

Smartphone Instrumentation for Collapse Detection of Buildings

A Mobile Crowd-Sourcing Approach Towards Earthquake Detection and Early Warning System

|  |
| --- |
| APPLIED COMPUTING PROJECT I & II SYSTEM EVALUATION DOCUMENT |

May 25, 2015

University of Oulu

Department of Computer Science and Engineering

|  |  |  |
| --- | --- | --- |
| ZEYUN ZHU  2382881  zeyunzhu@gmail.com | HAEJONG DONG  2292191  s2haejong@gmail.com | PERTTU PITKÄNEN  2261724  perttu.pitkanen@student.oulu.fi |

# Acknowledgement

We would like to express our gratitude towards Professor Vassilis Kostakos and Mr. Aku Visuri for their expert guidance. Meanwhile, we appreciate Professor Timo Ojala for providing us with a concrete outline for the evaluation document, respectively.

Table of Contents

[Acknowledgement 1](#_Toc420856742)

[Glossary 4](#_Toc420856743)

[Introduction 5](#_Toc420856744)

[Background and Motivation 5](#_Toc420856745)

[Background 5](#_Toc420856746)

[Motivation 5](#_Toc420856747)

[Evaluation Regime and Strategy 6](#_Toc420856748)

[Major assumptions and constraints 6](#_Toc420856749)

[Assumptions 6](#_Toc420856750)

[Constraints 7](#_Toc420856751)

[evaluation process 8](#_Toc420856752)

[Finding the accelerometer range 8](#_Toc420856753)

[Testing accelerometer delay mode 9](#_Toc420856754)

[Accuracy tests 9](#_Toc420856755)

[Battery consumption tests 9](#_Toc420856756)

[Stress testing on the server 9](#_Toc420856757)

[set up 10](#_Toc420856758)

[Test types and objectives 10](#_Toc420856759)

[Test procedures 10](#_Toc420856760)

[accelerometer range tests 10](#_Toc420856761)

[accelerometer delay tests 10](#_Toc420856762)

[Accurary test 11](#_Toc420856763)

[battery consumption tests 12](#_Toc420856764)

[Environment 12](#_Toc420856765)

[Users environment 12](#_Toc420856766)

[Developers environment 12](#_Toc420856767)

[Participants 13](#_Toc420856768)

[Data collection 14](#_Toc420856769)

[data and results 15](#_Toc420856770)

[Overview of data and finding 15](#_Toc420856771)

[Statistics 15](#_Toc420856772)

[analysis 22](#_Toc420856773)

[Overview 22](#_Toc420856774)

[Threshold tests 22](#_Toc420856775)

[Accuracy test results 22](#_Toc420856776)

[Server computing time results 22](#_Toc420856777)

[battery consumption test results 23](#_Toc420856778)

[Usability 23](#_Toc420856779)

[Scalability 23](#_Toc420856780)

[Fault tolerance 24](#_Toc420856781)

[Security 25](#_Toc420856782)

[References 26](#_Toc420856783)

[Contributions 27](#_Toc420856784)

[Team Meetings: 27](#_Toc420856785)

[Individual Contributions 27](#_Toc420856786)

[Zeyun Zhu 27](#_Toc420856787)

[Haejong Dong 27](#_Toc420856788)

[Perttu Pitkänen 28](#_Toc420856789)

[Summary 28](#_Toc420856790)

[Appendix 28](#_Toc420856791)

# Glossary

This section provides a brief explanation of the terms relating to the system evaluation document and the project in general. The terms are alphabetically ordered.

#### Accelerometer

Most of the contemporary smartphones have inbuilt sensors which are capable of measuring the devices acceleration. These acceleration sensors can be used through the Android API.

#### Accelerometer delay

Pre-defined delay for the smartphone’s accelerometer. Defines how often the accelerometer value gets updated.

#### Android

Popular operating system for smartphones.

#### API

Application programming interface. Tools for programming software applications and interfaces required to utilize different aspects of the system.

#### AWARE

Android framework for instrument and share phones inbuilt sensors’ data as well as user data.

#### Client

A computer software program that is designed to access data made available by a server.

#### DDoS

Distributed Denial of Service. An attack which tries to overwhelm the system by sending large amount of requests from multiple sources.

#### Heuristic evaluation

Methods that help to identify usability problems with user interfaces.

#### Micro-Electro-Mechanical Systems (MEMS)

Micro-Electro-Mechanical Systems, or MEMS, is a technology that in its most general form can be defined as miniaturized mechanical and electro-mechanical elements (i.e., devices and structures) that are made using the techniques of micro-fabrication. While the functional elements of MEMS are miniaturized structures, sensors, actuators, and microelectronics, the most notable elements are the micro-sensors and micro-actuators.

#### Plugin

Software application that is made to add specific functionality to an existing software application, in this case AWARE. In this work the client side implementation for smartphones is referred as the plugin.

#### Server

Running instance of application that is capable of responding to requests made by client.

#### Threshold

In this paper the collapse detection algorithm’s threshold is often referred with the word “Threshold”. The threshold in this case defines the square root sum value that triggers the fall event.

# Introduction

## Background and Motivation

### Background

The most common information available immediately following a catastrophic event such as an earthquake is its magnitude and epicenter location. However, an accurate and a full extent of the damage assessment require time and are currently mostly provided by estimates. For instance, the full extent of the damage from the 1995 Kobe earthquake, Japan, was not recognized by the central government in Tokyo until many hours later.  This greatly affected the rescue and recovery operations. Products such as ‘ShakeMaps’ [1] come close to the rescue by providing near real-time estimates using the sensors that are available from traditional seismic networks such as the Southern California Seismic Network (SCSN). A trade-off associated with these high-fidelity seismic sensors is its sparse distribution (approximately 10 km in case of SCSN [2]) resulting in maps of low-resolution. Increasing the density of such high-fidelity networks beyond that is needed for its basic function of locating the earthquake epicenter is cost prohibitive. The cost factor has been partially addressed by the use of open-network of low-cost microelectromechanical systems (MEMS) sensors that are hosted by volunteers [2]. However, the count of actual collapse of buildings and fatality is not obvious from these methods.

