Marshall Plan Foundation Scholarship Report: A web based visualization technique for avalanche risk mapping in Austria using mobile devices

Helbert Arenas

January 10, 2012

Contents

List of Figures

Chapter 1

What is this all about?

1.1 Introduction

The title of my Ph.D. research is An Agent-Based Simulation Model for the Business Reopening in New Orleans Post Hurricane Katrina. My research basically has three main components: the calibration of the model, the design and implementation of the simulation model, and the design of a visualization component for the model results. Austria is a country with a very different landscape in comparison with Louisiana where my current research is based on. However, the visualization techniques that I use are generic enough to be successfully employed in other spatio–temporal phenomena disregarding the specific location. The phenomena that caught my attention when examining the Austrian landscape are the large number of avalanches during the late fall, winter, and early spring seasons occurring in the Alps. Austria, due to its topographic relief and weather conditions faces this particular thread every year. When avalanches occur, fatalities or injuries are unfortunately not unusual [Turner, 2010] [Reuters, 2009] .The total number of fatalities since 1950 has been more than 1600, which results on an average of approximately 30 fatal victims per year [Holler, 2007].

An avalanche is a rapid flow of snow down a slope. There are mitigation measures designed to reduce the risk of a given area to avalanches. Among these

measures are the construction of deflecting dams, the redevelopment of mountain forests, the development of hazard maps, or the use of explosives in order to prevent larger avalanches by triggering smaller ones [AAA, 2010]. Current research on the avalanche field is focused on the forecasting, avalanche hazard mapping, and avalanche simulation models and information dissemination [Holler, 2007] [Eckerstorfer, 2008]. One of the most common causes of fatalities due to avalanches is the little knowledge that skiers and mountaineers have of avalanche hazards [Eckerstorfer, 2008].

Current technology allows us to provide updated information on the field using Mobile Geographic Information Systems (mobile GIS). Mobile GIS combines in a mobile computing device, Internet and GIS. It has been applied to location based services (LBS) [Shi et al., 2009]. LBS are services that provide relevant information based on the location of the mobile device [Longley et al., 2005].

Mobile devices with access to internet can access maps similar to the ones I have previously developed for my research, making the visualization approach previously described, very attractive. The small screen common in mobile devices, demands special care in the interface design. Until now I had not had this type of requirement in my previous projects. However I believe the techniques I have used could be adapted with minor changes to display dynamic maps, using the position (from the onboard GPS) of the mobile device as part of the spatial query.

Using Mobile GIS it is possible to deliver an updated avalanche risk map to users while they are still outdoors. The purpose of this research will be the design and implementation of the visualization component of a web based avalanche risk map system for mobile devices. The application will contain a map that will show the most updated avalanche risk information for the user position. The map will dynamically represent the changing climatic conditions that affect the avalanche occurrence probabilities.

1.2 Problem Statement

1.3 Research Questions

1. What are the requirements for a web based avalanche risk map visualization module for mobile devices?
2. What are the best tools available for the development of a web based avalanche risk map visualization module?
3. What are the capabilities and limitations of the available tools in the market?

1.4 Methodology

First, through a literature review I identified the requirements for a web based avalanche hazard map for mobile devices. Second, I evaluated the most common tools employed to represent spatio-temporal dynamic phenomena on the internet. Finally, I developed the web based map using the most suitable and available tools fulfilling the requirements identified in the first step.

The result of this research is a set of guidelines for the development of an avalanche risk application for mobile devices. As a second product, I developed a dynamic web based map designed for mobile devices.

1.5 Sections Overview

In Chapter two we talk about avalanches and briefly describe how people get caught on them, indicating the characteristics of a tool that might reduce the risk of avalanche incidents. In Chapter three we describe current technology related to geolocation, webmaps and mobile devices that could be used to deploy an avalanche risk application. In Chapter four we describe the application developed

for this project. Chapter five contains the conclussions, discussion and future research.

Chapter 2

Outdoor winter leisure activities

2.1 Introduction

Outdoor leisure activities involving a certain amount of risk are becoming more popular. Leisure activities considered in the past as "fringe" are becoming more main stream. There are many possible reasons for this growth among others: the media glorification of such activities, technological advancements, better equipment, better access to terrain, and successful marketing campaigns oriented to increase the custom base of these industries . A consequence of this growth is an increase in numbers of people injured or even dead due to accidents in these activities. We have reached the point that there is concern in a segment of society about the costs of risk leisure activities to the society as a whole in the form of financial burdens, personal injuries and emotional impact. There are voices raising questions about the costs of rescue teams or the morals of putting a rescue team in an "unnecessary" risk. Some voices argue that society should regulate these activities or even prohibit some, while others argue that mature rational individuals should have the right to pursue those activities while taking in consideration the consequences of their actions on others [Olivier, 2006].

Back country winter recreation is a group of those outdoor leisure activities with a growing popularity. A negative consequence of this growth is an increase in the number of injuries and fatalities due to avalanches For instance, Figure 2.1 represents the number of fatalities due to avalanches in U.S. from 1955 to 2005. Figure 2.2 shows a more detailed view of the avalanche fatalities. According to the Utah Avalanche Center, most of the victims in United States from 1995 to 2001 were people involved in some back country recreational activity (94%).

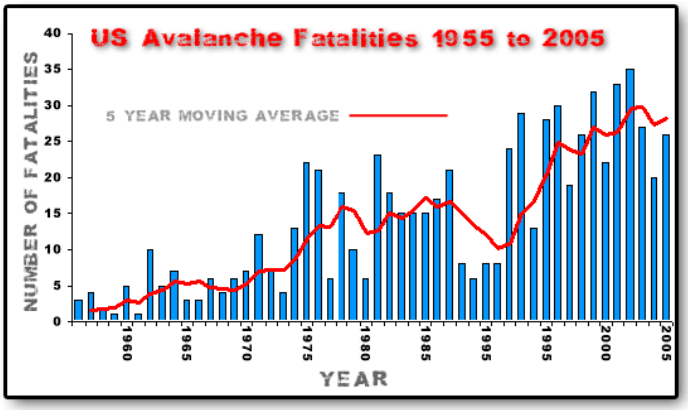

Figure 2.1: U.S. Avalanche fatalities from 1955 to 2005(source: [UAC, 2011])

A well recognized approach to reduce the negative impact of avalanches is education. Previous studies conducted by Furman et al. (2001) indicate human factors such overconfidence and inexperience regarding avalanche are fundamental to understand these type of incidents [Furman et al., 2010].

2.2 Avalanches

An avalanche occurs when a massive amount of snow, ice and rock debris, skid down a mountainside. Most of the recorded avalanches occur in mountainsides with slopes between 30° and 45° [Fonseca et al., 2011]. Slopes lower than 30° do not produce avalanches, and slopes steeper than 50° do not allow enough snow to accumulate to produce one [UAC, 2011] (See Figure 2.3) .

There are two kinds of avalanches :

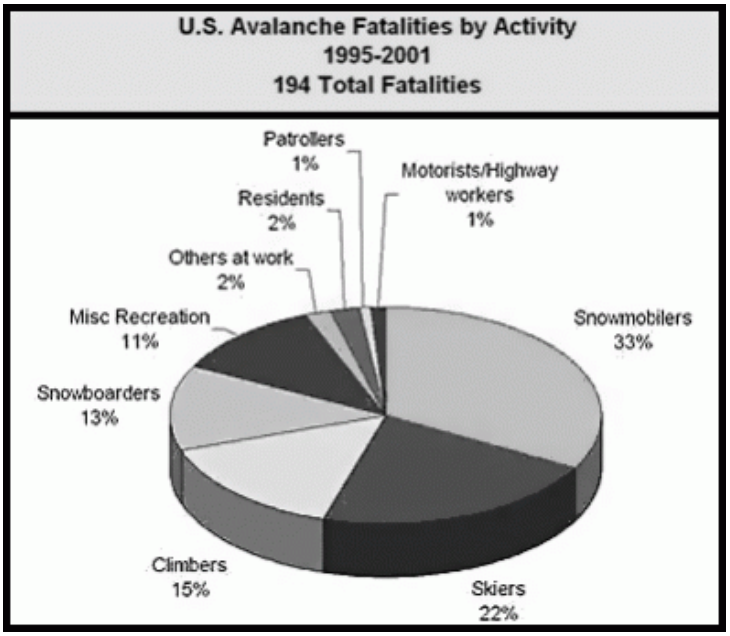

Figure 2.2: Who gets caught in avalanches? (source: [UAC, 2011])

- loose–snow avalanches: In this case the avalanche starts in one point and grows by accumulating loose snow down hill.

- slab avalanche: In this case the avalanche occurs because there is a cohesive snow layer on top of a less cohesive one. The avalanche occurs when the less cohesive layer fractures allowing the upper cohesive layer to slide downhill. In many cases the event is triggered by a sudden extra weight added too quickly, in most of the cases the extra stress is the weight of the victim or a member of the victim's party [UAC, 2011].

According to McCammon (2004) in 93% of the cases the victims themselves triggered the avalanche that caught them. Victims of avalanches are buried in a mixture of snow ice and debris. The cause of death is carbon dioxide poisoning. In the case of a avalanche the first minutes are crucial. Most of the victims (93%) can survive if rescued in the first 15 minutes. However if a victim is rescued after 45 minutes his/her survival chances decrease to between 20% – 30% [McCammon, 2004] (See figure 2.4).

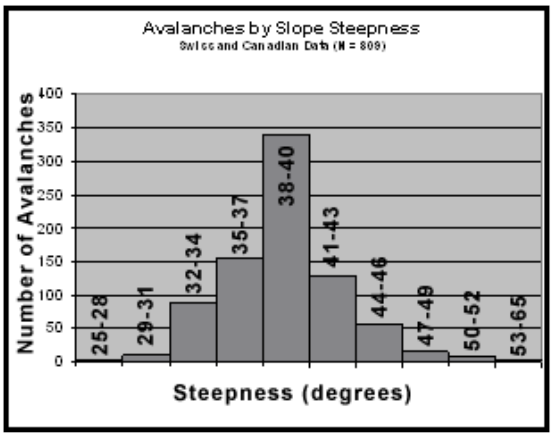

Figure 2.3: Avalanches by slope steepness (source: [UAC, 2011])

2.2.1 Who gets involved in avalanche incidents? and how?

Due to the growth of popularity of winter back country recreation activities nowadays most of the victims of avalanches are people involved in these kind of activities [Brugger et al., 2001]. In the case of United States for the period between 1995 to 2001, 94% of the fatal victims were involved in some kind of winter recreation activity (See Figure 2.2) [UAC, 2011].

Tase (2005) conducted an online survey among recreationists with the goal to identify the segment of the population that is at most risk, and try to identify the causes. The survey comprised 1463 people male and female, with various degrees of knowledge regarding avalanches and from different group ages. The results of the survey indicate that 31% of the sample had a previous experience with an avalanche and 22% were actually hit by one. When divided by gender, the results indicate that 33% of males had been involved in avalanche incidents against 16% of females. The researcher's hypothesis before the survey was that snowmobilers were in more risk than telemarkers, however the results indicated the opposite [Tase, 2005].

There is a significant amount of research on the physical qualities of avalanches, the snow and weather dynamics. However it is not always the case that this knowledge is available for the people that need it most. In many cases poor

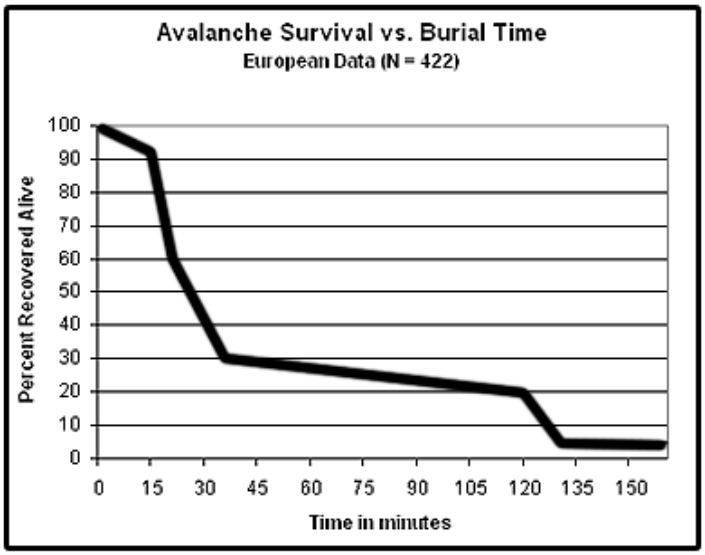

Figure 2.4: Avalanche survival vs. burial time (source: [UAC, 2011])

avalanche education is the main cause of avalanche incidents. Some untrained people might fail to recognize signs that might be obvious to the eyes of an expert, triggering an avalanche that might injure themselves or other parties nearby. In recent years a new approach to the study of these events reveals that the decision making process might be a relevant variable in avalanche incidents [Adams, 2005] [McCammon, 2004].

Adams (2005) conducted a survey among members of the Canadian Avalanche Association. When asked about the cause of the events 97% of the respondents considered that human factors had "great"(48%) or "very great" (39%) influence in avalanche incidents. The majority (67%) of the respondents also considered that education of the recreationists would improve their decision making process. Most of the respondents (81%) also considered that the information provided in the form of bulletins could be improved by increasing its frequency and detail, for example by creating local bulletins instead of regional ones.Although it would require some level of sophistication from the user, most of the surveyed experts (74%) considered that the identification of hazardous areas on maps would ease the decision making process for skiers.

A solid avalanche education would definitely improve the decision making in the field. However in many cases even good trained skiers get involved in avalanche incidents. McCammon (2004) studied 715 avalanche incidents in United States from 1972 to 2003. The results indicate that only 34% of the incidents involved parties with no training at all, 24% of the cases involved parties with a general awareness of the avalanche hazard, in 28% of the cases the parties had basic training and in 15% of the case they had advance formal training. In most of the cases the victims with some degree of training seemed to ignore obvious signs or risk [McCammon, 2004].

McCammon (2004) suggests that human psychology mechanisms that we apply to daily life without serious consequences when wrongfully applied to avalanche situations might lead to catastrophic results. He called these mechanisms "heuristic traps" . McCammon identified six heuristic traps that might lead a skier into peril [McCammon, 2004]:

Familiarity Many skiers decide their actions based on past events. However familiarity becomes a heuristic trap when skiers avoid a rigorous risk evaluation, by disregarding new information and believing that past experiences without incidents assure safety.

Consistency Consistency is the though mechanism that makes people stick to their original decisions. However in back country skiing climatic events might affect the field, making it riskier than at the moment when the original plan was decided. This though mechanism becomes an heuristic trap when a party that has decided a course of action refuses to reevaluate their decisions based on new evidence.

Acceptance Is the propensity to perform acts that a person think would improve their image in the eyes of people she/he respects or likes. In the case of adolescent and young men this might mean to get involved in risky activities that would make them get noticed to women. According to McCammon (2004) mixed parties (men and women) are in more risk that only women or only men parties.

The expert halo It has been noted that in many ski parties there is a member with an informal leadership role. This person could have attained that role by being the one with more expertise on skiing, or being the older or the more assertive. However this behaviour becomes an heuristic trap when the rest of the members of the group cease to evaluate the risk by themselves and totally rely on skills that the leader might or might not have.

Social facilitation In this case the presence of other people might affect the risks a person is willing to take. In this heuristic trap the presence of other people increases the perception of safety.

Scarcity Is the behaviour mechanism that lead people to take unnecessary risks when trying to secure resources or opportunities in the face of potential competitors. For instance when skiers try to be the first ones to access an untracked slope after a storm.

2.3 Summary

A consequence of the growth of popularity of winter back country leisure activities in increase of avalanche incidents involving recreationists. Education is an important tool to increase the safety of people enjoying winter outdoor activities, however with the increasing numbers of people interested in these activities, it is difficult to provide all of them with adequate instruction. Even more, according to McCammon (2004) even expert skiers that might be capable of recognizing alert signals on the field, in certain circumstances might get into trouble by falling into what he calls "heuristic traps" [McCammon, 2004].

A survey conducted by Adams(2005) among experts in the field indicate that most of them considered that in order to reduce the risk of avalanche incidents a good tool would be more detailed and frequent bulletins that would include risk maps, that would facilitate the decision making process of the skiers [Adams, 2005]

However as Adams (2005) points out in order to use an avalanche risk map the user requires a certain degree of sophistication. The tool required to increase the

safety of skiers must consider both the experts and non experts users. It should use maps to indicate the risk areas, but these maps should be easy to use, even by non expert map users.

Chapter 3

Technology Overview

3.1 Introduction

In this chapter we are going to review the current state of technology in the fields of the location of mobile users, the use of dynamic/custom–made maps, location based services and telecomunication devices that users might use while being involved in winter leisure outdoor activities.

3.2 GeoLocation

As mankind began exploring the world a couple of fundamental questions surged where am I? and where is the place I want to go to?. The answer to the first question is called Geolocation, the location of the navigator in real world terms. Navigators in order to identify their location in the world have developed tools with an increasing degree of accuracy along time, such as cross-staffs, astrolabes, quadrants, chronometers and sextants. Current technology allows the geolocation using different methods the most common ones are [Holdener III, 2011]:

- Global Navigation Satellite Systems
- IP Address

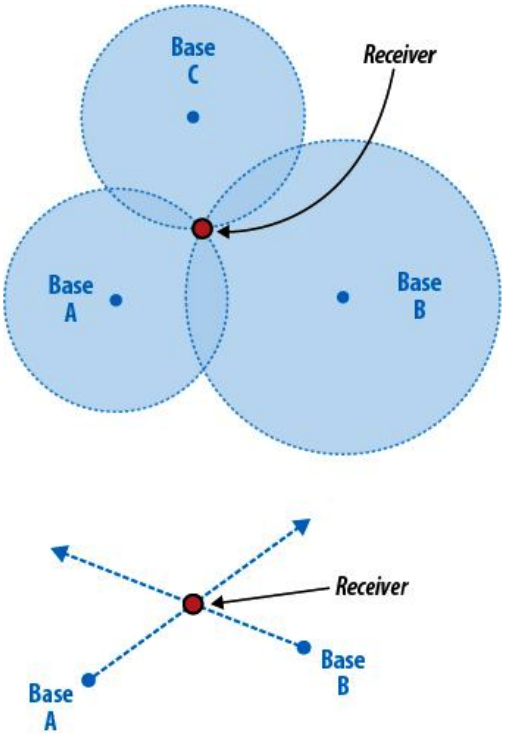

Figure 3.1: Triangulation, radial and directional (source: [Holdener III, 2011])

- GSM/CDMA Cell ID
- WiFi and Bluetooth MAC Address

3.2.1 Global Navigation Satellite Systems (GNSS):

This is one of the most widely used method nowadays. It has its origins in the early 20th. century when radio transmissions began to be used as guiding systems using a technique called Direction finding (DF). When two or more receivers are combined the location of the transmitter can be determined in a process known as triangulation (See Figure 3.1).

