
| Exception Flow | If the query to the database is erroneous, Java functions throw exception which is handled by Admin. |

6)Delete Data: Admin can remove a specific data from the database

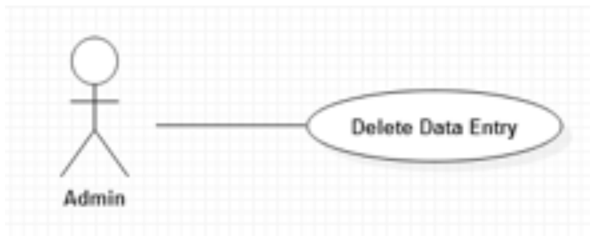


**Figure 2.2.7 Delete Data Entry**

| | |
|---|---|
| Use Case ID | RDSS_UC6 |
| Use Case Name | Delete_Data_Entry |
| Description | Delete_Data_Entry use case defines the user action that user can remove the entities for changes in previous data or renewed data. |
| Actors | Admin |
| Preconditions | Connecting the database shall be established |
| Trigger | Since these actions are handled by java functions automatically, there is no special trigger button for it. |

| | |
|---|---|
| Basic Flow | Step 1: Admin connects database<br>Step 2: Admin specifies entries to detele them<br>Step 3: Changes are reflected through database. |
| Alternate Flow | There is no alternate flow. |
| Exception Flow | If the query to the database is erroneous, Java functions throw exception which is handled by Admin. |

## 2.3. User Characteristics

Clients should know how to read a map and have an ability to use computer in intermediate level. They also need to be familiar with English to be able to use the RDSS. Last of all, they should follow step by step given supporting informations in order to apply them correctly in real-life environment.

## 2.4. Constraints

There is a single major problem concerning developers. The issue is that the system cannot make any optimization and modeling for users without having sufficient data in database. Firstly, data should train the system for further optimizations and modeling. Developers are supposed to add data as training input to  the database and use them to train the system. After then, Clients can query any other data to get possible outcome which is modeled from trained information. Other than these, the most important feature of this system is being a safety critical system. This issue will be held in later sections, in detail. Hardware and software topics are discussed in section 2.1.3 and 2.1.4.

There are some other main constraints on the system concerning the developer.

These constraints are listed below:

- Developer shall design the project compatible with Microsoft Windows.
- The development environment will be Eclipse.
- The development language will be Java and Javascript.
- The database will be SAP HANA’s cloud database
- The system must be a real-time system.
- ArcGIS will be used for visualizing the model.
- The response time of the system must be at most 10 minute.
- The system must provide best decisions according to situation ( Best decisions are based on using all provided information and finding highest rate similarity outcome ).

## 2.5. Assumptions and Dependencies

Dependencies are merged with the section 2.4. Relevant information about dependencies is in section 2.4. While optimizing and modeling the data, some assumptions will be made. Developers will assume the irrelevant environmental values as they are in their optimal value and do not affect the optimization algorithm (Any road will always be usable in case of an earthquake, rescue agents can always reach the desired destination)

## 2.6. Apportioning of Requirements

Below requirements are not urgent or essential for the next versions. However, extra functionalities that may or may not be combined in the future releases.

- Pilot area will be extended.
- Data entities may be extended.
- Other languages will be provided as an option.
- Some other functionalities may be added.
- Further optimization would be needed.

## 3. Specific Requirements

In this section, all of the software requirements specified in details so that designers can design the system and testers can test to the system to satisfy these requirements.

### 3.1. Interface Requirements

This section will explain the properties of the user interface of the RDSS. There is a simple User Interface to ease Clients usage of the system ( Since ArcGIS is relatively complicated software, working with ArcGIS may be challenging for middle level client ). User will interact with this interface ,which is shown in Figure 3.1.1, only by sending queries. Output will be visualised using ArcGIS interface. Sample ArcGIS interface can be shown in Figure 3.1.2. This will be used only for visualizing the heat map of modeled data. It has a single settings button which contains resizing window, minimizing and closing window options. On the other hand, UI of the RDSS has a button in order to initiate the sending process of the given query by the client. The client must fill in the both dropdown and 4 empty boxes so that one can receive  a correct result from the RDSS.