With the advent of space exploration new tools and techniques became available. In 1957 the Soviet Union launched Sputnik, an artificial satellite that orbited the earth broadcasting a radio signal. The analysis of Sputnik's signal allowed American scientists to discovered that they could locate the satellite position using

its Doppler effect. This discovery lead to more research on the use of satellites as global navigation systems with projects such as Transit, Timation, Project 621B and SECOR. The increasing knowledge gained from these projects eventually lead to the development of Navstar in 1973 by the American Government, which is the basis of the GPS system widely used nowadays. However at the beginning GPS and the technology related to it was restricted for military use [Pace et al., 1995] [Holdener III, 2011].

The military status of GPS technology changed in 1983 when the Korean Air Lines flight 007 was shot down by the Soviet Union. The civilian airplane lost course and entered into Soviet airspace , the Soviet Union air defense misidentified it by a spy plane and shot it down. After this incident the President Ronald Reagan ordered the U.S. military to allow civilian use of the Global Positioning System in order to avoid future tragedies like the Flight 007. After the release of GPS for civilian use the U.S. military enforced what is known as Selective Availability (SA) a procedure which downgraded the signals in order to reduce the precision of the geolocation for non military use. In the year 2000 President Bill Clinton ordered the Selective Availability to be turned off. Without Selective Availability the precision of the GPS geolocation went from 100 to 20 meters [Holdener III, 2011].

The standard configuration of GPS includes 24 active satellites and 3 as backup. They are distributed in 6 orbital planes, assuring 4 visible satellites at all times at a global scale. GPS satellite orbits are ellipses with a small eccentricity (e=0.003) with a flight altitude of proximately 20,180 km [Cojocaru et al., 2009].

GPS is the most widely used Global Navigation Satellite System nowadays, although is not the only one. The Soviet Union developed its own system calling it GLONASS. It is conceptually very similar to GPS. After the fall of the Soviet Union and because of lack of funding the system became only partially available with big gaps in coverage. GLONASS is a military project which Russia inherited after the dissolve of the Soviet Union. Currently Russia is investing in the system in order to restore it to its full operational capacity[Cojocaru et al., 2009]. At full operational state, GLONASS has 24 satellites using 3 orbital planes. GLONASS orbits are circles with a flight altitude of 19,100 km.

The European Union started implementing its own Global Navigation Satellite System in 2003. The development of an independent GNSS that would provide to the European users with more services and reliability than GPS or GLONASS. The system is called GALILEO, an important difference between GALILEO and its two predecessors is that the European system is civilian [Cojocaru et al., 2009].

GALILEO is designed to have 30 satellites, 27 active and 3 as backup. It will use 3 orbital circular planes with a radius of 29,600 km with an inclination of 56 degrees with the equatorial plane. The satellites will have a flight altitude of approximately 23,222 km. With this design it will be possible to have at least 6 visible satellites in any point of the planet at any time [Cojocaru et al., 2009].

The process used by a Global Navigation Satellite System to obtain the location is called trilateration. The signal broadcasted by the satellites contains an ephemeris and an almanac. The ephemeris contains information about the satellite orbit and clock corrections for that specific satellite. The almanac provides information regarding the orbits and clock corrections of the whole system. Figure 3.2 describes the method used by a GNSS to calculate the location of a user. The first step is to measure the distance from the user to satellite A, creating an sphere that contains all the possible user locations. The distance is calculated using the speed of light and the time difference between the satellite and the user. The second step is to calculate the distance to satellite B, the result is another sphere, the intersection of both spheres is a circle that limits the possible alternatives. When we calculate the distance to satellite C, we create a third sphere that when intersected with the circle resulting from the second step results in two possible points. When we calculate the distance to satellite D, we create a fourth sphere that intersects only with one of the possible points which is the geolocation of the user. [Holdener III, 2011].

GNSS can be used for geolocation purposes in mobile devices anywhere on the planet. Currently GPS accuracy is between 5 to 10 meters. Currently most of modern smartphones have an internal GNSS chipset. The disadvantages of using GNSS are:

- The relatively high power consumption.

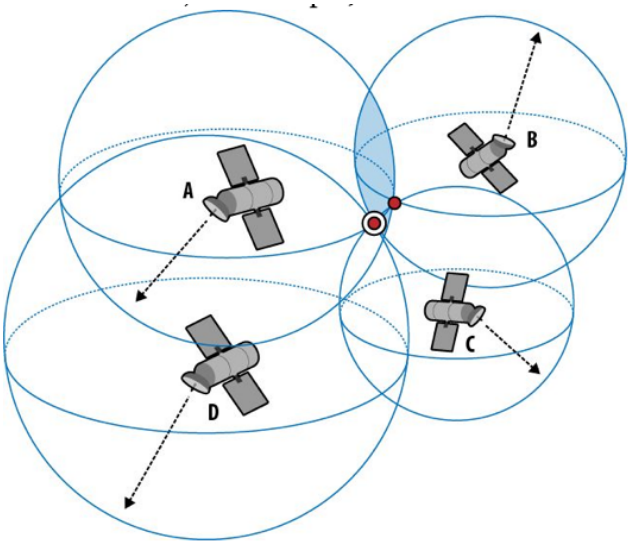

Figure 3.2: Global Navigation Satellite System location (source: [Holdener III, 2011])

- It can be used only in outdoors.
- Depending on the mobile device it might take a long time to lock on the satellite signals.

3.2.2 IP Address

An IP (Internet Protocol) address is an unique identifier for computers connected a computer network using Internet Protocol. The IP address is expressed by 32 bit unsigned binary value. It is represented in a doted decimal format. The mapping between the IP address and a human readable format is done by a Domain Name System (DNS) [Parziale et al., 2006]. The decimal format of an IP address looks like:

128.2.7.9

while its binary format would be:

10000000 00000010 00000111 00001001

The IP addresses are assigned to an Internet Service Provider (IPS) and registered in a local institution. Thanks to data collected by IPSs it is possible to identify the geographic location of a give IP address within a few meters of actual location. However depending on the IPS the precision of the location might vary to even kilometers from the actual location. There are companies specialized in collecting IP addresses information worldwide, creating databases that could be used for geolocation services [Holdener III, 2011].

3.2.3 GSM/CDMA Cell IDs

Mobile devices connected to a cell network have a unique identifier called Cell ID. The most two common networks are:

Global System for Mobile Communications (GSM): Is the oldest of the two and is more widely available than CDMA. It is a 2G technology and is used by 75% of the mobile users worldwide. It is relatively simple to migrate from GSM to 3G and 4G technologies.

Code Division Multiple Access (CDMA): Is a newer technology compared to GSM. It has been implemented as a 2G and 3G technology. Its advantage over GSM is that it allows many users to use the same frequency at the same time.

Using triangulation it is possible to obtain the user location. The precision of the geolocation is in direct relation to the number of towers used in the estimation of the position. Because of this, it works better in urban environments than in rural areas. In United States due to the Enhanced 911 services mandated by the Federal Government all carriers must be able to determine the location of the users with a 300 meters precision (see Figure 3.3).

There are four techniques to calculate the locations using the cell network [Shek, 2010]:

Cell of origin This was the first technique used by the mobile carriers to implement the Enhanced 911 requirement. It is possible to stablish the user location by identifying the cell that is at that moment providing service to

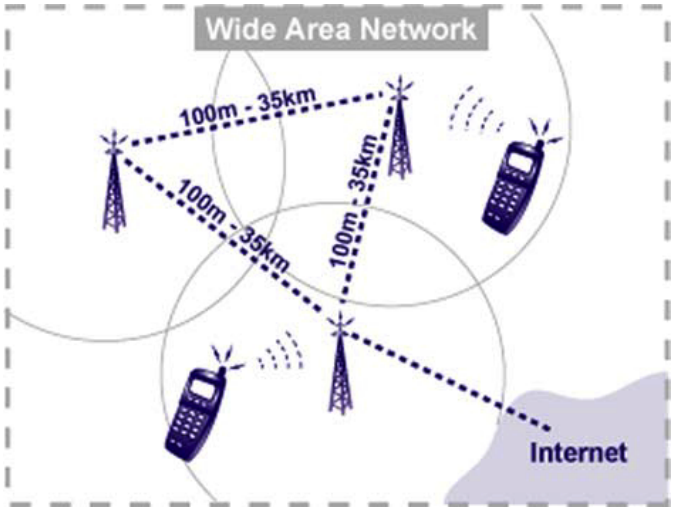

Figure 3.3: Wireless wide area network (source: [Steiniger et al., 2004])

his/her device. Depending on the cell size its accuracy could be around 150 meters in urban environments or around 1 kilometer in rural areas.

Time of arrival This technique uses the distance to base stations based on the time lag. All the calculations are done by the mobile network and not in the device. It has an accuracy between 50 to 150 meters.

Angle of arrival This technique uses the angle of the signals as received in the mobile device. It has an accuracy between 50 and 150 meters.

Enhanced observed time difference This technique is similar to Time of Arrival, although in this case the calculations are made on the device rather than on the network. It also has an accuracy between 50 and 150 meters.

3.2.4 WiFi and Bluetooth MAC Address

This geolocation method works similar to the IP address method. The MAC address is a unique identifier assigned by the manufacturer to each device, however there is a technique called MAC spoofing that allows the manipulation of MAC addresses [Holdener III, 2011].

The identities and the signal strengths that corresponds to distance to public WiFi

points are recorded by the mobile device. Later using a triangulation procedure it is possible to calculate the location with regard to these access points. It has an accuracy between 10 and 20 meters. It is faster and more accurate than using the mobile network techniques and uses less power than GPS, however because it relies on WiFi access points it might not be available in certain places[Shek, 2010].

3.3 Location Based Services

According to the Open Geospatial Consortium a service of this kind uses a wireless IP connection to access data that is used by a mobile terminal [Mabrouk, 2008].

Location based services deal with three basic questions:

- where am I?
- What is near by?
- How can I go to?

Location based services allow to [Shek, 2010]:

- access relevant information, filtering out vast amounts of information available that might not be relevant based on the user context.
- improve the decision making process by supplying the users with timely data.
- access relevant data more promptly, the location of the user is provided to the service automatically and in this way it reduces the amount of information that the user needs to submit to the service in order to obtain a suitable response.
- track the movements of the users, allowing for example to create a visualization of the footprint of the visitors to a park.

  It is predicted that the market for location based services will exceed $12 billion by the year 2014. This figure includes the application sales and mobile advertisement [Shek, 2010].

There are important differences between location based services and regular GIS applications. For instance most desktop GIS applications work in systems with extensive computing resources, while location based services operate in the restrictions of mobile computing environments, less processing power, smaller displays and limited battery run time.

The components of location based services are [Steiniger et al., 2004]:

- Mobile devices
- Communication network
- Positioning component
- Service and application provider
- Data and content provider

There are different ways to classify location based services. Based on their target market there are three types of services [Shek, 2010]:

Publically accessible: Mass market the target is general public, they need to be highly scalable and able to handle large number of requests, being performance a key requirement on their design.

Publically accessible: Niche market This type of applications are also aimed to the public but to a target audience, for example the customers of an specific food chain. In this case the priority of the design shifts to privacy and security.

Internal enterprise applications These applications are designed for the internal use of an organization. In the past it was common to use special hardware as the mobile devices however with the advent of smart phones there is an opportunity to use these devices in this type of applications, however their use also raises concerns about security.

It is also possible to classify location based services based on their purpose of use:

Navigation and routing They provide directions to the user.

Entertainment They include games and social networking services.

Information services Provide answers to questions like "find the closest restaurants".

Accident and emergency services They report emergency incidents to authorities and request for help.

Supply chain management and tracking This is a typical internal application for an organization, they allow the organization to keep track of goods and mobile units (delivery trucks).

It is also possible to classify location based services based on their technical characteristics:

Network vs device centric The location of the mobile device can be obtained using the cell network or using the internal gps of the device. Assisted GPS is also common, in this case the location is obtained by combining information from the mobile network with the GPS, reducing the geolocation time, although the location information is still calculated by the mobile device [Shek, 2010].

Reactive vs Proactive The reactive services are also know as (Pull Services). This kind of services only receive information when directly requested by the user. On the other hand we have the proactive services (Push Services). This kind of services receive information independently of a user action. The reception of information is triggered by an event like entering into a specific area, or a timer [Steiniger et al., 2004].

Most of the location based services operate in a combination of the following tasks:

- navigation: how to get to a certain point of interest.
- identification: obtain information about features located nearby.
- checking: look for events near the current location, this task also considers the temporal dimension.

Because location based services operate on mobile devices, their development have to face certain constrains:

- Most of the devices have limited computer power and memory resources.
- The limited battery power.
- The small displays in the mobile devices require special interface design.
- Because the devices have to operate in outdoors environments the have to face weather influences, like the sun reflection that might affect displays making them hard to use.
- Depending on the location it might be the case that there is a limited access to broadband communication networks.

3.4 Web mapping

3.4.1 Web Map Server

A server of this kind implements a web Map Service Interface Standard (WMS) providing a HTTP interface for requesting geo–registered map images. The WMS request contains information about the area of interest and the required information layers. The response to the request is one or more map images to be displayed in the client browser [de la Beaujardiere, 2006].

Most of the web mapping applications use a server–side tile generator architecture. In this approach the server produces one version of the map for each available scale. Later the map is tiled and stored in the server. When the server receives a request from the client, it provides a selected set of tiles according to the area of interest and scale level specified. This approach is not flexible, the server does not generate new tiles on demand and the user is limited to the fixed zoom levels pre–specified in the server [Kamel et al., 2010].

3.4.2 Web Feature Service

A Web Feature Service differs from previous implementations in the field of geographic data distribution. Previous implementations rely on the file as the unit for information exchange. In the case of Web Feature Service, the unit of exchange is the components of the file, the so called features. Allowing the users to request or modify an specific feature or even only a variable of it. The taxonomy of the services are defined in ISO 19119. The main purpose of a Web Feature Service is accessing to the information at the feature level, although it can also make coordinate transformation and format conversion operations [Vretanos, 2011].

A Web Feature Service can be used in a client - server archicture system providing rich content to a client side application. In many cases when a browser is used in the client as the user interface, it might be necessary to install a propietary plug in on top of the browser. HTML5 is being presented as a solution to this limitation. As by September 2011 this standard is still a work in progress [W3C, 2011]. However it promises very interesting features, for instance using HTML5 it is possible to draw vector graphics through scripting using the "canvas" element. Currently this component is available in most of the browsers in the market that comply with the current specification of HTML5. Using HTML5 it would be possible to display vector information and dynamically manipulate it using Cascading Style Sheets. Using scripting it would be possible to allow the user to interact with the features through events triggers like "click" [Kamel et al., 2010]. In the next section we are going to describe some of the data formats used by web feature services to deploy data in the Internet.

3.4.3 Data formats

XML

This is a markup language used to encode documents readable by both humans and computers. It is defined by W3C specification XML 1.0. It was designed to be used on the Internet, to represent any data structure. Along time many XML based languages have been created for specific purposes, in the next sections we are going to describe two of the most popular XML based languages designed to contain geographic information.

Geography Markup Language (GML) Is a XML based language designed to transport and store geographic information in the internet. It was developed by the OpenGIS Consortium (OGC), GML is an ISO standard (ISO 19136:2007). Is designed to contain not only vector data, but also coverages and sensor data [Lake, 2004] [Lake, 2000]. The purpose of GML is to contain geographic data, independent of its visualization. It is possible to use GML to make maps, by using a rendering tool that would interpret the GML data and transform it into graphics. A common approach is to use transform it into display formats like SVG, VML or X3D using a map styler.

As in any other XML based format GML data is formatted as text with tags. In the following listing we use GML to describe a polygon with associated alphanumeric information. In lines 1 we declare we are describing a polygon and assign a value as its identifier. In line 2 we declare the value for the alphanumeric variable *description*. From line 3 to 7 we declare the coordinate values for an external boundary and from lines 8 to 11 we declare the coordinate values for an internal island.

```
1  <gml:Polygon  gml:id="ExampleGML">
2          <gml:description>"Example
               01"</gml:description>
3      <gml:exterior>
4          <gml:LinearRing>
```

```
5                <gml:coordinates>30.4093,-91.1846
                     30.4084,-91.1850  30.4079,-91.1832
                     30.4088,-91.1828</gml:coordinates>
6            </gml:LinearRing>
7        </gml:exterior>
8        <gml:interior>
9            <gml:LinearRing>
10               <gml:coordinates>30.4089,-91.1846
                     30.4091,-91.1845  30.4087,-91.1830
                     30.4085,-91.1831</gml:coordinates>
11           </gml:LinearRing>
12       </gml:interior>
13 </gml:Polygon>
```

Keyhole Markup Language (KML) This is an XML file format originally created by Google. Its structure is based on XML with nested tagged elements and attributes [Google, 2011d]. It is designed to contain geographic information that needs to be represented on Internet based maps such as Google Maps and Google Earth. On December 2006 Google submitted KML to the OGC so that it could evolve within OGC and become standard [OGC, 2011]. The goal was to armonize KML version 3.0 with existing OGC standards such as GML, WFS and WMS [Schutzberg, 2005]. Datasets using this file format have the extension *.kml* an alternative extension *.kmz* contains *kml* zipped data [Holdener III, 2011].KML has a significant user base, however there are critics to their use, specially in the field of compactness.

A simple point specification using KML looks like:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <kml xmlns="http://www.opengis.net/kml/2.2">
3   <Placemark>
4     <name>my home</name>
5     <description>My current residence</description>
```

```
6       <Point>
7         <coordinates>30.4088, -91.1840</coordinates>
8       </Point>
9     </Placemark>
10 </kml>
```

Java Script Object Notation (JSON)

Is a data-interchange format based on a subset of JavaScript Programming Language. It is a text-based format, with conventions similar to the ones used in C, Java, JavaScript, Pearl or Python languages, allowing its use with these languages.

JSON data is composed by the following elements [Crockford, 2006]:

objects Unordered set of name/value pairs, each name is followed by a colon (:) and the name/value pairs are separated by a comma (,).

array An ordered list of values. An array starts and ends with brackets ([ ]), with value separated by commas (,).

A value can be a string (starts and ends with "), a number, a boolean value (true/false), an *object* or an *array*. The following example contains objects with a nested configuration.

```
1 {
2       "firstName": "Helbert",
3       "lastName" : "Arenas",
4       "address"  :
5       {
6           "streetAddress": "3650 Nicholson Dr.",
7           "city"         : "Baton Rouge",
8           "state"        : "LA",
9           "postalCode"   : "70802"
10      },
```

```
11        "student":false,
12        "phoneNumber":
13        [
14             {
15               "type"   : "home",
16               "number": "225  221-153"
17             },
18             {
19               "type"   : "fax",
20               "number": "225  255-136"
21             }
22        ]
23   }
```

In the previous JSON example, in line 2 and 3 we assign string values to the variables "firstName" and "lastName". in line 11 we assign a boolean value to the variable "student", and in line 12 we assign an array to the variable "phoneNumber" (from line 13 to 22).

When compared to XML based languages JSON has certain advantages. XML based languages because of their tag nature require more characters than JSON formated information, therefore JSON is lighter for data transmission, while is still human readable. The next important advantage is that JSON objects are typed (string, number, array, boolean) while XML is typeless, all data is represented as string, which makes it easier to process using JavaScript [Shin, 2010].

GeoJson Is a geospatial data interchange format based on JavaScript Object Notation (JSON). It is designed to encode geometries, features or collection of features. It supports the following geometries: Point, LineString, Polygon, Multipoint, MultiLineString, MultiPolygon and GeometryCollection. The default coordinate reference system is WGS84, using latitude and longitude with units in decimal format [Andrews, 2007]. Support for GeoJSON has already been implemented in several projects

like QGIS and PostGIS that can create GeoJSON files on the fly.

In the following example we define an object of the type "FeatureCollection". It contains two features, the first one is of type "Point" , it has a location defined by its coordinates, and has one property called "description" which value is a string "residence" (lines 3 to 7). The second feature is of type "Polygon", defined by an array of coordinates. It also has a property "description" which has as value the string "ECE Building" (lines 8 to 19).

```
1 { "type": "FeatureCollection",
2   "features": [
3     { "type": "Feature",
4       "geometry": {"type": "Point", "coordinates":
5       [30.4088, -91.1840]},
6       "properties": {"description": "residence"}
7       },
8     { "type": "Feature",
9        "geometry": {
10         "type": "Polygon",
11         "coordinates": [
12           [ [30.4089,-91.1846],
                [30.4091,-91.1845],
13           [30.4087,-91.1830],[30.4085,-91.1831] ]]
14       },
15       "properties": {
16         "description": "Building footprint",
17         "name": "ECE building"
18         }
19       }
20     ]
21   }
```

Using GeoJSON it is possible to describe the Coordinate Reference

System by defining a CRS object. The following example describes the Geographic, Equidistant Cylindrical projection WGS84 (EPSG:4326).

```
1 "crs": {
2   "type": "name",
3   "properties": {
4     "name": "urn:ogc:def:crs:OGC:1.3:CRS84"
5     }
6   }
```

OSM JSON This is an implementation of JSON following the OpenStreetMap (OSM) data model. It consists on nodes, ways (arcs) and relations. This design allows the definition of nodes, that are later used to define arcs which are later used to define polygons. The advantage over GeoJSON in that in the case of shared borders between polygons, using OSM-JSON there is no need of redundant node definitions.

In the following example we define four nodes (lines 3 to 6) and use them to describe one polyline. From lines 12 to 18 we define its geometry making reference to the previous defined nodes, and in lines 21 and 22 we assign values to two alphanumeric variables “name” and “type”.

```
1 {"osm":
2    {"node":
3       [{"id": "001","lon": "-91.1846" ,"lat":
           "30.4089"},
4       {"id": "002","lon": "-91.1845" ,"lat":
           "30.4091"},
5       {"id": "003","lon": "-91.1830" ,"lat":
           "30.4087"},
6       {"id": "004","lon": "-91.1831" ,"lat":
           "30.4085"}]
7    }
8    {"way":
9       [
```

```
10              {
11                  "visible": "true",
12                  "nd":
13                  [
14                      {"ref": "001"},
15                      {"ref": "002"},
16                      {"ref": "003"},
17                      {"ref": "004"}
18                  ],
19                  "tag":
20                  [
21                      {"v":"name","k":"Nicholson Dr."},
22                      {"v":"type","k":"road"}
23                  ]
24              }
25          ]
26      }
27  }
```

3.4.4 Some Implementations

GoogleMaps

This is one of the best known webmap applications available on the web. It started as a standalone application written in C developed by *Where 2 Technologies*. Later this company was acquired by Google in October 2004. From that point the development model changed and it was decided to be implemented as a Web Application [LeMay, 2005]. This approach allowed a wider user base and according to the designers it was easier to deal with the particularities of multiple web browser than with multiple operative systems.

By the end of the fall of 2004 Paul Rademacher added map mashups to Google maps, allowing the users to overlay their own information on top

of Google maps. In April of 2005 Google launched its own map mashups called *My Maps* allowing users to customize the maps by adding their own information[Ratliff, 2007].

Google Maps provides interactive, highly responsive maps using AJAX (Asynchronous JavaScript ans XML). It provides street and aerial/satellite imagery from Google's Web Map Servers in the form of georeferenced tiles. It also allows user to create customized maps using an open (initially free) API. Google Maps interface allows the users to zoom in/out, pan, and use the mouse to drag and navigate in the map. Using the Google Map API it is possible to overlay on top of Google Maps any user's data. For example, a police department could map the location of crimes in the city, or a real state agent could map the location of its properties [Pimpler, 2006]. On 2011 Google anounced plans to charge for the use of Google Maps API, however the fees will apply only to users that whose pages have more than 25000 visits per day [Google, 2011c].

Currently Google offers the following APIs for Google Maps: 1)JavaScript 2)Flash, 3)Google Earth 4)Maps Image and 5)Web Services. The current version for the JavaScript API is 3 which was designed taking in account mobile devices. In order to work, a website using Google Maps should include a Key provided by Google. Users customizing their own maps can overlay their own information on top of Google Maps layers using as reference system EPSG:3857 (Google Maps use Geographic Coordiantes using a Mercator projection with the WGS84 ellipsoid)[Aitchison, 2011].

The source code for a basic example using Google Maps can be seen in the following listing (Source: [Google, 2011b]):

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta name="viewport" content="initial-scale=1.0,
          user-scalable=no" />
5     <style type="text/css">
```

```
6         html { height: 100% }
7         body { height: 100%; margin: 0; padding: 0 }
8         #map_canvas { height: 100% }
9       </style>
10      <script type="text/javascript" src=
            "http://maps.googleapis.com/maps/api/js?key=
            USER_KEY & sensor=TRUE">
11      </script>
12      <script type="text/javascript">
13        function initialize() {
14          var myOptions = {
15            center: new google.maps.LatLng( 30.4088,
                  -91.1840),
16            zoom: 8,
17            mapTypeId: google.maps.MapTypeId.ROADMAP
18          };
19          var map = new
                google.maps.Map(document.getElementById
                ("map_canvas") , myOptions);
20        }
21      </script>
22    </head>
23    <body onload="initialize()">
24      <div id="map_canvas" style="width:100%;
            height:100%"></div>
25    </body>
26  </html>
```

In line 10 we include the map API JavaScript indicating the key we are using for this map. In line 13 we create a function (initialize()) that contains a set of commands that are required for the map. In line 14 we set some variables for the map, we stablish the coordinates for the map center (line 15), we indicate the initial zoom level for the map (line 16) and the type of

view in this case the road map (line 17). In line 19 we create a JavaScript variable that will store the map and indicate the html element where it will be displayed. The funtion initialize() runs when the website is loaded as indicated in line 23. It is possible to customize the maps by overlaying user's information. In Google Maps jargon those features are called *overlays* and can be of any of the following kinds: GMarker (point), GPolyline or GPolygon.

Google Earth

This is a virtual globe launched by Google in 2004. It was originated as a product of Keyhole Inc. named *Earth Viewer*. The company was later purchased by Google and the product was renamed as *Google Earth* [Bailey and Chen, 2011].

It is designed as a standalone application, by 2007 it had been downloaded more than 250 million times[Ratliff, 2007]. This application allows the user to take a virtual tour to any place in the world. It includes satellite imagery as well as information about cities, countries, roads and administrative boundaries of subnational units. The images are displayed in a 3D digital elevation model. The satellite imagery is stored in Google servers and is updated regularly [Revuelto Luque, 2011]. It is possible to overlay the user's data on top of the Google data/imagery using KML files on the web or locally stored.

Bing Maps

This is a web mapping service provided by Microsoft. It offers 2Pbytes of pre rendered tiles plus 2D and 3D data layers. The original Bing Maps was based on Ajax. In December 2009, Microsoft released Silverlight a plug in similar to Adobe Flash. Silverlight allows Bing to provide 3D experience, using a technique called deep zoom on pre rendered images[Pendleton, 2010].

The street maps provided by Bing offer the following views [Pendleton, 2010]:

Road view This is the default view, it shows a 2D cartographic representation of the roads and buildings.

Aerial view This view results of overlapping satellite imagery on top of the road view.

Bird's eye view This option shows oblique aerial photographs , in rural areas it is generated by combining aerial photos with digital elevation models, while in urban areas the process includes lidar, and GPS measurements to create 3D models.

Bing Maps are designed as a canvas on top of which it is possible for the user to add his/her own data, elements like tweets from tweeter, or information regarding rental properties from *oodle.com* or links to social networks. Using Photosynth a tool released by Microsoft it is possible for a user to geo-register his/her photographs using Bing information and link his/her images to images of other users creating an augmented reality environment [Pendleton, 2010].

OpenLayers

There are already implementations for the visualization of geospatial information using HTML5. One of those is OpenLayers, described as an open source, pure object–oriented client side JavaScript library that allows the creation of dynamic interactive web maps viewable in almost any web browser [Kulawiak et al., 2010]. Because it is a library it does not require the installation of any special software in the client computer. It was originally developed by Metacarta in 2005, later it was released as open source, currently it is an Open Source Geospatial Foundation project.

Maps developed with OpenLayers require minimun programming skills, and because is open source its users do not need to pay fees. Currently there is a strong community of developers working in this project. It is also possible to obtain plenty of documentation as well as to find plenty of examples on the Internet. Much efford has been put in making OpenLayers compatible to

multiple web browsers. It does provide support for mobile devices supporting touch screens even with multitouch gestures. Because is an open source library it allows much more flexibility and customization than commercial alternatives [Hazzard, 2011].

OpenLayers allows the client application to connect to map servers such as Google Maps, Yahoo, Esri ArcGIS, WFS or OpenStreet Maps and display their information. The information displayed in an application using OpenLayers is structured as layers. The data source for a layer can be a traditional Web Map Server or vector data from online remote servers or vector data locally stored. There is more than one standard for the transmission of geodata through the internet, some of them are based on XML, for example, GML, or KML, however there are alternatives like using Javascript Object Notation (JSON).In the case of OpenLayers, it allows the use of KML and GeoJSON files [Hazzard, 2011].

The following listing illustrates a webpage that contains a basic map using *OpenLayers*. In line 5 we include the library *OpenLayers.js* that provides the interface funcionality to the map, allows us to display vector and raster information and access Web Map Servers. In line 7 we create a variable called *map* and in line 8 we create a variable called *layer*. In line 10 we define the variable *map* as an instance of the class *OpenLayers.Map* and indicate what element of the html document will contain it. From line 11 to 14 we declare *layer* as an instance of the class *OpenLayers.Layer.WMS* , we are going to use it to contain data from a Web Map Service, we define the name of the layer, then we indicate the URL of the WMS among other characteristics. On line 20 the actual body of the html page starts, the map is included inside a *div* section.

```
1 <html>
2   <head>
3     <meta http-equiv="Content-Type"
4     content="text/html; charset=utf-8">
5     <script src="../OpenLayers.js"></script>
```

```
6     <script type="text/javascript">
7         var map;
8         var layer;
9         function init(){
10            map = new OpenLayers.Map( 'map' );
11            layer = new OpenLayers.Layer.WMS(
12            "OpenLayers WMS",
13            "http://vmap0.tiles.osgeo.org/wms/vmap0",
14            {layers: 'basic'} );
15            map.addLayer(layer);
16            map.zoomToMaxExtent();
17        }
18    </script>
19  </head>
20  <body onload="init()">
21    <div id="map" class="smallmap"></div>
22  </body>
23 </html>
```

Maps using OpenLayers can access locally stored data, allowing them to work using this data even in offline mode, with no Internet connection. In the case of applications that can not rely on a constant Internet connection, this represents an advantage, applications of this kind would use the Internet connection periods to update the locally stored information allowing them to operate in a suitable way while they can not access the Internet.

The default installation of OpenLayers only provides limited GIS capabilities, however more advanced features can be added manually. There is a great number of examples that allow the developer to gain understanding of advanced components. A well documented example is described by Kulawiak et al. (2010), in this document the authors describe the development of a Web GIS client in the framework of the MARCOAST project. The goal of the project is to visualize and map spreading scenarios of oil splills in the

Aegean Sea, Greece. The authors indicated that by the time they developed their application OpenLayers had certain capabilities not found in ArcIMS, however as both softwares evolved the difference in capabilities tended to dissapear. As a general conclusion they indicated that it is becoming much easier to develope advance Web GIS services using Open Source solutions [Kulawiak et al., 2010].

Cartagen

Another interesting implementation is Cartagen, an open source vector mapping framework developed by the MIT Media Lab's Design Ecology group. It generates maps based on vector data in the client side using JavaScript. Because the features are drawn in the client, it is able to produce a smooth zooming, it does not depend on fixed scale levels on the server. Cartagen reads vector map data in a JSON format that follows the OpenStreetMap data model, similar in structure to the OpenStreetMap XML data format, although with a JSON structure [Kamel et al., 2010]. This format eliminates redundant node definition making it more efficient in terms of file size than GeoJSON. However to our best knowledge there are no software implementations that can create OpenStreetMap JSON on the fly.

3.5 Overview of Mobile devices

Nowadays mobile devices known as smartphones and tablets are increasingly gaining market share. What exactly are these devices?

Let's start with the smartphones. Although there is not a canonical definition we can say that these are devices that have capabilities comparable to miniature computers, while at the same time are able to receive phone calls.The first smartphone were created by IBM in 1992, they call it Simon [Fendelman, 2011]. Smartphones originated by combining features from PDA's, computers, and regular cell phones. There are certain characteristics

that most would consider a requirement for a cellphone to enter into the smartphone category [Cassavoy, 2011]:

Operative System Smartphones do need to have an operative system that allows them to install and run third party applications.

Applications Smartphones should be able to download and run customised software programs.

Web Access A device of this kind should be able to access the internet through 4G, 3G and Wi-Fi networks.

QWERTY keyboard It should include a keyboard that allows the user to interact with the device. The keyboard can be hardware (with physical keys) or software based (touch screen).

Messaging Most of cellphones can handle sms, however smartphones are able to access email and instant message servers.

The second type of mobile devices that interest us are tablets. Microsoft introduced the first windows tablet PC in 2000. It had many common characteristics to a laptop, although it also included a touch screen capable of hand writing recognition. Until the year 2010 other providers created similar devices although with limited success [pcmag.com, 2010]. In the year 2010 Apple introduces the iPad which becomes an instant sales success. This device is basically a portable touch screen that allows the users access to applications and services provided by Apple.

It is worth to note that Apple has never called its creation a "tablet". According to Bajarin(2011) this is a way to differentiate its product from Microsoft's tablet PC [Bajarin, 2011] [Sutter, 2010]. However after the release of iPad other providers began to offer products with similar characteristics.

The common characteristics of products of this kind are:

Operative system That allows the device to install third party applications.

Screen This is the main interface of the device, is a touch screen, currently is more common to find multigesture touchscreens that have the ability to detect two or more points of contact allowing more complex interactions with the device. Most of the screens are designed to produce less saturated colors, in an attempt to increase screen brightness while preserving power efficiency and battery time, however these characteristics also limit the use of the tablet in outdoors environments. The larger the tablet screen the more difficult it is for the user to position him/her self and the screen in order to avoid reflections. There is a limited viewing angle in many of the tablets which becomes an issue if there are multiple viewers [Bibby, 2010].

Processing capabilities Superior to smartphones.

Networks connectivity 3G, 4G or Wi-Fi The device is capable to connect to the internet and access online information.

Media display The device is capable to display multimedia.

Gartner estimates that the sales of tablets are around 69.7 millions for the year 2011, while the sales of this type of products might reach 108 million in the year 2012 and will be around 3 hundred million for the year 2015 [Bibby, 2010]. It is expected that the number of users accessing the internet from mobile devices will exceed the number of users using desktop devices by the year 2014 [Davis and Rocchio, 2011].

3.6 Most common Operative Systems

The sales of smartphones and tablets have increased dramatically in the last couple of years. In the first quarter of 2010 mobile phone vendors sold worldwide 314.7 million of units, including smart and non smart phones. By the first quarter of 2010 smartphones represented 17.3 percent of the total mobile phone sales. However this figure represents an increase of 48.7 percent in comparison with the previous year (13.7%), indicating a strong

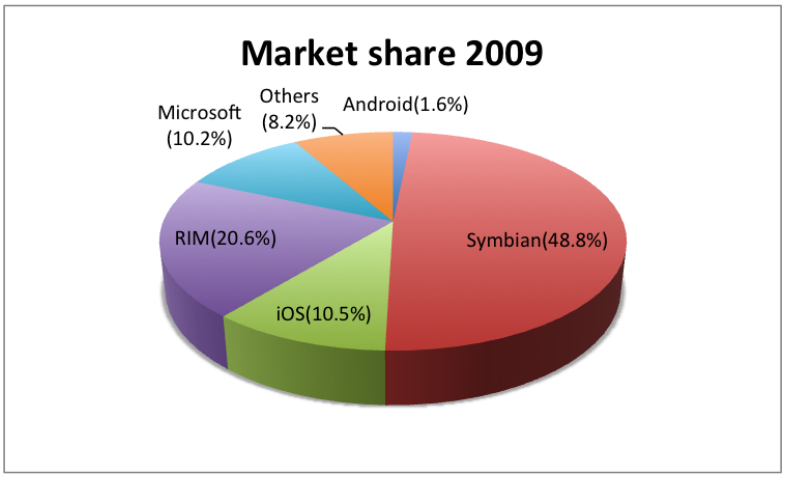

Figure 3.4: Global market share 2009 (source: [Gartner, 2011a])

growth in this segment of the market. The most common operative systems deployed with these devices are Symbian, BlackBerry, iPhone OS, Android and Windows Mobile (see figures 3.4 and 3.5).