In dropdown box, user will determine the district of the pilot area. Then the client will enter the required attributes which are latitude, longitude, magnitude and focal depth. After filling the input requirements, client shall use the send query button and the result shall be visualized for client. Further information about range ,precision and unity of the values in the boxes and the error cases and their's messages are given in section 2.1.2.

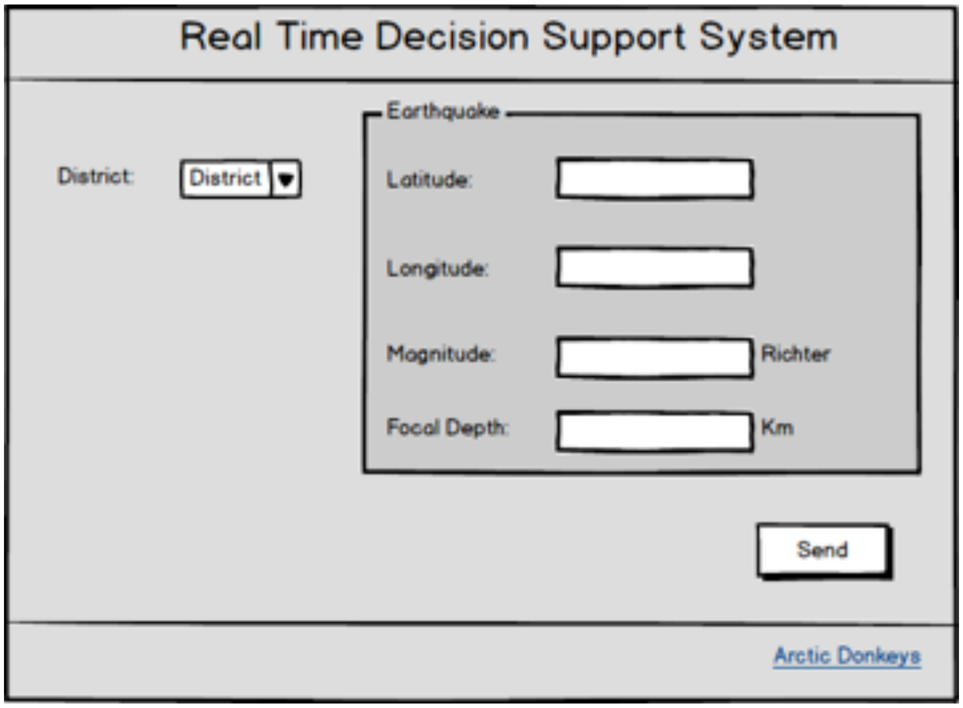
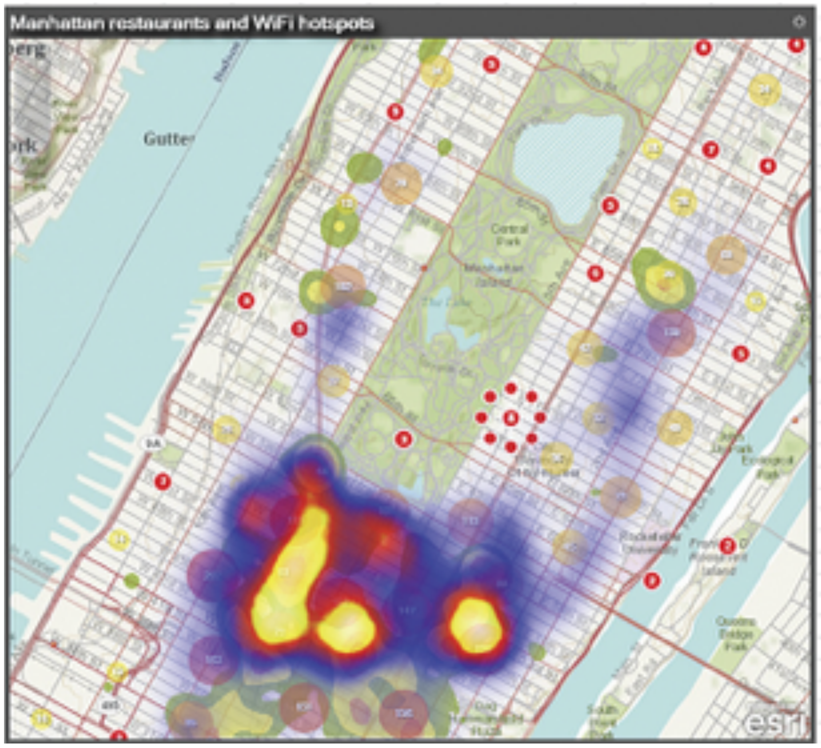


**Figure 3.1.1 - Client Interface**



Retrieved from http://www.esri.com/news/arcuser/1012/a-workflow-for-creating-and-sharing-maps.html

**Figure 3.1.2 User Interface of ArcGIS**

## 3.2. Functional Requirements

This section clarifies major software functions and data flow among the participants such as client and admin of the system. The client, in another saying the end-user, will be the one who sends queries to the system and retrieves results of the sent query. Admin will be the one who creates data like former earthquakes, latest buildings, grounds, rescue agents and fault lines information. Moreover, admin of the system deletes and updates and reads data from the database. Operations are performed by admins' optimization codes.

### 3.2.1 Admin Use Cases

#### 3.2.1.1 Use Case : Create Data Entry

3.2.1.1.1. Functional Requirement 1.1

This function shall enable Admin to add new entry to the database in order to extend the content of the database of the RDSS. This operation is also crucial for the enabling Client to use system ( If there is no training data, there will be no realistic outcome ).

#### 3.2.1.2 Use Case : Delete Data Entry

3.2.1.2.1. Functional Requirement 2.1.

This function shall enable Admin to delete old or wrongly entered data from the the database in order provide more reliable data for the RDSS to use and analyze.

#### 3.2.1.3 Use Case : Read Data Entry

3.2.1.3.1. Functional Requirement 3.1.

This function shall enable Admin to read the current data on the RDSS database so that one can also validate the entered data or use them when needed.

#### 3.2.1.4 Use Case : Update Data Entry

3.2.1.4.1. Functional Requirement 4.1.

This function shall enable Admin to update the current data for changes that could happen in the future so that the database of RDSS remains up to date.

### 3.2.2. Client Use Cases

#### 3.2.2.1. Use Case: Send Query

3.2.2.1.1. Functional Requirement 1.1

Client shall choose district from the drop-down list.

3.2.2.1.2. Functional Requirement 1.2

Client shall enter the latitude of the earthquake to provided text box. By the help of this information system will calculate the heart of earthquake.

3.2.2.1.3. Functional Requirement 1.3

Client shall enter the longitude of the earthquake to provided text box. By the help of this information system will calculate the heart of earthquake.

3.2.2.1.4. Functional Requirement 1.4

Client shall enter the focal depth of the earthquake to provided text box.

3.2.2.1.5. Functional Requirement 1.5

Client shall enter the magnitude of the earthquake to provided text box.

3.2.2.1.6. Functional Requirement 1.6

RDSS shall let client to enter latitude value between 40.28 and 41.33.

3.2.2.1.7. Functional Requirement 1.7

RDSS shall let client to enter longitude value between 28.01 and 29.55.

3.2.2.1.8. Functional Requirement 1.8

RDSS shall let client to enter latitude and longitude values in 2 floating point precision.

3.2.2.1.9. Functional Requirement 1.9

RDSS shall let client to enter magnitude value in Richter unit.

3.2.2.1.10. Functional Requirement 1.10

RDSS shall let client to enter magnitude value between 4 and 10.

3.2.2.1.11. Functional Requirement 1.11

RDSS shall let client to enter magnitude value in 2 floating point precision.

3.2.2.1.12. Functional Requirement 1.12

RDSS shall let client to enter focal depth value between 0 and 200.

3.2.2.1.13. Functional Requirement 1.13

RDSS shall let client to enter focal depth value in kilometres.