In this section we are going to provide a brief description of each one of the current most important operative systems deployed with smartphones or tablets.

3.6.1 Symbian OS

This operative system started as EPOC a product developed by Psion for portable devices in the 80s. In 1998 Psyon Software became Symbian Ltd. a joint venture between Nokia, Ericksson, Motorola and Psion, and EPOC was renamed as Symbian OS. In 2009 Symbian OS became open source under Eclipse Public License (EPL) [Grossman, 2000].

Currently Symbian OS has the largest share of the smartphone market, however its consumer base is being eroded by the growth of Android and iPhone OS [Gartner, 2011a] (see figures 3.4, 3.5 and 3.9). On February 2011 Nokia the main contributor to Symbian OS code, announced that it would adopt Windows Mobile for its smartphones. This event indicates that the number

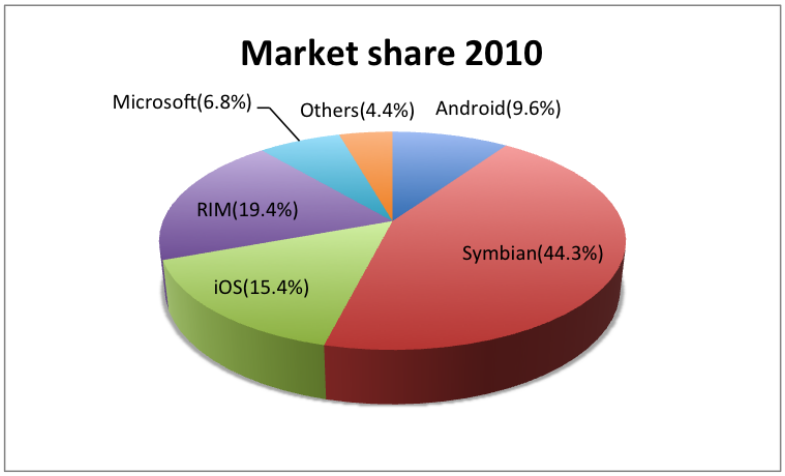

Figure 3.5: Global market share 2010 (source: [Gartner, 2011a])

of devices using Symbian OS will diminish rapidly in the future.

Applications for Symbian are developed using C++, using many possible development environments. Currently there are many commercial and free development tools for Symbian OS. Applications for Symbian are compiled for specific target devices. An application for Symbian OS is deployed through a *SIS* file. Nokia launched in May 2009 its Ovi Store, designed to provide Nokia customers with mobile games, applications, videos, images and ringing tones. Currently is the third largest application store behind App Store (Apple) and Android Market.

3.6.2 iOS

This is an operative system designed by Apple for its products:iPhone, iPod Touch and iPad. It was released on June of 2007. originally it was called iPhone OS, it was later renamed to iOS. It is derived from Mac OS X an operative system used in laptop and desktop Apple computers, which is based on Darwin, an Unix like operative system [Liu et al., 2011].

iOS has four technology layers as are depicted in figure 3.6:

– Cocoa Touch

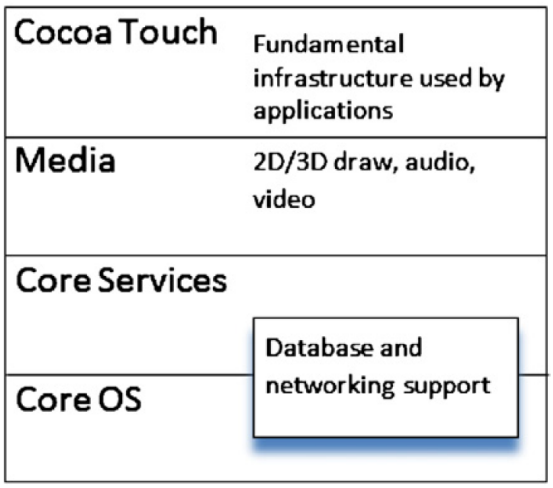

Figure 3.6: iOS Technology layers (source: [Liu et al., 2011]).

- Media
- Core Services
- Core OS

Core OS and Core Services provide support for basic functions like access to databases and networking. The Media layer provides support to 2D and 3D visualization, audio and video. Cocoa Touch is the layer that provides the infrastructure for developers. It was developed based on Cocoa the interface used for OS X, however Cocoa Touch was specifically designed to deal with touch screens. The development language for applications in iOS is Objective C. The most common approach when developing applications for iOS is to start with the top layers and use lower level libraries only when more precise control of the device is necessary [Liu et al., 2011].

Applications for iOS can be developed using Objective C and Object Pascal. There are free development tools, testing the application is free, however installing it on the actual device requires a fee and a signing key. Applications for iOS are only deployed through the App Store, and need to be reviewed and approved by Apple Inc. [AppleInc., 2012]. Currently Apple's App Store has the largest share of the applications market in terms of sales and at the

same time it has the largest number of available applications, by October 2011, there were more than half million available applications in Apple App Store [Wauters, 2011].

3.6.3 Android

Android is a creation of Google and the Open Handset Alliance. As its main creator, Google has made agreements with different telephone handset manufacturer and service providers to aggressively promote its use. Figure 3.9 depicts the evolution of the sales of smartphone handsets sales in North America every quarter since the first quarter of 2009 until the third quarter of 2010 [Butler, 2011]. In the figure we can see that the only operative systems showing sustained growth growth are Android and Apple iOS, while the rest seem stagned.

Android has a Linux kernel based operative system. The inner components of the system are developed in C or C++, however applications designed for Android are written in Java language using the Dalvik virtual machine [Ableson et al., 2011]. Although the applications developed for Android use the Java language, they do not use a Java VM in runtime. Android uses its own virtual machine called Dalvik. At this moment, Oracle the current owner of Sun Microsystems the original creators of Java, alleges that there has been a patent infringement and is currently challenging Google in the USA legal system [Frankel, 2011].

The market share of Android is constantly increasing since its release. By the second quarter of 2010 Android's market share was 17.2% one year later it was 43.4%, by the third quarter of 2011, 52.5% of the smartphones had Android as their operative system [Gartner, 2011b](see figures 3.7 and 3.8).

An application in Android can have components of four different types, although not necessarily all of them:

Activity: This component represents a graphic user interface. An application can have more than one activity depending on the tasks the user

wants to perform with the application. To create a component of this kind, it is necessary to instantiate the class Activity embedded in *android.app.Activity* Java package [Google, 2011a] [Ableson et al., 2011]. Elements of this kind implement two methods:

onCreate(Bundle)

onPause()

Service: If the application needs to perform tasks on the background while the user interacts with other applications or even turns off the screen, it is necessary to implement a *service*. A component of this kind is an instace of the class *android.app.Service* [Google, 2011a] [Ableson et al., 2011].

The service is created when someone executes the order *Context.startService()* at this moment the instantiated service runs its method *onCreate()*. The service continues working until the command *Context.stopService()* is executed.

BroadcastReceiver: Components of this class are instances of *android.content.BroadcastReceiver*. They are used to send messages between applications[Ableson et al., 2011].

ContentProvider: This type of components allow the application to share information with other applications through read/writing files, SQLite databases, or even memory–based hash map [Ableson et al., 2011].

Android is a powerful software platform and it is getting more popular among users, developers and hardware providers

3.6.4 BlackBerry OS

This is a mobile operative system developed by the Canadian company *Research in Motion (RIM)* for its Blackberry smartphones and mobile devices. It supports multitasking and specialized input devices commonly found in RIM devices (trackwheel, trackball). The first version of this operative sys-

tem was released in 1999 for the Pager BlackBerry 580. The current version is BlackBerry OS 7.1.

This operative system supports regular cellphone functions plus functions like "Direct connect"(walkie talkie mode) and "Group Connect" (conference call). It also provides user with email and internet access. One of the most attractive characteristics of Blackberry is its encrypted mailn [Straus et al., 2007].

Applications for BlackBerry OS are distributed through BlackBerry App Store. This environment allows RIM users to browse download and update third party developed applications. The service was launched on April 2009 although the number of available applications is still smaller compared to Android Market or App Store for iPhone or iPad (See figure 3.10) [Distimo, 2011].

3.7 Comparison of current Operative Systems

An analysis of the global sales of smartphones considering its operative system show us that this market is dynamic. In the second quarter of 2010 Symbian market share was 40.9%, by the second quarter of 2011 it was 22.1% and by the third quarter of the same year it diminished to 16.9%. In the case of iOS, its market share by the second quarter of 2010 was 14.1%, one year later its market share was 18.2% but by the third quarter of 2011 it was 15.0%. RIM OS had 18.7% of the market by the second quarter of 2010, its share reduced to 11.7% one year later, and by the third quarter of 2011 it was 11.0%. In the case of Android its share by the second quarter of 2010 was 17.2%, it increased to 43.4% by the second quarter of 2011 and increased again to 52.5% by the third quarter of the same year [Gartner, 2011a] [Gartner, 2011b](see figures 3.7 and 3.8). An analysis of the sales in North America [Butler, 2011] indicate that the sales of devices using Android and Apple iOS have a strong growth, while the sales of devices using the rest of most common operative systems remain stagned (See figure 3.9).

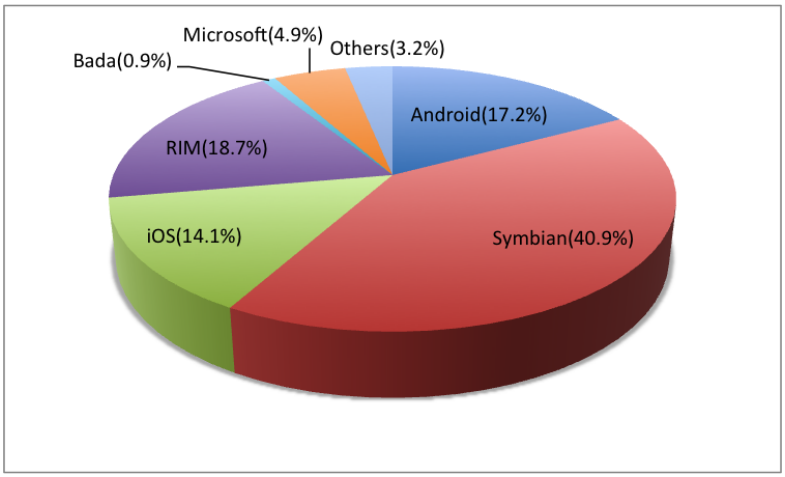

Figure 3.7: Global sales of smartphones in the second quarter of 2010 (source: [Gartner, 2011a])

According to the analysis firm Distimo, a comparison of the application stores by number of available applications indicates that the top three are iPhone, then Android and in third place iPad. On the other hand, Nokia Ovi store and BlackBerry App World remain much smaller compared to the first three [Distimo, 2011]. The iPhone application store is the largest in terms of available applications, however by March 2011 it was the one with the slowest growth compared to the rest of application stores. An analysis of the type of applications in the markets indicates that Google Android Market has the largest number of free applications [Distimo, 2011] (see figure 3.10).

An important issue when developing an application is the availability of documentation. For this research we compared the available programming books in Amazon.com for the four most used operative systems. Our results indicate that by January 2012 the platform with most available information is iOS with 532 books, then Android with 477 books, then Symbian with 184 books and finally RIM OS with 75 books.

Based on the analysis of the market trends, available applications and reference books we believe that the most suitable platforms for the development of the avalanche risk application prototype are Android and iOS. For the purposes of an avalanche risk application, devices using these operative sys-

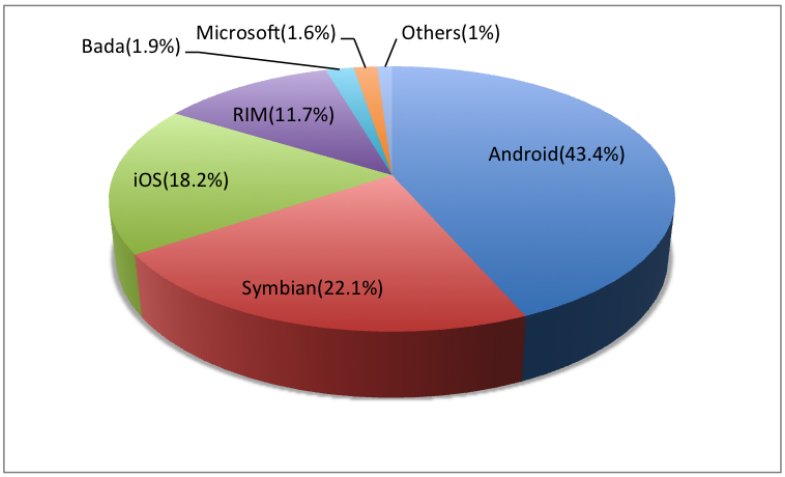

Figure 3.8: Global sales of smartphones in the second quarter of 2011 (source: [Gartner, 2011a])

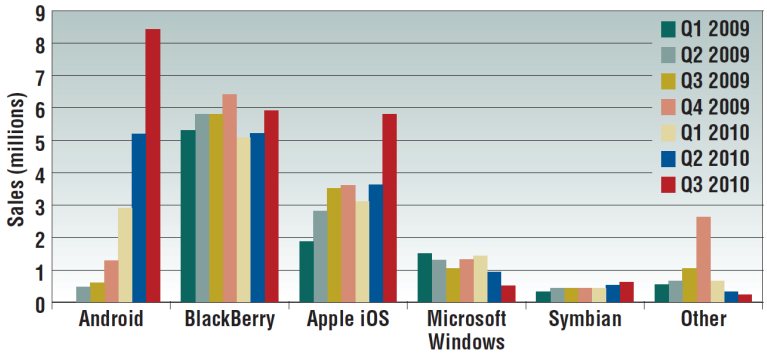

Figure 3.9: North America handset sales by quarter (source: [Butler, 2011]).

tems have in broad terms similar hardware characteristics (GPS, internet, multimedia).

In terms of application development and deployment the main difference between Android and iOS is is that in the case of the later a payment of a fee and the approval of Apple Inc. is required. For commercial purposes these barrier assures the quality of the product but also limits the number of free applications in the Apple App Store. In the case of Android there are no barriers for the application development.

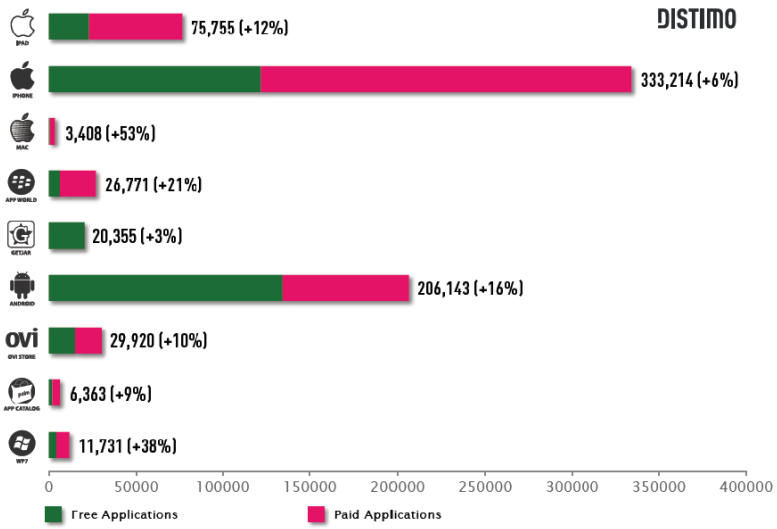

Figure 3.10: Number of Available Applications (source: [Distimo, 2011])

3.8 Summary and conclusions

For the specific case of an avalanche risk application the network access becomes a limitation. It is possible that users of an application of this source might find themselves in locations with limited or unreliable network connectivity. An avalanche risk application should consider this factor, and take advantage of the moments when the device can connect to the Internet to obtain updated data to use later when no connection is possible.

Because of the network limitations, an avalanche risk application would require to obtain the user location using a GNSS. Currently from the discussed options GPS is the most viable alternative.

In section 3.3 we discussed Location Based Services. The application will be deployed in areas where the access to a network is unreliable. Because of this, an avalanche risk application would need to be *device centric*, in the sense that all the processes must run on the device. The application also needs to be proactive in the sense that it should request avalanche risk information without the human user intervention when a Internet connection is available.

An avalanche risk application requires to perform analysis with spatial avalanche risk information requested from a server. This type of analysis is easier to perform using vector data, because of this, the system would use a Web Feature Service. In section 3.4.3 we described the most used vector spatial data formats for transmission on the web. Based on the characteristics of the formats we consider that the most suitable is GeoJSON. It is more compact than XML alternatives and because of its structure is easier to parse using JavaScript or Java.

In section 3.4.4 we discussed some of the most common implementations for web mapping. Because of the characteristics of the environment where the application is going to operate, it is not possible to assure an internet connection 100% of the time. Therefore is required an implementation able to work offline. OpenLayers fulfill this requirement, it is open source, does not require any payments and there is a good amount of documentation available that would support any project using it.

An avalanche risk application for mobile devices should consider the trends in the market. It would be better to choose a platform with an increasing number of users in order to secure its future use and further development. Another important requirement is the vailability of adequate documentation. Our book survey in Amazon.com as well as the analysis of the sales of smartphones/tables indicates that both Android and iOS are currently the operative systems showing the strongest growth. In terms of hardware characteristics, devices using these operative systems are in broad terms, similar (Multigesture touch screen, GPS, network connectivity, multimedia, etc). However due to the restrictions imposed by Apple Inc. it is easier to develop and deploy a prototype using Android.