3.2.2.1.14. Functional Requirement 1.14

RDSS shall let client to enter focal depth value in 2 floating point precision.

3.2.2.1.15. Functional Requirement 1.15

In order to get result from RDSS, client shall fill all provided boxes and select district from provided drop-down list.

3.2.2.1.16. Functional Requirement 1.16

After pressing “send” button, all modeled views and query results shall be visualized on map by using ArcGIS environment and it should be returned to the client as an output.

3.2.2.1.17. Functional Requirement 1.17

ArcGIS return map shall satisfy all searching criteria.

3.2.2.1.18. Functional Requirement 1.18

According to search criteria, result map shall show the optimal rescue zone positions.

3.2.2.1.19. Functional Requirement 1.19

According to search criteria, result map shall show the optimal path from nearest rescue agent to prioritized rescue area.

3.2.2.1.20. Functional Requirement 1.20

According to search criteria, result map shall show the regions by using heat map with respect to the risk potentials.

3.2.2.1.21. Functional Requirement 1.21

User interface of RDSS shall be reusable. RDSS shall let client to send multiple queries and analyze output map respectively.

3.2.2.1.22. Functional Requirement 1.22

Client shall choose specific area in the district by mouse clicking to that region on the map.

3.2.2.1.23. Functional Requirement 1.23

RDSS shall draw a shortest path from the nearest rescue agent to clicked region.

## 3.3. Non-Functional Requirements

### 3.3.1. Performance Requirements

Performance of making real time decision support system is very important issue because system is aimed to be real-timed. In other words, the system should have dependently high execution rate. Besides that, since the RDSS is a desktop application, any amount of users can use this system as long as  they have the software on their personal computers. Other performance requirements are listed below:

- The estimated time of learning RDSS is 30 minutes.
- Response time of RDSS will be at worst 10 minutes.
- RDSS will try to find the best fitting decision for supporting the client.

### 3.3.2. Design Constraints

#### Reliability

RDSS should successfully create data whose components were specified by admin before. Also, RDSS should successfully delete and update data which is already added by admin. RDSS should give an elucidator message to users if there is a component that is not response at that time, and Mean-time-to-failure(MTTF) shall be min 20 hours. Moreover, RDSS should display the desired map with user’s criteria and should allow mouse clicks on desired map.

#### Security

Since our dataset contain critical information about buildings and fault lines such as age of building, how long the fault line is active, database should not be updated by the user. Furthermore, database and its data should be closed to some dangerous attacks. In other words, database should not be broken in any case.

#### Usability

The scope of the our system will be emergency managers and civil administrators. Hence disaster relief organizations, emergency managers, civil administrators can use it.

**Portability**

As it is discussed in previous sections, software will only be usable in Windows environment. User shall install the software to their personal computer in order to use it. Java will be used as a programming language because of the platform independency property of
it.

**Availability**

The system should be accessible 24 hours of a day, 7 days a week, without any undesired breakdowns. Since our system is going to be exposed to heavy demand in emergency case, there should not be any undesired breakdowns. In case of breakdowns, system should alert to the user. If such situation occurs, system should recover data to maintain availability.

**Maintainability**

Since newer versions of the system will include new features, system should be maintainable. To make this happen, documentations of the system should be easy to understand and well designed. Management of requirement and change should be followed in development phase.Besides, if there is need to configure the system, it should be done via under version control system to provide traceability.  Moreover, an object oriented programming language will be used to make the coding process maintainable.

## 4. Data Model and Description

This section describes information domain for the software.

### 4.1. Data Description

Data objects that will managed/manipulated by the software are described in this section.