Current mobile devices such as smartphones and tablets have capabilities that allow their use as tools to reduce the risk of avalanche incidents for users on the field. Such capabilities are Geolocation, Multimedia, access to internet and the capability to display customized maps. However these devices have certain limitations that have to be considered when develop-

ing applications for them. First, they have much more limited resources compared to desktop computers, in terms of display, battery and processing power. The smaller displays require special attention. It is necessary to careful consider what information to show because of the limited space. Another limitation is related to the display and the environment. These devices operate on outdoors and under certain circumstances sun reflection might make them difficult to operate.

Chapter 4

Avalanche Application

4.1 Introduction

In the last years there has been an increase on the number of avalanche related accidents, this is due to the larger number of people involved in winter outdoor recreation activities. The reasons why people get involved in this type of accidents are not only related to lack of proper training/education. There are psychological reasons that affect the sportsmen behaviour making them take unnecessary risks or disregard obvious signs of danger (See section 2.2.1).

There is a need for an application designed to reduce the avalanche risk for people involved in winter outdoor sports [Meng and Reichenbacher, 2005]. This application is designed for both experts and unskilled users involved in winter outdoors recreation activities. In the case of non experts users the application would provide an educated advice about how to proceed while in the field. In the case of expert users the application would help the decision process by providing an opinion that is unaffected by social/psychological variables. An application deployed in mobile devices with access to the most updated risk information would act as an expert adviser providing a reliable opinion under all circumstances.

4.2 Use Cases

We have defined two use case scenarios:

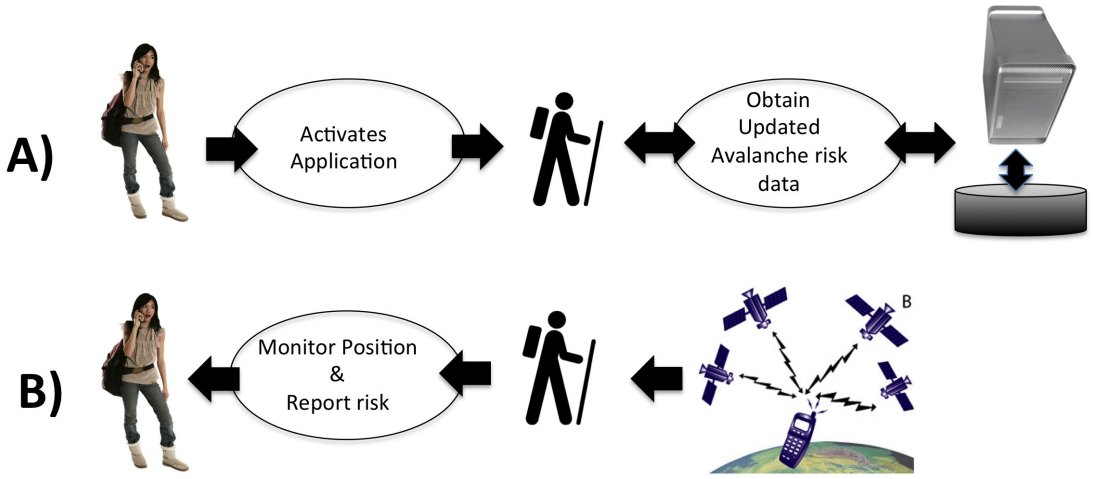

Figure 4.1: User case scenarios

Downloading the risk information: In this case the user is located in an area with internet access. When the device detects that it can access the internet, it automatically connects it self with an internet server and requests the mos updated avalanche risk data. The information is downloaded in the device and stored for future use (See figure 4.1).

Monitoring position in the field: When the user activates the application in the field, the software proceeds to obtain the location of the user from the onboard GPS and monitors that position by comparing it to the avalanche risk data. When the user enters into an area described as risky the application proceeds to warn the user by activating an alarm (See figure 4.1).

The first part of the system has not been developed, currently an internet server that provides updated high resolution avalanche risk information does not exist. There are plans to implement it in the near future, however by the time we developed this application it was not in place yet. For this project we assumed a Web Feature Server providing vector avalanche risk

data is in operation. The implementation of a server of this nature is not technically complicated, but requires a certain amount of investment in order to keep it working and providing updated information. For this application we developed the second part of the system. Due to the lack of a server we uploaded the information manually into the device before deploying it in the field.

4.3 Implementation

The application was developed on Android, this is an open source software platform designed for mobile devices such as smart phones and tablets. Android is a creation of Google and the Open Handset Alliance. Android has a Linux kernel based operative system. The inner components of the system are developed in C or C++, however applications designed for Android are written in Java language using the Dalvik virtual machine [Ableson et al., 2011].

It is important to remark that although the applications developed for Android use the Java language, they do not use a Java VM in runtime. Android uses its own virtual machine called Dalvik, which according to their creators is not the same as the original Java VM [Frankel, 2011].

4.4 Model Specification: Software components

This application is composed by five main elements: (see figure 4.2)

- AvalancheUI
- RiskEvaluatorService
- RiskPolygon
- MyLocationListener
- Webview (visualization module)

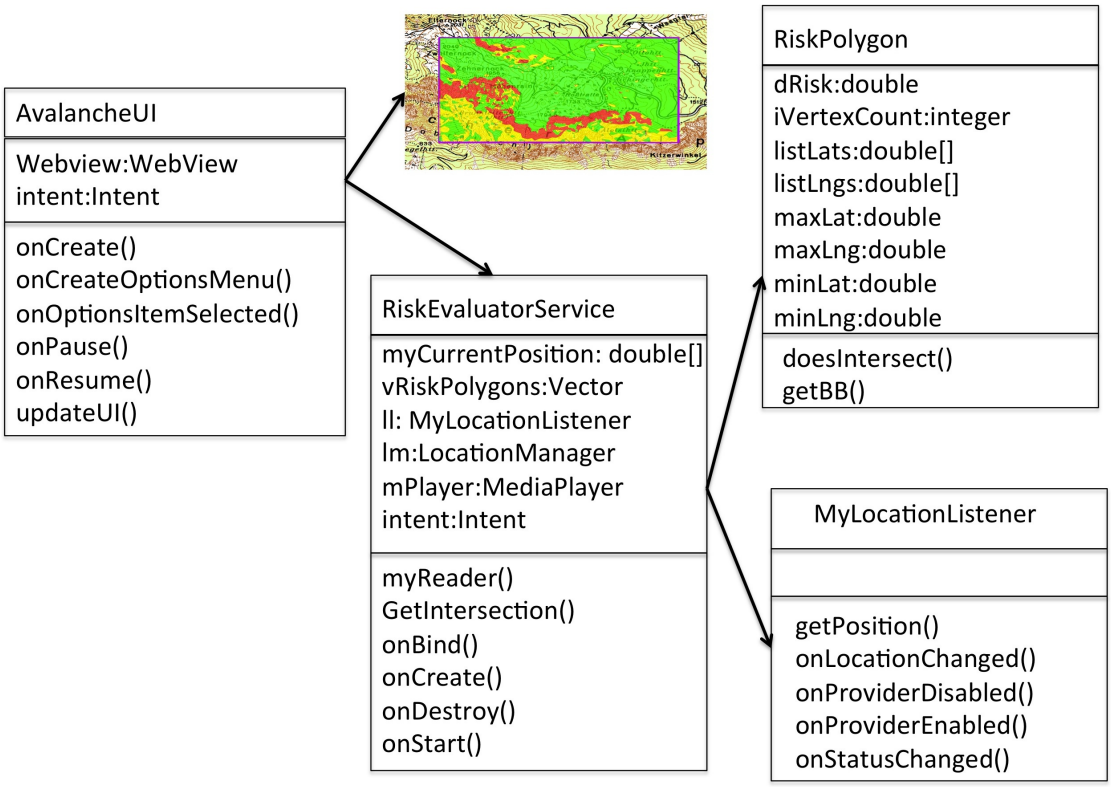

Figure 4.2: UML Class diagram of the application

4.4.1 AvalancheUI

This is an *activity* component and works as the main user interface of the application. It is created when the user loads the application on the mobile device. The interface is composed by a visualization module that contains the map and a menu component.

Menu component

This component contains the buttons that allow the user to interact with the application. It is only visible when the user selects the menu button of the device, the rest of the time it remains hidden. It was designed in this form because of the limited screen area of smart phones. The menu is encoded as a XML file, the Application reads this information and transforms it into the

application menu (see Appendix A.1, lines 86 to 91). It contains 3 buttons. It is associated with a listener event that performs certain task based on the user input (see figure 4.3):

Start Service This menu button activates the service that monitors the location and evaluates the risk (see Appendix A.1, lines 99 to 108).

Stop Service This menu button stops the location monitoring and risk evaluation. Monitoring the position puts an extra demand on the battery of the device, this option is necessary in order to save battery power (see Appendix A.1, lines 110 to 117).

Turn On/Off Risk Layer This button is used to communicate the java application with the map interface. It executes a JavaScript command on the *webview* that turns on or off the risk data layer (see Appendix A.1, lines 119 to 125).

```
webview.loadUrl("javascript:layerTurnOffOn()");
```

The full code of the JavaScript function *layerTurnOffOn()* can be found in the Appendix B.2 from lines 102 to 115.

Visualization module (map)

The map contains a cartographic representation of the area of interest. In the application we create it as an instance of the class *android.webkit.WebView*. This type of components are used to display web pages including JavaScript code (see Appendix A.1, lines 33 to 37). The map was developed using OpenLayers an open source JavaScript library and it is stored in the device. The html page has links to the file that contains the core of OpenLayers functionality, to the one that stores the risk polygons as well as to the file that contains the customized JavaScript functions (Appendix B.1 lines 10, 13 and 15 respectively). Because the OpenLayers libraries are stored locally the application does not need to access the internet in order to have functionality. The user can interact with the map, turn on or off layers, zoom in/out or

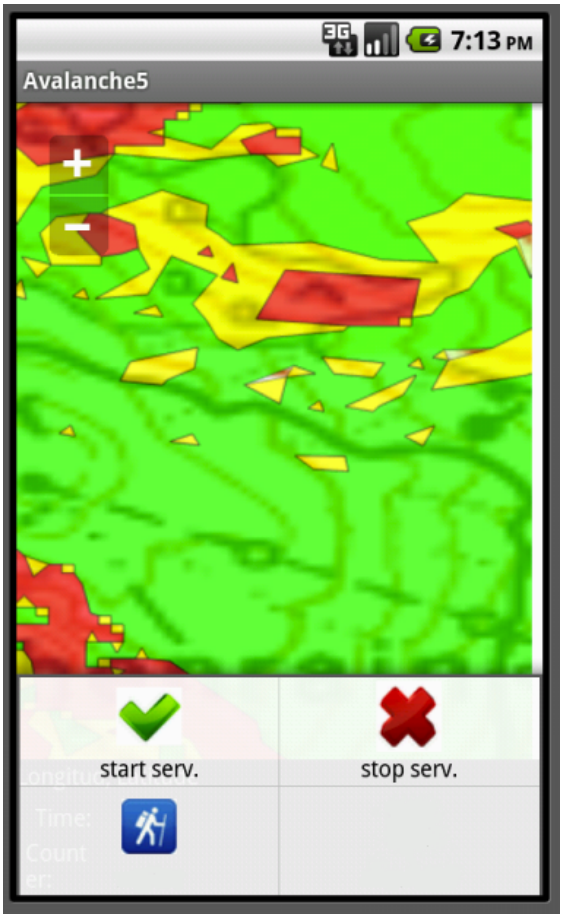

Figure 4.3: Application Interface

pan using locally stored data, with no need for an internet connection. This fact is an advantage compared to alternatives like Google Maps or Bing.

The map has two data elements (See figure 4.4):

Risk layer: This is dynamic data in the sense that is updated periodically. In the design we consider this as a fundamental requirement. The avalanche risk information is stored in the device in vector format.For this application we decided to use GeoJSON formatted data because it is more compact than any XML based alternative like KML. This format allows the representation of complex polygons. A short example of the risk polygons can be seen in Appendix B.3. The visualization

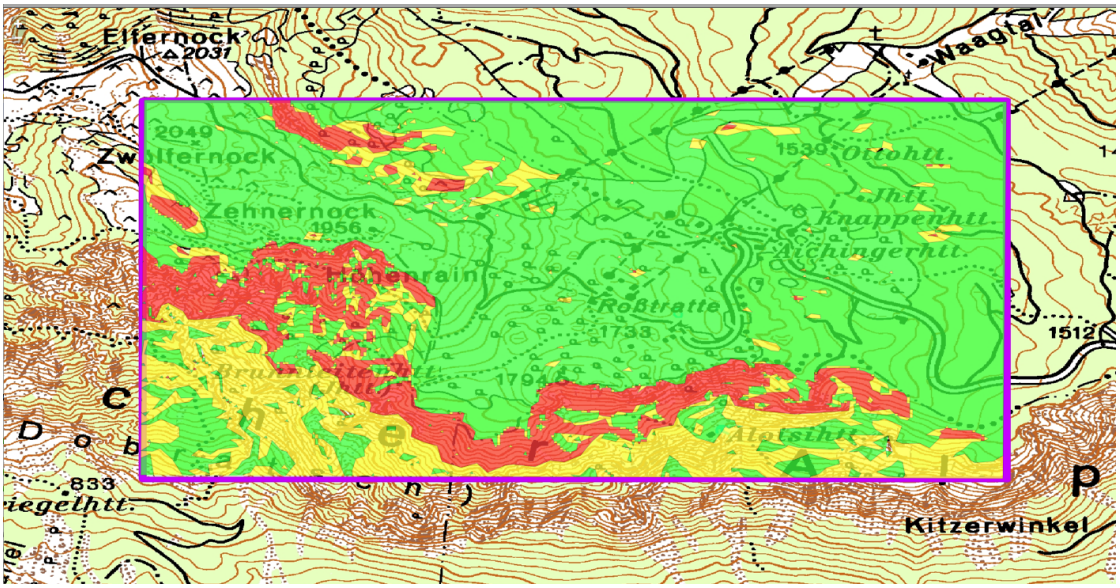

Figure 4.4: Map of the test area using OpenLayers

module is constructed as a html webpage with JavaScript functions that works with locally stored data. Appendix B.2 contains the code used by the webpage. Because it is formated as regular JavaScript data, we can include the risk data in the header of the html file (line 13 of Appendix B.1).

```
<script src="geojson/risk.js"></script>
```

Appendix B.2 contains the JavaScript code we use to set the map. From lines 32 to 34 we define how we want the polygons to look on the screen (fill color, stroke line color, width etc) using CSS. From lines 37 to 44 we define 3 information layers, one for each risk level polygons and finally we use the information stored in variables to draw the risk polygons (lines 75 to 82).

Background data: This type of information is static. It represents features of the area of interest like roads, topography or places of interest. It can be loaded in vector or raster format. For the final prototype we loaded a scanned image of a cartographic map of the area. *OpenLayers* allows the use of images and its georeferencing. For this purpose is only

necessary to indicate where is the image located, its bounding box and its size (See Appendix B.2 lines 27 to 29).

4.4.2 RiskEvaluatorService

An *activity* stops working when the user activates other application. An application designed to monitor the position of the user and contrast it with the avalanche risk information would require that the application is permanently operating. To solve this problem Android allows the creation of *Services*. These are a series of commands that the application runs on the background without user intervention even allowing the user to run other applications. In this application the *activity* allows the user to start the service that monitors the current position and evaluates the risk periodically. This component of the application performs a set of tasks on the background without requiring user input. The complete code for the service RiskEvaluatorService is in Appendix A.2.

When the service is initialized it executes the function *onCreate()* that includes the following set of processes: (see Figure 4.5).

myReader() When the service is created its first task is to get the risk information. For this task the service executes a function called *myReader()* (See Appendix A.2 from lines 155 to 201). This function has access to the internal storage of the device. The function reads a GeoJSON formatted file that contains the avalanche risk information in the form of a set of polygons each one with a predetermined risk factor. The information is parsed and we create an instance of the class *RiskPolygon* for each one of the polygos stored in the data file (See Appendix A.2 line 183 and Appendix A.3 from line 5 to 41) (See figure 4.2). Once the instance is created it is stored in an array of objects of type *Vector* called *vRiskPolygons* (See Appendix A.2 line 185). A vector is a Java construction that has a flexible size, it can store any number of objects of diverse nature.