### 4.1.1. Data Objects

This subsection of the documents explains system's classes and their relations with each other. System functionalities are represented in Figure **4.1.1.1**

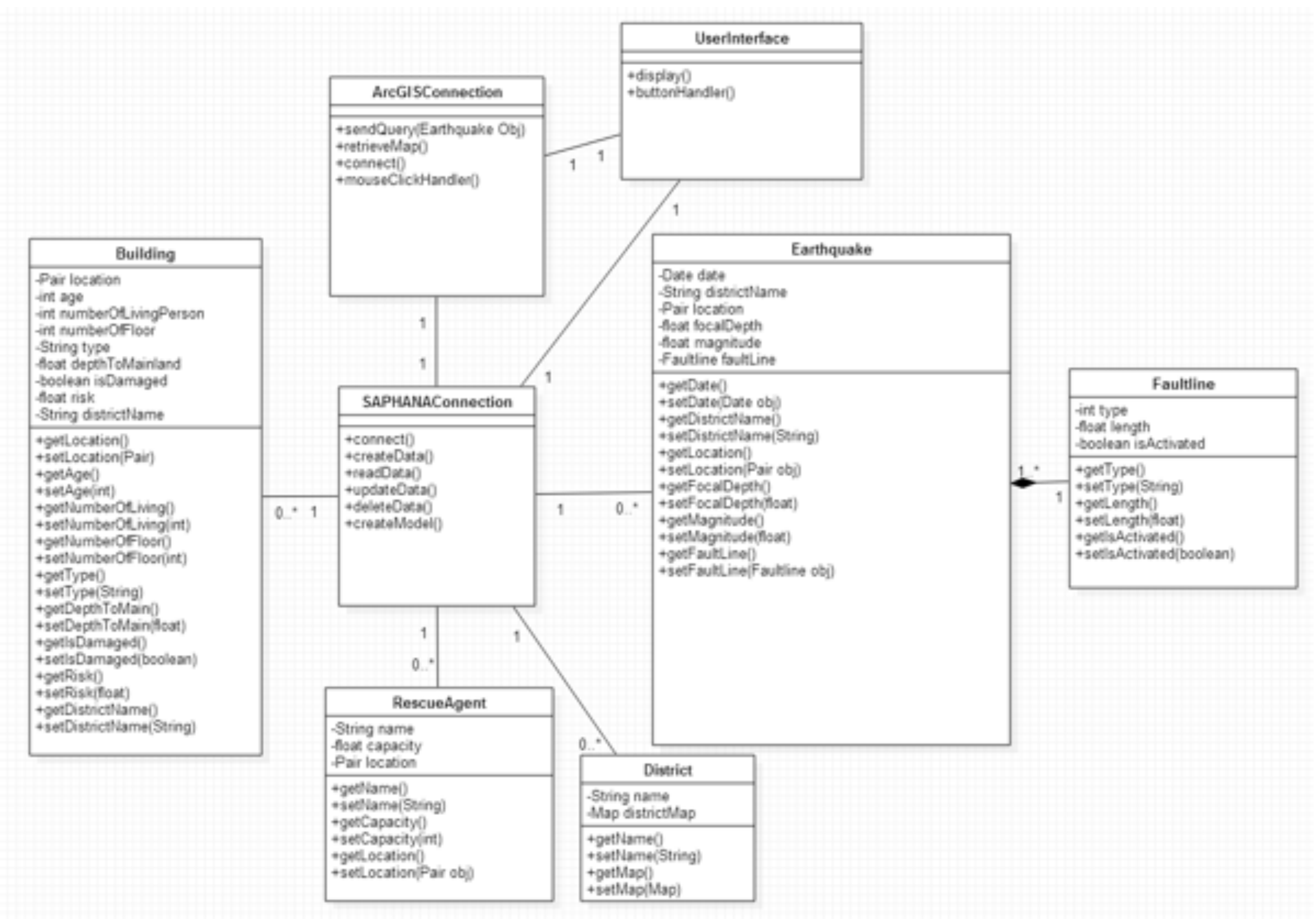


**Figure 4.1.1.1: Class Diagram**

Building: This class represents the building entity and it keeps the information of buildings which are located in selected area. Its attributes are age, location of building, number of living person, number of floor, ground type, depth to mainland, registered district name, whether the building is damaged or not before. In addition, destruction risk of building will be calculated and stored in this object. It keeps location attribute as primary key.