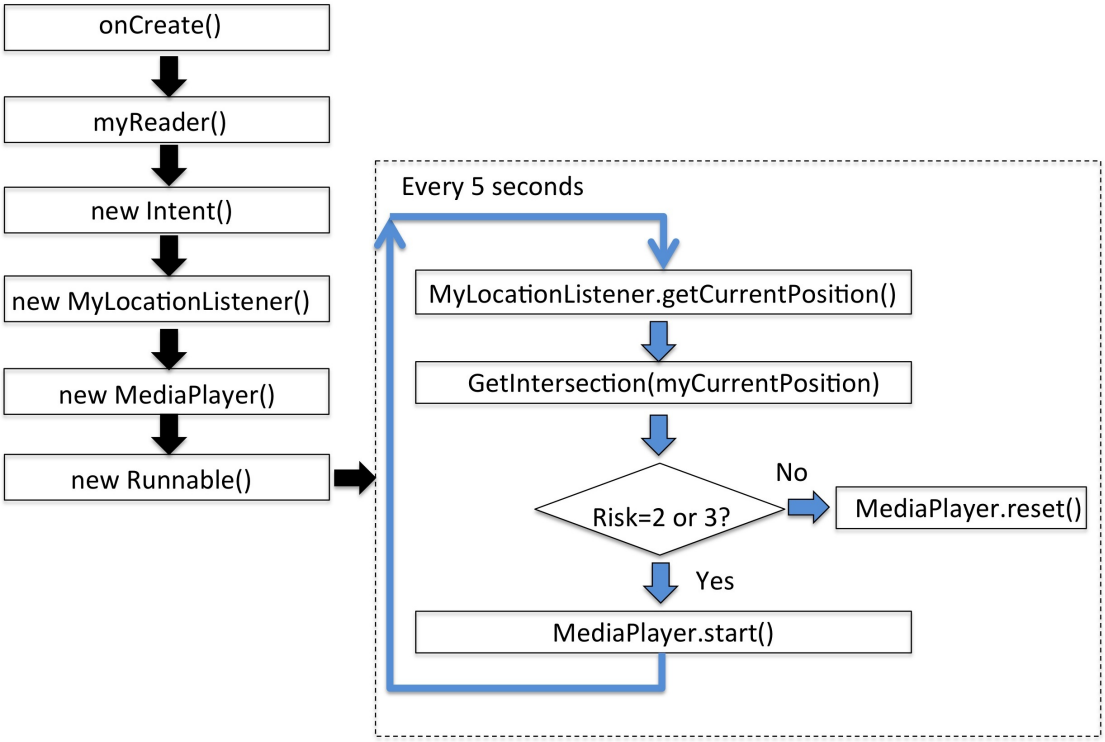

Figure 4.5: Flow chart diagram of Service onCreate event

intent=new Intent() In the next step the service creates an *Intent*, which is a component used as a communication link between the *Service* and the *Activity*.

ll=new MyLocationListener() Then the service creates an instance of a *MyLocationListener* class, called *ll* that is used to communicate with the GPS hardware onboard the device. This component is used to obtain the location of the device once the GPS is able to make a link with the GPS satellites. *MyLocationListener* implements the methods of the native class *LocationListener*.

mPlayer=new MediaPlayer() In the next step the service creates an instance of the class *MediaPlayer*. This instance is used to control the audio system of the device (Appendix A.2 from lines 54 to 57). The application plays an specific sound according to the risk level.

myRunnable=new Runnable() In this step the application creates a *Runnable()* which is a list of tasks performed every 5 seconds (See Appendix A.2 from line 68 to 130). The specified tasks are:

myLocationListener.getCurrentPosition() With this function the service obtains the current position of the device from the *LocationListener*

GetIntersection(myCurrentPosition) In this step the service determines the risk factor for the current location. This process involves the execution of the function *GetIntersection*,which compares the current location with all the elements of the vector *vRiskPolygon* (See Appendix A.2 from line 204 to 235). Each of the elements contained in the vector *vRiskPolygon* is an instance of the class *RiskPolygon* when loaded it executes the internal function *DoesIntersect()* (See Appendix A.3 from line 82 to 136). The result of the function is a boolean that indicates if the current location intersects with the loaded polygon.

Evaluate risk and Sound Alarm Currently the information regarding the avalanche risk is coded using three values 1,2,and 3. Polygons with value 1 are considered safe, while value 2 represent a medium risk and value 3 represents high avalanche risk. In this step the service determines if execute or not an action based on the current risk factor. If the risk factor is one, the user is located in a safe area and therefore not further action is required. If the risk factor is 2 or 3 the media player is activated and a pre-specified sound (wav) is played (See Appendix A.2 from line 85 to 103).

4.4.3 RiskPolygon

Instances of this class are created when the function *myReader()* of the class *RiskEvaluatorService* is executed. This function, reads a GeoJSON file stored in the mobile device. The program creates an instance of the class *RiskPoly-*

gon for each one of the polygons contained in the GeoJSON file. The GeoJSON file stores geometric and alphanumeric information for each polygon in a text based format. The risk data file was created from a PostGIS database. Using a DBMS like PostgreSQL and PostGIS allows easy maintenance and retrieval of the information.

The source code for the class *RiskPolygon* is in Appendix A.3. The list of internal processes required for the creation of an instance of the class *RiskPolygon* are depicted in figure 4.6. In first place we need to input a GeoJSON polygon with the associated *risk* value as a string (See Appendix A.3 line 5). The software retrieves the risk value of the polygon, then it retrieves its geometry, still in text format. By analysing the text the software counts the number of vertices in the polygon, and creates two arrays to store the longitudes and latitudes (See Appendix A.3 lines 33 to 38). Then the software iterates along the arrays to detect the most extreme coordinates that define the bounding box of the polygon (See Appendix A.3 lines 40 and lines 43 to 79).

An important component of the class *RiskPolygon* is its function *DoesIntersect()*, used to evaluate if a given latitude and longitude are located within the polygon or not. This function first evaluates if the point is located within the bounding box of the polygon (See Appendix A.3 lines 86 to 90). If it is outside no further analysis is required, the point is located outside the boundaries of the polygon. If it is located within the bounding box the software counts the intersections of a horizontal line originated in the current location with the segments that compose the polygon. If the number of intersections is odd the current location is within the polygon, if it is even or zero the location is outside (See Appendix A.3 lines 93 to 134).The algorithm is represented in figure 4.7, in this case we have three points p1, p2, and p3. The point p1 is outside the boundaries of the polygon, when a horizontal line is traced from it, it intersects the polygon in two points (even), on the other hand when a horizontal line is traced from p2, it only crosses the segments of the polygon once (odd). The algorithm can be used for complex polygons even when they include islands.

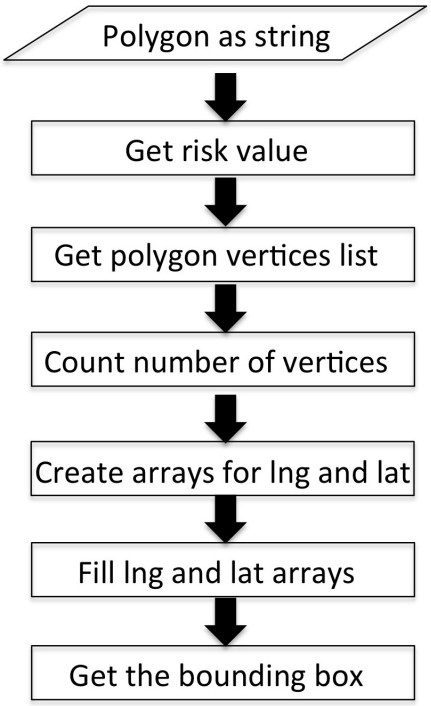

Figure 4.6: Instantiation of the class *RiskPolygon*

4.5 Conclusions

We conducted a field test in Mount Dobratsch, on October of 2011. We tested the application using a tablet Acer Iconia tab a500 and a smartphone Samsung Galaxy Europa i5500. The first device is a 10.1 inches tablet with a multi touch screen, while the second device is smartphone with a 2.8 inches screen, that does not support multi touch gestures. The application successfully worked on both devices, being able to read the risk information in vector format, obtain the user position and compare it with the set of polygons that represent the avalanche risk in the area. The application in both devices monitored the user position and warned the user in case he or she entered into a risk area.

The map included in the application provides the user topographic, tracks and avalanche risk information., relevant for the user decision making process.

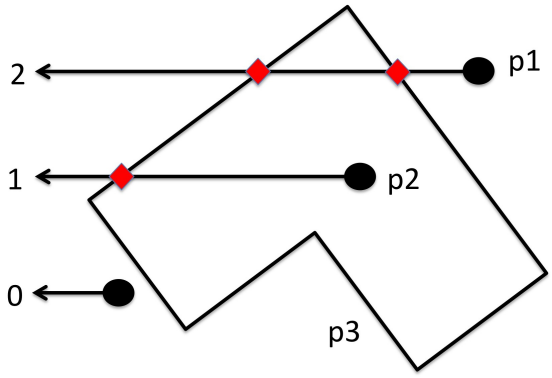

Figure 4.7: Point in polygon algorithm

In the case of the tablet device, the field test showed that the screen device was extremely reflective making it difficult for the user to see the map in certain circumstances. Also we discovered that larger devices are harder to use in outdoors, they are more difficult to shade or put in the correct angle in order to avoid sun reflection. However the multi touch screen proved to be a significant advantage over the smartphone for navigation purposes. I believe a screen size in between with a less reflective screen than the tablet would be a significant improvement on the user experience using the application.

Chapter 5

Conclussions

5.1 Discussion

The growing number of people interested in winter outdoor recreational activities increases the risk of avalanche incidents. Educating people to identify signs of avalanche peril would reduce the risk, however due to the larger numbers of sportsmen education is becoming a difficult task. Even more McCammon (2004) suggest that even expert skiers under certain circunstances fail to recognise inequivocal risks signs and get involved in avalanche incidents [McCammon, 2004].

A survey conducted by Adams(2005) among experts in the field suggests that more frequent bulletins with detailed maps indicating risk areas would increase the safety of the recreationists. However, the cartographic information should be deployed in a form usable by even non expert map users[Adams, 2005].

Currently smartphones and tablets are a suitable platform to deploy easy to use customized/dynamic maps. Nowadays these devices allow the development of third party applications, have onboard GPS which allow them to obtain their location, they can access the internet and have multimedia capabilities.

In the case of an avalanche risk application the network access should not be consider reliable. It might be the case that users find themselves in locations with limited or null network connectivity. An avalanche risk application design should consider this. For the location of the user we use a GNSS (GPS) that does not require the connection to a network. The application is designed to be *device centric*, in the sense that most of the processes run on the device. The application also is designed to be *proactive* in the sense that it request avalanche risk information without the human user intervention when a Internet connection is available.

The application requires to conduct a number of spatial operations that are more efficient using vector data. Because of this reaseon we decided to use GeoJSON, a format that is easy to process using JavaScript and Java, and at the same time is more compact than any XML alternative.

The limited network connectivity forced us to look for map tools that would work in offline mode. The selected tool is OpenLayers a solid JavaScript open source library. OpenLayers allows the development of customized maps using vector and raster data.

There are few operative systems that cover most of the smartphone/tablets market they are: Android, iOS, Symbian and RIM OS, however only the first two are strongly increasing in terms of users, available applications, and reference books which made them the most attractive platforms for the development of an avalanche risk application prototype. In terms of hardware devices using Android or iOS have similar characteristics. We decided to use Android because it is less constraining for application development in terms of fees or deployment approval (Applications in iOS Store require approval from Apple Inc.).

On October 2011, we conducted a field test in Mount Dobratsch using two devices:

- Acer Iconia tab (10.1 inches multitouch screen)
- Samsung Galaxy Europa i5500 (2.8 inches screen)

The application successfully worked in both devices. The map deployed with the prototype application had two layers, the first one was a scanned image of a topographic map and the second was a vector layer containing polygons that represented the different risk levels in the study area. The GPS onboard the devices worked properly and the application managed to determine the risk level of the user position by comparing the GPS information with the risk information in vector format stored on the device.

The field test showed us that glossy screens make it difficult for devices to be used in outdoors. The problem is more serious with larger screens that are more difficult to shade from the sun light. Our test also indicated us that a device with a multi gesture screen capability provides a better user experience than single gesture devices while navigating.

5.2 Limitations

Originally the system was conceptualized including a web feature server that would provide updated risk information in the form of vector data. This part was not implemented. An internet web feature server providing high resolution avalanche risk information does not exist. Although there are plans for the implementation of a data server with those characteristics in the future. For this project we assumed a Web Feature Server providing vector avalanche risk data is in operation. The implementation of Web Feature Server providing avalanche risk information is not technically complicated, although it requires a certain amount of investment in order to keep it operating. Due to the lack of a server we uploaded the information manually into the device before deploying it in the field.

Devices using android are provided by multiple manufacturers with multiple hardware configurations. In our field test we only evaluated two devices. The minimum characteristics the application requires are: onboard GPS and internal sdcard. There is plenty of room for more field tests using devices with different characteristics.

5.3 Future research

Currently smarthphones and tables based on Android have the capability to display 3D data, Google maps already uses this type of spatial information to represent buildings in 3D. A future venue of research might be the use of 3D maps for avalanche risk warning. Google Maps does not use 3D for topographic representation. To my best knowledge at this point there is not an open source alternative equivalent to Google 3D for the 3D representation of spatial data.

Nowadays available sensors in smartphones allow developers to create applications that to a certain degree can detect the user location and perform accordingly. It is natural to expect that future smartphones and tablets will contain more sensors that currently exists only in experimental not main stream devices. Sensors like barometers, temperature, humidity sensors are already being developed in projects like Intel/University of Washington Mobile Sensing Platform (MSP), these sensors for instance allow the device to detect if the user is climbing stairs and in which direction. It is not unrealistic to expect in the future devices with sensors able to asses air quality and pollution [Lane et al., 2010].

In the near future it might be possible to deploy in the mobile devices used by skiers not only a risk data and a warning system, but also use the device to obtain environmental and climatic information. Sensors deployed in the devices would collect climatic data, turning the user into mobile environmental monitor units. When the user connects to the network he/she obtains an updated report for avalanche risk, but also provide the system with climatology measurements taken in the field, with a tag indicating the time and location of the measure. The system would collect the data and use it to improve the risk analysis.

Applications that are able to detect if the user has fall down are already implemented [Dai et al., 2010]. The same approach with certain modifications could be implemented in an application that using the accelerometer detects

when the user is hit by an avalanche. The application automatically executes an acction, for example sends a text message with the last known coordinates, enters into beacon mode producing a distinct sound/ring or activates the bluetooth port and starts emitting a signal.

Bibliography

[AAA, 2010] AAA (2010). American Avalanche Association website. http://www.americanavalancheassociation.org.

[Ableson et al., 2011] Ableson, W. F., Sen, R., and King, C. (2011). *Android in action*. Manning.

[Adams, 2005] Adams, L. (2005). Supporting good decicions. *The Avalanche Review*, 23(3):12–17.

[Aitchison, 2011] Aitchison, A. (2011). The Google Maps/Bing Maps Spherical Mercator Projection. http://alastaira.wordpress.com/2011/01/23/the-google-maps-bing-maps-spherical-mercator-projection/.

[Andrews, 2007] Andrews, C. J. (2007). Emerging technology:AJAX and GeoJSON. *Directions Magazine*.

[AppleInc., 2012] AppleInc. (2012). ios developer library. http://developer.apple.com.

[Bailey and Chen, 2011] Bailey, J. E. and Chen, A. (2011). The role of virtual globes in geoscience. *Computers & Geosciences*, 37:1–2.

[Bajarin, 2011] Bajarin, T. (2011). Is the Apple ipad a tablet? http://www.pcmag.com/article2/0,2817,2392263,00.asp.

[Bibby, 2010] Bibby, A. (2010). The rapid development of Tablet computing. www.tcodevelopment.se.

[Brugger et al., 2001] Brugger, H., Durrer, B., Adler Kastner, L., Falk, M., and Tschirky, F. (2001). Field management of avalanche victims. *Resuscitation*, 51(1):7–15.

[Butler, 2011] Butler, M. (2011). Android:Changing the Mobile Landscape. *Pervasive Computing, IEEE*, 10(1):4–7.

[Cassavoy, 2011] Cassavoy, L. (2011). What makes a smartphone smart? http://cellphones.about.com/od/smartphonebasics/a/what_is_smart.htm.

[Cojocaru et al., 2009] Cojocaru, S., Barsan, E., Bratinca, G., and Arsenie, P. (2009). GPS-GLONASS-GALILEO: A dynamical comparison. *Journal of Navigation*, 62(1):135–150.

[Crockford, 2006] Crockford, D. (2006). JSON: The fat-free alternative to XML. www.json.org/fatfree.html.

[Dai et al., 2010] Dai, J., Xiaole, B., Yang, Z., Shen, Z., and Xuan, D. (2010). Mobile phone–based pervasive fall detection. *Personal and ubiquitous computing*, 14(7):633–643.

[Davis and Rocchio, 2011] Davis, J. and Rocchio, R. A. (2011). Mobile: Letting go of the device and building for innovation. *Educause Review Magazine*, 46(2).

[de la Beaujardiere, 2006] de la Beaujardiere, J. (2006). OpenGIS Web Map Server Implementation Specification. Technical report, Open Geospatial Consortium Inc.

[Distimo, 2011] Distimo (2011). The battle for the most content and the emerging tablet market. www.distimo.com.

[Eckerstorfer, 2008] Eckerstorfer, M. (2008). Cartographic analysis of avalanche hazard maps. In *6th ICA Mountain Cartography Workshop Mountain Mapping and Visualisation.*

[Fendelman, 2011] Fendelman, A. (2011). How are cell phones different from smartphones? http://cellphones.about.com/od/coveringthebasics/ - qt/-cellphonesvssmartphones.htm.

[Fonseca et al., 2011] Fonseca, P., Colls, M., and Casanovas, J. (2011). A novel model to predict a slab avalanche configuration using $m : n - ca^k$ cellular automata. *Computers, Environment and Urban Systems*, 35:12–24.

[Frankel, 2011] Frankel, A. (2011). In Oracle v. Google, jury will hear from court-appointed expert. Thomson Reuters. http://newsandinsight.thomsonreuters.com/Legal/News/ViewNews.aspx.

[Furman et al., 2010] Furman, N., Shooter, W., and Schumann, S. (2010). The Roles of Heuristics, Avalanche Forecast, and Risk Propensity in the Decision Making of Backcountry Skyers. *Leisure Sciences*, 32:453–469.

[Gartner, 2011a] Gartner (2011a). News room. http://www.gartner.com/it/page.jsp?id=1764714.

[Gartner, 2011b] Gartner (2011b). News room. http://www.gartner.com/it/page.jsp?id=1848514.

[Google, 2011a] Google (2011a). Android developers. website. http://developer.android.com/index.html.

[Google, 2011b] Google (2011b). Google Maps JavaScript API V3. http://code.google.com/apis/maps/documentation/javascript/tutorial.html.

[Google, 2011c] Google (2011c). Introduction of usage limits to the Maps API. http://googlegeodevelopers.blogspot.com/2011/10/introduction–of–usage–limits–to–maps.html.

[Google, 2011d] Google (2011d). KML tutorial. http://code.google.com/apis/kml/.

[Grossman, 2000] Grossman, W. M. (2000). Wireless warrior. http://www.salon.com/2000/05/15/colly$_m yers$/.

[Hazzard, 2011] Hazzard, E. (2011). *Open Layers 2.10*. Packt Publishing.

[Holdener III, 2011] Holdener III, A. T. (2011). *HTML5 Geolocation*. O'Reilly.

[Holler, 2007] Holler, P. (2007). Avalanche hazards and mitigation in Austria. *Natural Hazards*, 43:81–101.

[Kamel et al., 2010] Kamel, M. N., Warren, J., Gong, J., and Yue, P. (2010). Web GIS in practice VIII: HTML5 and the canvas element for interactive online mapping. *International Journal of Health Geographics*, 9:14.