Earthquake: This class represents the earthquake entity and it has attributes location, focal depth, magnitude, date of earthquake and which fault line causes this earthquake. Moreover, current earthquake object holds the district name that client wants to analyze on map. Also, this class has many-to-one aggregation association with fault line class.__

District: This class represents the district entity and it has name and map of district.

Rescue Agent: This class represents the rescue agent entity. Its attributes are location, name and capacity of rescue agent.

Fault Line: This class represents the fault line entity. It has type, length of fault line and whether the fault line is activated or not.

ArcGIS Connection: This class represents the ArcGIS Connection object which binds RDSS and ArcGIS. RDSS retrives desired map from the ArcGIS and handles the mouse click of clients. This object acts as a link between RDSS and SAPHana Cloud Database Platform. Client sends query to SAPHana Cloud Database Platform through ArcGIS. Also, this class has one-to-one association with user interface class.

SAPHana Connection: This class represents the SAPHana Cloud Database Platform Connection object which binds RDSS and SAP Hana Cloud Database Platform. Admin makes create, update, delete and read operations in SAP Hana Cloud Database Platform. Furthermore, admin creates optimization model in SAP Hana Cloud Database Platform. Also, this class has one-to-one associations with ArcGIS Connection and User Interface classes and except fault line class, this class has one-to-zero or more association with all other classes.

User Interface: This class represents the user interface object of RDSS. This object contains “send” query button handler and client displays desired map from user interface of RDSS.

### 4.1.2. Data Dictionary

| Term | Definition |
|---|---|
| location | Location provides latitude and longitude information for building, rescue agent and earthquake classes and it uniquely distinguishes building and rescue agent classes(Primary key). Location provides latitude and longitude information |
| age | It is age of the related building |
| numberOfLivingPerson | It is the number of people living in a particular building |
| numberOfFloor | It is the floor number of a particular building |
| type | It is the type of ground where the building is based on(Rocky, drifting sand, marsh water coast area). Also it denotes the type of a faultine(Primary, secondary). |
| depthToMainland | It is the distance between the ground of the building and mainland |
| isDamaged | It stores the information if the building is damaged in previous disasters. |
| risk | Risk factor will be calculated and heat map will be provided according to this risk factor |

| | |
|---|---|
| districtName | It is the district where the building is located |
| name | It stores the name of the rescue agent and district |
| capacity | It is the person capacity of the related rescue agent. |
| districtMap | It is the map of a district. It is visualized by the help of ArcGIS |
| date | It is the date of a former earthquake |
| districtName | It is the district name of a former earthquake |
| focalDepth | It is the depth of an earthquake |
| magnitude | It is the magnitude of an earthquake |
| length | It is the length of a specific fault line |
| isActivated | It states whether that fault line is active or not |

## 5. Behavioral Model and Description

This section provides the overall behavior of the system. Major events and states are displayed in a state chart diagram.

### 5.1. Description for Software Behavior

The state chart in Figure 5.2.1 illustrates the behavior of the RDSS. In RDSS Admins creates data to train the system and these data are kept in SAP HANA cloud system. After creation procedure admins can also read and update necessary data and delete irrelevant data to give the best result. Then, Admins use these data to train modeling system. All statistical calculations and regression testing mechanisms are performed at that stage. Then, system has a pattern about the earthquake properties-damage relationship. After then, system is ready for creating a model for incoming inputs. This model includes details about possible resultant destruction of the given earthquake with magnitude-latitude-longitude-focal depth information in chosen district. After all these processes, Client is able to use the system by sending queries by filling required data via user interface. This interface is created from Eclipse platform using Java language. By default, ArcGIS-Eclipse and SAP HANA-Eclipse platforms were connected by Admins. ArcGIS and SAP HANA connection is established automatically. Given query is processed by trained data and outcome will give a model which estimates danger-input result. This result is passed ArcGIS through Eclipse as a processable data to be used in mapping procedure. ArcGIS displays the output as a heat map so that Client can simply understand the high risk areas and observe possible paths from the rescue agents to the prioritized locations. After this procedure, Client is also able to choose an area by clicking to see an optimized path considering nearest rescue agent. This mouse click activates reverse data flow from ArcGIS to Eclipse. Graph search algorithm is implemented in Eclipse and it gives the shortest path. This result is visualized through ArcGIS. Unless an action is performed, system preserves its current state.

## 5.2. State Transition Diagrams

After execution of the program, system enters a safe state. In that state, system waits a valid action. Admins have four database operations to trigger the system. These are create, read, update and delete data. After any of these operations is done, system enters Algorithm Modification state. In that state, system can continue to modify database. After all of these modifications, data are ready for the training operations. At that point, system enters Train Modified Data for Modeling state. This state applies statistical operations to whole data. After completion of the training process, Admins get the desired result and this state flow ends. Second alternative from the Safe State is that Client sends a query to the trained system. Client can keep sending queries without waiting for a result. After a successful querying operation, state switch occurs to Modelling Algorithm State. In that state, algorithm is applied using trained dataset to model the result. This result is sent to the related software and displayed on the ArcGIS interface. Unless a new action occurs , system keeps displaying the map. Other than this, there are three more options. One of them is the mouse click state where Client selects a zone from displayed map. If so, system shifts to Path Draw Algorithm state in which Graph Search Algorithm is performed. This algorithm draws shortest path and it is given to the Display Output on Map state. Displayed map is updated. Second option is closing the display map user interface and changes its state to the Send Query state. In that state, Client can send a new or same query by following the procedure. The last option is passing to end state where Client closes the software.

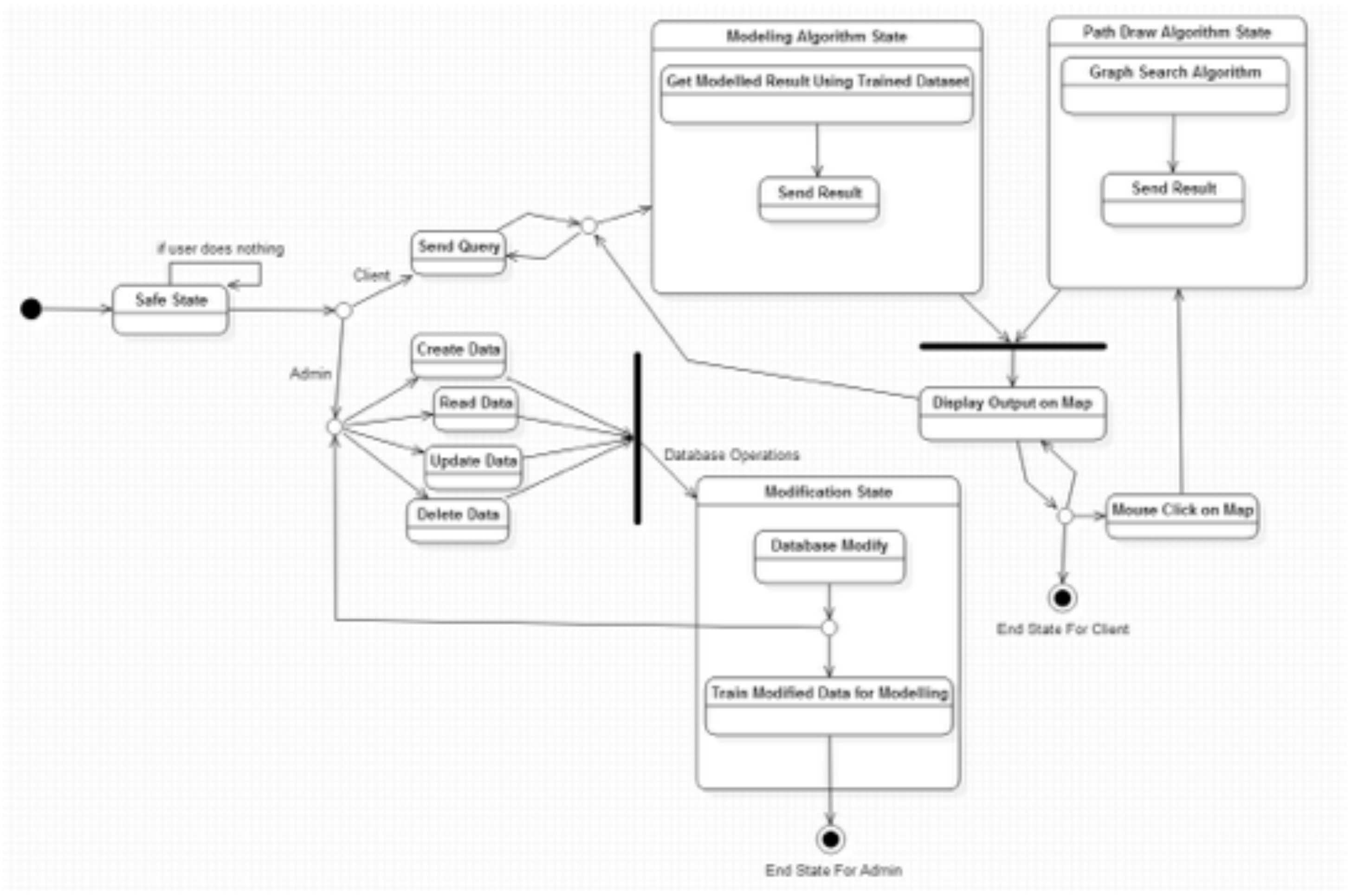


**Figure 5.2.1. State Diagram for RDSS**

# 6. Planning

## 6.1 Team Structure

In the Arctic Donkeys project team there are 4 developers. All members of Arctic Donkeys are dealing with their own target objective. However, every member will work through every development stage in order to establish synchronization between team members.

The project will be consisted of three different system design stages. These stages are as followed:

i. Data Gathering Phase (Phase 1)
ii. Processing and Analyzing Data Phase (Phase 2)
iii. Visualizing Phase (Phase 3)

For phase 1, all of the team members will be working on gathering and storing the data. In Phase 2, again all of the team members will be working on algorithms to process and analyze the gathered data in Phase 1.