[Kulawiak et al., 2010] Kulawiak, M., Prospathopoulos, A., Perivoliotis, L., Luba, M., Kioroglou, S., and Stepnowski, A. (2010). Interactive visualization of marine pollution monitoring and forecasting data via a Web-based GIS. *Computers & Geosciences*, 36(8):1069–1080.

[Lake, 2000] Lake, R. (2000). Making Maps with Geography Markup Language (GML). http://spatialnews.geocomm.com/whitepapers/MakingMapsInGML2.pdf.

[Lake, 2004] Lake, R. (2004). Introduction to GML. http://www.w3.org/Mobile/posdep/GMLIntroduction.html.

[Lane et al., 2010] Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., and Campbell, A. T. (2010). A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150.

[LeMay, 2005] LeMay, R. (2005). Google mapper:Take browsers to the limit. *cnet*.

[Liu et al., 2011] Liu, C., Zhu, Q., Holroyd, K. A., and Seng, E. K. (2011). Status and trends of mobile health applications for iOS devices: A developer's perspective. *The Journal of Systems and Software*, 84(11):2022–2033.

[Longley et al., 2005] Longley, P., Goodchild, M., Maguire, D., and Rhind, D. (2005). *Geographic Information Systems and Science*. Wiley.

[Mabrouk, 2008] Mabrouk, M. (2008). OpenGIS Location Services (OpenLS): Core Services. Technical report, Open Geospatial Consortium Inc.

[McCammon, 2004] McCammon, I. (2004). Heuristic traps in recreational avalanche accidents: Evidence and implications. *The Avalanche Review*, 22(3):11–13.

[Meng and Reichenbacher, 2005] Meng, L. and Reichenbacher, T. (2005). Map-based mobile services. In Meng, L., Zipf, A., and Reichenbacher, T., editors, *Map–based Mobile Service:Theories, Methods and Implementations*, chapter 1. Springer.

[OGC, 2011] OGC (2011). OGC KML. website. http://www.opengeospatial.org/standards/kml.

[Olivier, 2006] Olivier, S. (2006). Moral dilemmas of participation in dangerous leisure activities. *Leisure Studies*, 25(1):95–109.

[Pace et al., 1995] Pace, S., Frost, G., Lachow, I., Frelinger, D., Fossum, D., Wassem, D. K., and Pinto, M. (1995). *The Global Positioning System Assessing National Policies*. Rand.

[Parziale et al., 2006] Parziale, L., Britt, D. T., Davis, C., Forrester, J., Liu, W., Matthews, C., and Rosselot, N. (2006). *TCP/IP Tutorial an Technical Overview*. IBM.

[pcmag.com, 2010] pcmag.com (2010). Tablet computer. http://www.pcmag.com/encyclopedia_term.

[Pendleton, 2010] Pendleton, C. (2010). The world according to Bing. *Computer Graphics and Applications,IEEE*, 30(4):15–17.

[Pimpler, 2006] Pimpler, E. (2006). Introduction to Developing with Google Maps. http://www.directionsmag.com/articles/introduction–to–developing–with–google–maps/123188.

[Ratliff, 2007] Ratliff, E. (2007). Google maps is changing the way we see the world. *Wired Magazine*, (15).

[Reuters, 2009] Reuters (2009). Six killed in Austrian Alps avalanche. The Independent 5/3/2009. http://www.independent.co.uk/news/world/europe/six-killed-in-austrian-alps-avalanche-1678308.html.

[Revuelto Luque, 2011] Revuelto Luque, R. M. (2011). The Use of digital image mapping and resource training as secondary. Some details about Google Earth. *Boletin de la Asociacion de Geografos Espa noles*, (55):417–422.

[Schutzberg, 2005] Schutzberg, A. (2005). KML gets two thumbs up from file format experts. *Directions Magazine*.

[Shek, 2010] Shek, S. (2010). Next-generation location-based services for mobile devices. CSC_Grant_2010_Next_Generation_Location_Based_Services_for_Mobile_Devices.pdf.

[Shi et al., 2009] Shi, W., Kwan, K., Shea, G., and Cao, J. (2009). A dynamic data model for mobile GIS. *Computers & Geosciences*, 35:2210–2221.

[Shin, 2010] Shin, S. (2010). Introduction to JSON (java script object notation. Presentation www.javapassion.com.

[Steiniger et al., 2004] Steiniger, S., Neun, M., and Edwards, A. (2004). Foundations of Location Based Services. http://www.spatial.cs.umn.edu/Courses/Fall07/8715/papers/IM7_steiniger.pdf.

[Straus et al., 2007] Straus, S. G., Bikson, T. K., Balkovich, E., and Pane, J. F. (2007). Mobile Technology and Action Teams: Assessing BlackBerry Use in Law Enforcement Units, RAND Corporation. Rand Corporation Working paper.

[Sutter, 2010] Sutter, J. D. (2010). What is a tablet, anyway? http://articles.cnn.com/2010-01-09/tech/ces.tablet.computers_1_tablet-touch-screen-keyboard?_s=PM:TECH.

[Tase, 2005] Tase, J. (2005). Backcountry recreationists risk of exposure to avalanche hazards. *The Avalanche Review*, 24(2):7.

[Turner, 2010] Turner, R. (2010). Avalanches in the Alps kill several people. DW-World.de Deutsche Welle. http://www.dw-world.de/dw/article/0,,5217280,00.htm.

[UAC, 2011] UAC (2011). Utah avalanche center education website. webpage. http://utahavalanchecenter.org/education/.

[Vretanos, 2011] Vretanos, P. A. (2011). OpenGIS Web Geature Service 2.0 Interface Standard. Technical report, Open Geospatial Consortium Inc.

[W3C, 2011] W3C (2011). HTML5 website. http://www.w3.org/TR/html5/.

[Wauters, 2011] Wauters, R. (2011). Apple:100 Million Downloads from Mac App Store in Less than one year. http://techcrunch.com/2011/12/12.

Appendix A

Java Classes

A.1 AvalancheUI.java

```
1  package com.avalanche5;
2
3  import java.text.DecimalFormat;
4  import android.app.Activity;
5  import android.content.BroadcastReceiver;
6  import android.content.Context;
7  import android.content.Intent;
8  import android.content.IntentFilter;
9  import android.os.Bundle;
10 import android.util.Log;
11 import android.view.Menu;
12 import android.view.MenuInflater;
13 import android.view.MenuItem;
14 import android.webkit.WebView;
15 import android.widget.TextView;
16
17 public class AvalancheUI extends Activity {
18    private static final String TAG = "BroadcastTest";
19    private Intent intent; //for communication purposes
```

```
        between the Activity and the Service
20      private String st_myLocation;
21      private WebView webview;//This item will be used to
            display the map
22      private TextView myCurrLoc;
23      private DecimalFormat df = new
            DecimalFormat("#.#####");
24
25      @Override
26      public void onCreate(Bundle savedInstanceState)
27      {
28      super.onCreate(savedInstanceState);
29      setContentView(R.layout.main);
30      //we define the new Intent as a link between this
            Activity and the Service called
            RiskEvaluatorService.class
31      intent = new Intent(this,
            RiskEvaluatorService.class);
32      //we define a new WebView
33      webview= (WebView)
            findViewById(R.id.webview_compontent);
34      //we change the settings of the WebView in order to
            make it capable of running JavaScript
35      webview.getSettings().setJavaScriptEnabled(true);
36      //we indicate the URL of the webpage that displays
            the map, that is locally stored on the device
37      webview.loadUrl(
            "file://mnt/sdcard/avalanche_map/visor.html");
38      myCurrLoc=(TextView)findViewById(R.id.curr_loc);
39      myCurrLoc.setText("Longitud,Latitude");
40      }
41
42      private BroadcastReceiver broadcastReceiver =
43      new BroadcastReceiver() {
```

```
44          @Override
45          //used for communicate with the service, using
               the previously created Intent
46          public void onReceive(Context context, Intent
               intent)
47          {
48          updateUI(intent);
49          }
50      };
51
52      @Override
53      public void onResume()
54      {
55          super.onResume();
56      }
57
58      @Override
59      public void onPause()
60      {
61          super.onPause();
62      }
63
64      private void updateUI(Intent intent)
65      {
66      //set of commands, information that are received
           from the Service
67      String counter = intent.getStringExtra("counter");
68      String time = intent.getStringExtra("time");
69      double coord_lng=intent.getDoubleExtra("coord_lng",
           -9);
70      double coord_lat=intent.getDoubleExtra("coord_lat",
           -9);
71      //We use this comand to put a marker on the map in
           the user current location
```

```
72    webview.loadUrl("javascript:addLocMarker
          ("+coord_lng+","+coord_lat+")");
73    //for testing purposes to test the communication
          between the service and the activity
74    TextView txtDateTime = (TextView)
          findViewById(R.id.txtDateTime);
75    TextView txtCounter = (TextView)
          findViewById(R.id.txtCounter);
76    TextView txtCurrLoc=(TextView)
          findViewById(R.id.curr_loc);
77    txtDateTime.setText(time);
78    txtCounter.setText(counter);
79    txtCurrLoc.setText("Current Location: " +
          df.format(coord_lng) + "," +
          df.format(coord_lat));
80    }
81
82    //the menu is stored in a XML file on the device
83    //these lines of code are used to read the XML and
          transform
84    //it into the menus
85    @Override
86    public boolean onCreateOptionsMenu(Menu menu)
87    {
88    MenuInflater inflater = getMenuInflater();
89    inflater.inflate(R.menu.avalanche_menu, menu);
90    return true;
91    }
92
93    @Override
94    public boolean onOptionsItemSelected(MenuItem item)
95    {
96    // Handle item selection
97    switch (item.getItemId())
```

```
98          {
99          case R.id.start_service:
100             {
101             //to start the service
102             st_myLocation="start service";
103             //we start the service associated with the
                    intent
104             startService(intent);
105             //we register the link in order to be able to
                    listen information broadcasted by the
                    service RiskEvaluatorService
106             registerReceiver(broadcastReceiver, new
                    IntentFilter
                    (RiskEvaluatorService.BROADCAST_ACTION));
107             return true;
108             }
109
110         case R.id.stop_service:
111             {
112             st_myLocation="stop service";
113             unregisterReceiver(broadcastReceiver);
114             //we stop the service associated to the
                    Intent (RiskEvaluatorService)
115             stopService(intent);
116             return true;
117             }
118
119         case R.id.clear_map:
120             {
121             st_myLocation="clear map";
122             //to turn on or off the risk data on the map
123             webview.loadUrl("javascript:layerTurnOffOn()");
124             return true;
125             }
```

```
126         default:
127             return super.onOptionsItemSelected(item);
128         }
129     }
130 }
```

A.2 RiskEvaluatorService.java

```
1  package com.avalanche5;
2
3  import java.io.File;
4  import java.io.FileInputStream;
5  import java.io.IOException;
6  import java.util.Date;
7  import java.util.Scanner;
8  import java.util.Vector;
9  import android.app.Service;
10 import android.content.Context;
11 import android.content.Intent;
12 import android.content.res.AssetFileDescriptor;
13 import android.location.Location;
14 import android.location.LocationListener;
15 import android.location.LocationManager;
16 import android.media.MediaPlayer;
17 import android.os.Bundle;
18 import android.os.Handler;
19 import android.os.IBinder;
20 import android.util.Log;
21
22 public class RiskEvaluatorService extends Service{
23 private static final String TAG = "BroadcastService";
24 public static final String BROADCAST_ACTION =
     "com.websmithing.broadcasttest.displayevent";
25 private final Handler handler = new Handler();
26 private LocationManager lm;
27 private MyLocationListener ll;
28 public double[] myCurrentPosition=new double[2];
29 private Vector vRiskPolygons;
30 private double[] infoCurrentPolygon=new double[2];
31 private MediaPlayer mPlayer;
```

```
32 private AssetFileDescriptor soundRiskLevel2;
33 private AssetFileDescriptor soundRiskLevel3;
34
35 Intent intent;
36 int counter = 0;
37
38 @Override
39 public void onCreate()
40 {
41    super.onCreate();
42    //we create a new vector vRiskPolygons that will
         store the risk polygons
43    vRiskPolygons=new Vector();
44    //to execute the function that reads the risk
         polygon data
45    myReader();
46    //to create the link between the Service(background
         processes) and the Activity (user interface)
47    intent = new Intent(BROADCAST_ACTION);
48    //lm and ll to communicate with the onboard GPS
         hardware
49    lm = (LocationManager)
         getSystemService(Context.LOCATION_SERVICE);
50    ll = new MyLocationListener();
51    //we request updates from the GPS hardware every
         5000 miliseconds
52    lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,
         5000, 5, ll);
53    //we create two ringtones for risk levels 2 and 3
54    soundRiskLevel2 =
         getResources().openRawResourceFd(R.raw.ding);
55    soundRiskLevel3 =
         getResources().openRawResourceFd(R.raw.ding_ling);
56    to access the media player capabilities of the
```

```
        device
57      mPlayer=new MediaPlayer();
58  }
59
60  @Override
61  public void onStart(Intent intent, int startId)
62  {
63      handler.removeCallbacks(myRunnable);
64      handler.postDelayed(myRunnable, 5000); // we repeat
            myRunnable every 5000 miliseconds
65  }
66
67  // the set of commnands that repeat every 5 seconds
68  private Runnable myRunnable = new Runnable()
69  {
70      public void run()
71      {
72      //obtain current location
73      myCurrentPosition=((MyLocationListener)
            ll).getCurrentPosition();
74      DisplayLoggingInfo();
75
76      try
77      {
78          //the initial value of both latitude and
                longitude is 0
79          //different values would indicate a position
                obtained from the GPS hardware
80          if((myCurrentPosition[0]!=0)
                &&(myCurrentPosition[1]!=0))
81          {
82              // we proceed to test the current location
                    using the function GetIntersection()  the
                    result of the test is stored in the array
```

```
                infoCurrentPolygon, the value [1] of the
                array contain the risk level of the
                polygon that intersects with the current
                location, in case there is any.
83              infoCurrentPolygon=GetIntersection
                (myCurrentPosition[0],
                myCurrentPosition[1]);
84              // if the risk level is 1 means no risk we
                stop any sound
85              if (infoCurrentPolygon[1]==1)
86              {
87                  mPlayer.reset();
88              }
89              // if the risk level is 2 or 3 we play the
                appropiate alarm
90              if (infoCurrentPolygon[1]==2)
91              {
92                  mPlayer.reset();
93                  mPlayer.setDataSource
                        (soundRiskLevel2.getFileDescriptor(),
                        soundRiskLevel2.getStartOffset(),
                        soundRiskLevel2.getLength());
94                  mPlayer.prepare();
95                  mPlayer.start();
96              }
97              if (infoCurrentPolygon[1]==3)
98              {
99                  mPlayer.reset();
100                 mPlayer.setDataSource
                        (soundRiskLevel3.getFileDescriptor(),
                        soundRiskLevel3.getStartOffset(),
                        soundRiskLevel3.getLength());
101                 mPlayer.prepare();
102                 mPlayer.start();
```

```
103             }
104         }
105         else
106         {
107             Log.d("SERVICE RUNNABLE", "with fake coords
                    ");
108         }
109     }
110
111     catch (IllegalArgumentException e)
112     {
113         // TODO Auto-generated catch block
114         e.printStackTrace();
115     }
116     catch (IllegalStateException e)
117     {
118         // TODO Auto-generated catch block
119         e.printStackTrace();
120     }
121     catch (IOException e)
122     {
123         // TODO Auto-generated catch block
124         e.printStackTrace();
125     }
126     handler.postDelayed(this, 5000); // 5 seconds
127     }
128 };
129
130      // the information that is sent to the user
            service (user interface)
131 private void DisplayLoggingInfo()
132 {
133     Log.d(TAG, "entered DisplayLoggingInfo");
134     intent.putExtra("time", new
```

```
        Date().toLocaleString());
135     intent.putExtra("counter",
            String.valueOf(++counter));
136     intent.putExtra("coord_lng", myCurrentPosition[0]);
137     intent.putExtra("coord_lat", myCurrentPosition[1]);
138     sendBroadcast(intent);
139 }
140
141 @Override
142 public IBinder onBind(Intent intent)
143 {
144     return null;
145 }
146
147 @Override
148 public void onDestroy()
149 {
150     handler.removeCallbacks(myRunnable);
151     super.onDestroy();
152 }
153
154     //To access a file stored onboard that contains the
            risk polygons in vector format
155 private void myReader()
156 {
157     String risk_polys="";
158     String sFileName="";
159     String sLine="";
160
161     // we create a file indicating the path where the
            file is located
162     File rFile=new
            File("/sdcard/avalanche_map/geojson/risk.js");
163     //we test if the file exists and is accessible
```

```
164     if(rFile.exists() && rFile.canRead())
165     {
166         //we create a file input stream to contain the
              info stored in the file
167         FileInputStream fis=null;
168         try
169         {
170             // now fis contains the info of stored in
                  rFile
171             fis=new FileInputStream(rFile);
172             // we create a scanner in order to be able to
                  read one line at the time
173             Scanner scanner = new Scanner(new
                  FileInputStream(rFile), "UTF-8");
174             // we iterate along the scanner one line at
                  the time until no more lines exist
175             while (scanner.hasNextLine())
176             {
177                 //sLine has the value of the current line
178                 sLine=scanner.nextLine();
179                 // the file contains one feauture per
                      line, we test if this line has the word
                      [geometry], indicating that the line
                      has a spatial feature
180                 if(sLine.contains("\"geometry\":"))
181                 {
182                     //if the line contains a spatial
                          feature we create an instance of the
                          class RiskPolygon with this
                          information. We send the string with
                          the feature information to the class
                          RiskPolygon().
183                     RiskPolygon myPolygon=new
                          RiskPolygon(sLine);
```

```
184                 //we store the instance into the vector
                        vRiskPolygons
185                 vRiskPolygons.add(myPolygon);
186             }
187         }
188         //Message to be displayed when all the
                polygons have been parsed
189         Log.d("PARSING POLYGONS:", "done " +
                vRiskPolygons.size());
190         scanner.close();
191       }
192
193       //to catch Input/Output errors
194       catch (IOException e)
195       {
196         // TODO Auto-generated catch block
197         Log.d("READER", "IOException");
198         e.printStackTrace();
199       }
200     }
201 }
202
203     // this function uses as input a position, testLng,
          testLat and compares it with all the risk
          polygons.
204 private double[] GetIntersection(double testLng,
      double testLat)
205 {
206     RiskPolygon tempoRiskPolygon;
207     double[] infoPolygon=new double[2];
208     int iCountVector=0;
209     int countIntersections=0;
210     // we iterate along all the polygons stored in
          vector vRiskPolygons
```

```
211     for (iCountVector=0;
            iCountVector<vRiskPolygons.size();
            iCountVector++)
212     {
213         // tempoRiskPolygon is the current polygon to
                test
214         tempoRiskPolygon=(RiskPolygon)vRiskPolygons.get(iCountVector);
215         // tempoRiskPolygon is an instance of the class
                RiskPolygon
216         // the class RiskPolygon has a function called
                DoesIntersect()
217         // it tests any given position and responds if
                it is outside or inside
218         if (tempoRiskPolygon.DoesIntersect (testLng,
                testLat))
219         {
220             countIntersections++;
221             infoPolygon[0]=tempoRiskPolygon.dID;
222             infoPolygon[1]=tempoRiskPolygon.dRisk;
223         }
224     }
225     // countIntersections must be 1 or 0
226     // it could be more than 1 if there are overlaping
            polygons
227     // if it is 0 the point does not intersect any
            polygon in the dataset
228     if(countIntersections==0)
229     {
230         Log.d("INTERSECTION RESULT", "No intersection
                found");
231         infoPolygon[0]=0;
232         infoPolygon[1]=0;
233     }
234     return infoPolygon;
```

```
235 }
236
237 class MyLocationListener implements LocationListener
238 {
239     private Location x_location;
240     private boolean status_location=false;
241     private String msgLocationListener="";
242
243     @Override
244     public void onLocationChanged(Location location)
245     {
246         if (location != null)
247         {
248             Log.d("LOCATION CHANGED",
                    location.getLatitude()+ "");
249             Log.d("LOCATION CHANGED",
                    location.getLongitude()+ "");
250             status_location=true;
251             x_location=location;
252         }
253     }
254
255     @Override
256     public void onProviderDisabled(String provider)
257     {
258         msgLocationListener="provider disabled";
259         Log.d("LOCATION CHANGED",msgLocationListener);
260     }
261
262     @Override
263     public void onProviderEnabled(String provider)
264     {
265         msgLocationListener="Provider Enabled";
266         Log.d("LOCATION CHANGED",msgLocationListener);
```

```
267     }
268
269     @Override
270     public void onStatusChanged(String provider,
271     int status, Bundle extras)
272     {
273         msgLocationListener="Status changed: " + status;
274         Log.d("LOCATION CHANGED",msgLocationListener);
275     }
276
277     public double[] getCurrentPosition()
278     {
279         double [] myPosition = new double[2];
280         if (status_location==true)
281         {
282             myPosition[0]=x_location.getLongitude();
283             myPosition[1]=x_location.getLatitude();
284         }
285         else
286         {
287             msgLocationListener="not available";
288             myPosition[0]=0;
289             myPosition[1]=0;
290         }
291         Log.d("LOCATION GET
                POSITION",msgLocationListener);
292         return myPosition;
293     }
294 }
295 }
```

A.3 RiskPolygon.java

```
1  package com.avalanche5;
2
3  public class RiskPolygon
4  {
5     public RiskPolygon(String polygon_as_string)
6     {
7        //polygon_as_string contains the info related to
            one polygon in JSON format,
8        //this class process the info and creates a Java
            representation of the polygon that can be
            used by the avalanche application
9        st_Polygon=polygon_as_string;
10       //we know the JSON structure of the info, so we
            identify the position of certain key elements
            in the string
11       int
            iPropertiesBegin=st_Polygon.indexOf("\"properties\":");
12       sProperties=st_Polygon.substring
            (iPropertiesBegin, st_Polygon.indexOf("}",
            iPropertiesBegin));
13       dID=0;
14       //grid_code is the name of the column that
            contains the risk level in our dataset
15       //we identify the position of "grid_code" and
            then get the value for this element
16       int
            iRiskBegin=sProperties.indexOf("\"grid_code\":");
17       String sRisk=sProperties.substring
            (iRiskBegin+12);
18       dRisk=Double.parseDouble(sRisk);
19       //the vertex values of the polygon are
            associated to the element "coordinates"
```

```
20        //we identify the position of the string
             "coordinates" and get the value of the
             vertices
21        int iCoordinatesBegin= st_Polygon.indexOf
             ("\"coordinates\":");
22        sCoordinates=st_Polygon.substring
             (iCoordinatesBegin+14,
             st_Polygon.indexOf("}", iCoordinatesBegin));
23        sCoordinates=sCoordinates.replace ('[', ' ');
24        sCoordinates=sCoordinates.replace (']', ' ');
25        sVertex=sCoordinates.split(",");
26        icount=0;
27        //first we define the lenght of the arrays by
             counting the number of vertices
28        iVertexCount=(sVertex.length+1)/2;
29        listLngs=new double[iVertexCount];
30        listLats=new double[iVertexCount];
31        //then we store the value of the vertices in two
             arrays one for
32        //latitude and one for longitude
33        for (ic=0;ic<sVertex.length;ic=ic+2)
34        {
35           listLngs[icount]=Double.parseDouble
                (sVertex[ic]);
36           listLats[icount]=Double.parseDouble
                (sVertex[ic+1]);
37           icount++;
38        }
39        //getBB() internal function to obtain the
             bounding box for this polygon
40        getBB();
41     }
42
43     private void getBB()
```

```
44    {
45        double xMinLng;
46        double xMinLat;
47        double xMaxLng;
48        double xMaxLat;
49
50        xMinLng=listLngs[0];
51        xMaxLng=listLngs[0];
52
53        xMinLat=listLats[0];
54        xMaxLat=listLats[0];
55        //we iterate along the arrays listLngs and
              listLats and find extreme values
56        for (icount=0;icount<listLngs.length;icount++)
57        {
58            if (listLngs[icount]<xMinLng)
59            {
60                xMinLng=listLngs[icount];
61            }
62            if (listLngs[icount]>xMaxLng)
63            {
64                xMaxLng=listLngs[icount];
65            }
66            if (listLats[icount]<xMinLat)
67            {
68                xMinLat=listLats[icount];
69            }
70            if (listLats[icount]>xMaxLat)
71            {
72                xMaxLat=listLats[icount];
73            }
74        }
75        minLng=xMinLng;
76        maxLng=xMaxLng;
```

```
77          minLat=xMinLat;
78          maxLat=xMaxLat;
79      }
80
81      //to test if the any given coordinates (double
            CoordLng, double CoordLat) are within or outside
            this polygon
82  public boolean DoesIntersect(double CoordLng, double
        CoordLat)
83  {
84      boolean bIntersection=false;
85      //first we test if the testing position are inside
            the bounding box, if the position is outside the
            bounding box, no further test is performed
86      if((CoordLng<minLng) || (CoordLng>maxLng) ||
            (CoordLat<minLat) || (CoordLat>maxLat))
87      {
88          bIntersection=false;
89          return bIntersection;
90      }
91      //if the point to test is located inside the
            bounding box we proceed to count the
            intersections between a horizontal line from the
            point to the Y axis. If there are an odd number
            of intersections the point is inside otherwise
            the point is outside.
92      else
93      {
94          double lng1=listLngs[0];
95          double lat1=listLats[0];
96          double lng2=0;
97          double lat2=0;
98          double dXinters=0;
99          double counter=0;
```

```
100         for (int i=1;i<iVertexCount;i++)
101         {
102             lng2=listLngs[i];
103             lat2=listLats[i];
104             if(CoordLat>Math.min(lat1, lat2))
105             {
106                 if(CoordLat<=Math.max(lat1, lat2))
107                 {
108                     if(CoordLng<=Math.max(lng1,lng2))
109                     {
110                         if(lat1!=lat2)
111                         {
112                             dXinters=(CoordLat-lat1)*
113                             (lng2-lng1)/(lat2-lat1)+lng1;
114                             if (lng1 == lng2 || CoordLng <=
                                    dXinters)
115                             {
116                                 counter++;
117                             }
118                         }
119                     }
120                 }
121             }
122             lng1=lng2;
123             lat1=lat2;
124         }
125
126         if (counter % 2 == 0)
127         {
128             bIntersection=false;
129         }
130         else
131         {
132             bIntersection= true;
```

```
133         }
134     }
135     return bIntersection;
136 }
137 
138 public double minLng;
139 public double minLat;
140 public double maxLng;
141 public double maxLat;
142 public double[] listLats;
143 public double[] listLngs;
144 public int iVertexCount;
145 private String st_Polygon;
146 private String[] sVertex;
147 private int ic=0;
148 private int icount=0;
149 private String sProperties="";
150 private String sCoordinates="";
151 private String sID="";
152 public double dRisk=0;
153 public double dID=0;
154 }
```

Appendix B

Visor module

B.1 HTML

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Visor Y</title>
5         <meta http-equiv="Content-Type"
              content="text/html; charset=utf-8" />
6         <meta name="viewport" content="
              width=device-width, initial-scale=1.0,
7         maximum-scale=1.0, user-scalable=0">
8         <meta name="apple-mobile-web-app-capable"
              content="yes">
9         <link rel="stylesheet" href="style.mobile.css"
              type="text/css">
10        <script
              src="OpenLayers-2.11/lib/OpenLayers.js?mobile">
              </script>
11        <script
              src="OpenLayers-2.11/lib/Firebug/firebug.js">
              </script>
```

```
12
13          <script src="geojson/risk.js"></script>
14
15          <script src="visor_z.js"></script>
16
17          <style>
18              html, body {
19                  margin  : 0;
20                  padding : 0;
21                  height  : 100%;
22                  width   : 100%;
23              }
24              @media only screen and (max-width: 600px) {
25                  html, body {
26                      height  : 117%;
27                  }
28              }
29              #map {
30                  width    : 100%;
31                  position : relative;
32                  height   : 100%;
33              }
34              .olControlAttribution {
35                  position       : absolute;
36                  font-size      : 10px;
37                  bottom         : 0 !important;
38                  right          : 0 !important;
39                  background     : rgba(0,0,0,0.1);
40                  font-family    : Arial;
41                  padding        : 2px 4px;
42                  border-radius  : 5px 0 0 0;
43              }
44              div.olControlZoomPanel
                    .olControlZoomInItemInactive,
```

```
45              div.olControlZoomPanel
                    .olControlZoomOutItemInactive {
46                  background: rgba(0,0,0,0.2);
47                  position: absolute;
48              }
49              div.olControlZoomPanel
                    .olControlZoomInItemInactive {
50                  border-radius: 5px 5px 0 0;
51              }
52              div.olControlZoomPanel
                    .olControlZoomOutItemInactive {
53                  border-radius: 0 0 5px 5px ;
54                  top: 37px;
55              }
56              div.olControlZoomPanel
                    .olControlZoomOutItemInactive:after ,
57              div.olControlZoomPanel
                    .olControlZoomInItemInactive:after{
58                  font-weight: bold;
59                  content    : '+';
60                  font-size  : 36px;
61                  padding:   7px;
62                  z-index: 2000;
63                  color      : #fff;
64                  line-height: 1em;
65              }
66              div.olControlZoomPanel
                    .olControlZoomOutItemInactive:after{
67                  content: '';
68                  line-height: 0.9em;
69                  padding: 0 8px;
70              }
71              div.olControlZoomPanel
                    .olControlZoomToMaxExtentItemInactive {
```

```
72                  display: none;
73              }
74              #title, #shortdesc {
75                  display: none;
76              }
77          </style>
78      </head>
79      <body>
80          <div id="map"></div>
81
82       <h1 id="title">Visor Y</h1>
83          <div id="tags">
84
85          </div>
86
87          <script>
88              init();
89          </script>
90      </body>
91  </html>
```

B.2 JavaScript

```
1  //initialize map when page ready
2 var map;
3 var markers = new OpenLayers.Layer.Markers( "Markers"
    );
4
5 var size = new OpenLayers.Size(21,25);
6 var offset = new OpenLayers.Pixel(-(size.w/2),
    -size.h);
7 var  icon = new
    OpenLayers.Icon('img/mobile-loc.png',size,offset);
8 var marker;
9
10 // Get rid of address bar on iphone/ipod
11 var fixSize = function() {
12     window.scrollTo(0,0);
13     document.body.style.height = '100%';
14     if
         (!(/(iphone|ipod)/.test(navigator.userAgent.toLowerCase())))
15     {
16          if (document.body.parentNode)
17          {
18               document.body.parentNode.style.height =
                   '100%';
19          }
20     }
21 };
22 setTimeout(fixSize, 700);
23 setTimeout(fixSize, 1500);
24
25 var options = {numZoomLevels: 32};
26
27 var graphic = new OpenLayers.Layer.Image( 'sat1',
```

```
    'dobratsch/sat_img1.png',
28 new OpenLayers.Bounds(13.17867378, 46.3829557766,
    13.9900207874, 46.9191297766),
29 new OpenLayers.Size(687, 454),options );
30
31 // to set the style of the layers that contain the
    risk polygons
32 var st_risk1=new
    OpenLayers.Style({'strokeColor':'#000000',
    'fillColor':'#3cff3c','fillOpacity':0.75,'
    strokeWidth':0.25});
33 var st_risk2=new
    OpenLayers.Style({'strokeColor':'#000000',
    'fillColor':'#ffff3c','fillOpacity':0.75,'
    strokeWidth':0.25});
34 var st_risk3=new
    OpenLayers.Style({'strokeColor':'#000000',
    'fillColor':'#ff3c3c','fillOpacity':0.75,'
    strokeWidth':0.25});
35
36 // we create a single layer for each of the risk levels
37 var geojson_risk1=new OpenLayers.Format.GeoJSON();
38 var vector_layer_risk1=new
    OpenLayers.Layer.Vector("Risk 1",
    {styleMap:st_risk1});
39
40 var geojson_risk2=new OpenLayers.Format.GeoJSON();
41 var vector_layer_risk2=new
    OpenLayers.Layer.Vector("Risk 2",
    {styleMap:st_risk2});
42
43 var geojson_risk3=new OpenLayers.Format.GeoJSON();
44 var vector_layer_risk3=new
    OpenLayers.Layer.Vector("Risk 3",
```

```
    {styleMap:st_risk3});

var init = function ()
{
    // create map
    map = new OpenLayers.Map({
        div: "map",
        theme: null,
        controls: [
            new OpenLayers.Control.Attribution(),
            new OpenLayers.Control.TouchNavigation({
                dragPanOptions: {
                    enableKinetic: true
                }
            }),
            new OpenLayers.Control.ZoomPanel()
        ],
        layers: [ graphic,vector_layer_risk1,
           vector_layer_risk2,
        vector_layer_risk3,markers],
        center: new OpenLayers.LonLat(13.70422884,
           46.60030561),
        zoom: 1,
        maxExtent: new OpenLayers.Bounds( 13.6227,
           46.5215,
        14.0373, 46.7537),
        scales:[100, 150, 225, 338,506,
           759,1139,1709,2563,
        3844,5767,8650,12975,19462,29193,43789, 65684,
        98526,147789,221684,332526,498789 ]
    });

    // we indicate the source of the data for each
       layer, the information is stored in variables
```

```
        in the file "geojson/risk.js"
73      // the variable risk1 contains all the polygons
        with risk level 1, variable risk2 contains the
        polygons with risk level 2
74      // we also indicate that we want the layers to be
        visible
75      vector_layer_risk1.addFeatures (geojson_risk1.read
        (risk1));
76      vector_layer_risk1.setVisibility(true);
77
78      vector_layer_risk2.addFeatures (geojson_risk2.read
        (risk2));
79      vector_layer_risk2.setVisibility(true);
80
81      vector_layer_risk3.addFeatures (geojson_risk3.read
        (risk3));
82      vector_layer_risk3.setVisibility(true);
83
84      map.zoomToMaxExtent();
85
86 };
87
88 // function to add a new marker, we use it to insert a
     new marker on the user location
89 var addLocMarker=function(cLng,cLat)
90 {
91      OpenLayers.Console.log("addLocMarker");
92      var markerLocation=new OpenLayers.LonLat (cLng,
        cLat);
93      markers.addMarker(new OpenLayers.Marker
        (markerLocation,icon));
94 }
95
96 var setMapZoom=function(myZoom)
```

```
97 {
98     map.zoom=myZoom;
99 };
100
101 // to turn or or off the risk layers, it is executed
      from the menu of the avalanche risk application
102 var layerTurnOffOn=function()
103 {
104     if (vector_layer_risk1.visibility==true)
105     {
106         vector_layer_risk1.setVisibility (false);
107         vector_layer_risk2.setVisibility (false);
108         vector_layer_risk3.setVisibility (false);
109     }
110     else
111     {
112         vector_layer_risk1.setVisibility (true);
113         vector_layer_risk2.setVisibility (true);
114         vector_layer_risk3.setVisibility (true);
115     }
116 };
117 var moveCenter=function(cLng,cLat)
118 {
119       map.center=new OpenLayers.LonLat(cLng, cLat);
120 };
```

B.3 Avalache Risk Information

```
1 var risk1={
2 "type": "FeatureCollection",
3 "features": [
4 { "type": "Feature", "id": 10, "properties": {
    "grid_code": 1 }, "geometry": { "type": "Polygon",
    "coordinates": [ [ [ 13.690312, 46.596046 ], [
    13.690363, 46.595841], [13.690194, 46.595892 ], [
    13.690312, 46.596046 ] ] ] } },
5
6 { "type": "Feature", "id": 13, "properties": {
    "grid_code": 1 }, "geometry": { "type": "Polygon",
    "coordinates": [ [ [ 13.695323, 46.595916 ], [
    13.695323, 46.595826 ], [ 13.695192, 46.595826 ], [
    13.694931, 46.595827 ], [ 13.694932, 46.596007 ], [
    13.695063, 46.596006 ], [ 13.695062, 46.595916 ], [
    13.695323, 46.595916 ] ] ] } }
7 ,
8 { "type": "Feature", "id": 18, "properties": {
    "grid_code": 1 }, "geometry": { "type": "Polygon",
    "coordinates": [ [ [ 13.691145, 46.595749 ], [
    13.690964, 46.595839 ], [ 13.691147, 46.595929 ], [
    13.691145, 46.595749 ] ] ] } }
9 ]
10 };
```