In the last Phase, two of the team members will be handling the visualization of the data and the other two members will be handling the integration of the tool used for visualization into the development framework.

## 6.2 Estimation

| Estimation Date | Task |
|---|---|
| 29.09.2014 | Deciding Project |
| 06.10.2014 | Project Idea Proposal |
| 13.10.2014 - 03.11.2014 | Researching Required Systems and Tools |
| 03.11.2014 - 24.11.2014 | Gathering Data |
| 24.11.2014 - 30.11.2014 | Software Requirement Specification (SRS) |
| 01.12.2014 - 07.12.2014 | Trying Software Tools |
| 29.12.2014 - 03.01.2015 | Documenting Design Reports |
| 05.01.2015 - 08.01.2015 | Updated Reports |
| 01.12.2014 - 11.01.2015 | Implementing Base RDSS |

## 6.3. Process Model

We are applying one of agile methodology which is Scrum for our Real Time Decision Support System so that our system can respond quickly to changing requirements without excessive rework. Scrum methodology is based on an iterative approach, each sprint takes approximately three weeks. Each sprint involves sprint planning in which we estimate the requirements of that sprint, implementation, testing and sprint retrospective. In sprint retrospective, we discuss what we did in last sprint and what we are going to do in next sprint. In addition, on each day of a sprint the team holds a daily scrum meeting. Daily scrum meetings enable the team to self organize by close collaboration of all team members. The sprint ends with sprint review and sprint retrospective. Once we will generate the initial version of RDSS, then our system will be developed according to performance results on the basis of a country data. As a result, by using scrum methodology, the team delivers products and clients can change priorities and requirements more quickly.

## 7. Conclusion

This Software Requirement Specification document is prepared to state requirement details of the RDSS project. Firstly, definition of the problem and the general description of the system are given. Then, all the functional, non-functional and interface requirements, data and behavioral models are stated in a detailed manner. Finally, structure of the development team, basic planning and estimation of the development process and the model of the development process are explained. This document will be helpful for constructing a basis for design and development of the system to be developed. Design details of the project will be explained in the Software Design Description document.

## Appendix

No appendix is available

## Index