Another popular approach towards obtaining earthquake measurements data is the use of crowdsourcing. The “Did You Feel It” (DYFI) product of the US Geological Survey does this with a simple post-earthquake questionnaire [3] . The form of the questionnaire and the method for assignment of intensities are based on an algorithm developed by Dengler and Dewey for determining a "Community Decimal Intensity" [4]. Recent mild but widely-felt earthquakes in Los Angeles region have produced over 40,000 entries; supporting the likelihood of the popularity of such systems. However, this system has its caveats. One drawback of this form of sensing is that the human responders in the areas of heavy shaking usually do not make the data entry their first priority, and hence information from the most critical areas is usually late. Further, as this method of sensing requires users’ attention during times of peril, they could be considered hazardous in nature.

### Motivation

As a consequence, earthquakes and specifically the building collapses that happen during the earthquake are a serious threat to many people living in high earthquake risk areas. Collapse of buildings is one of the primary causes of fatality in any earthquake. Although hi-fidelity seismic sensors stations are deployed for early detection of earthquake tremors, little effort has been put to detect collapse of buildings in real time. The post-earthquake period of detecting collapsed buildings mostly involve image processing of aerial photographs. This method introduces a significant delay between the time of occurrence of the actual ‘collapsing’ event and the time the Emergency Response Units (ERUs) are deployed, thus causing a rise in fatality rates.

In this project, we focus on sensing the earthquakes and detecting collapsed buildings utilizing the smartphones as the distributed sensors. It introduces a system to cater to the needs of smartphone users with the help of AWARE framework.

## Evaluation Regime and Strategy

During this evaluation phase we focused on finding the best settings for the implemented collapse detector plugin using mostly a passive approach on testing the different settings for false-positives. We study the ways of using plugin with different accelerometer delay modes and with different fall detection thresholds. Meanwhile, an active approach was used to test the accuracy of the plugin and the server’s capabilities on handling many simultaneous clients. Moreover, we took user feedback into consideration by conducting questionnaires.

The primary components for the plugin in the evaluation phase are threshold and accelerometer frequency. The pre-defined threshold’s purpose is triggering a fall when the value of vector sum of accelerometer values is less than it. The accelerometer delay defines the frequency in which the accelerometer sensor will update.

We explain the data gathering and preparation process in detail, as these are crucial steps when real world data is analyzed. After data collection, we do the data analysis for it. The data analysis will show which accelerometer modes and thresholds works best for the system.

## Major assumptions and constraints

Because of the lack of similar systems being deployed we had to make some assumptions which will affect how the whole system will work. More specifically, the assumptions we made and the constraints we encountered are:

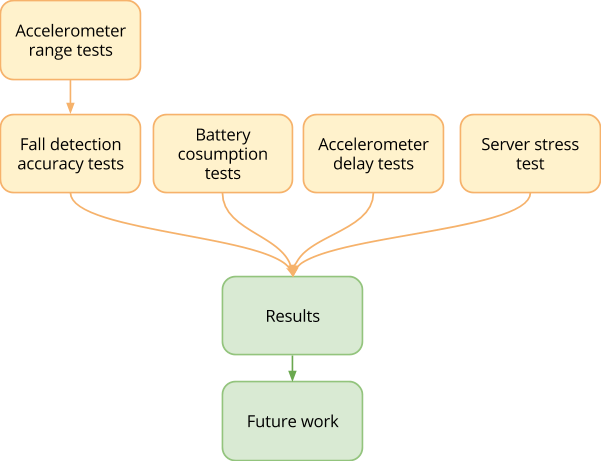
### Assumptions

1. **Smartphone is stationary.** An assumption upon the smartphones is that users will leave their smartphones on the table for the most of time throughout the day. However, evaluation of our system will indicate the feasibility and effectiveness of our approach when the system is deployed in real life environment where users tend to carry their smartphones with them.
2. **Fall event definition.** We measured the acceleration along the three axes: x, y and z. By taking the root-sum-of-squares of the accelerometer’s three axes, we are able to determine the acceleration. In theory when phone is in free fall the vector sum would be zero. If selected threshold doesn’t match the real-life fall event’s accelerometer values the plugin could result in false positives or false negatives depending on whether the threshold is too high or too low.
3. **The phone enters a free fall during a building collapse.** In theory it would seem probable that phones among other objects would be in a free fall just enough to trigger the fall detection. Although we haven’t been capable of testing system functionality in a realistic situation. It is unsure what kind of values the accelerometer will record during collapse and the values could depend on the location of the phone inside the building and whether the phone can fall freely or if encounters obstacles like other objects during the collapse and thus doesn’t enter a free fall. Multiple free falls could be also detected with phones during an earthquake without the building being collapsed. The phones could, for example, fall from tables at the same time in multiple locations inside the building because of the shakings caused by an earthquake.
4. **Collapse defining algorithm.** The assumption upon it is that when a fall event is observed by the server it generates a circle with a radius of 125 meters at the location of the event. Once this happened the server waits for a pre-defined time (time window) of ‘60’ seconds and registers all the incoming clients as a group of fall event. We assume when there are more than two fall events in the same circle that there will be probability of a building collapsed in the center coordinate of the circle. The probability varies depending on the number of fall events in a circle as well as the number of fall events that are registered in a group by the server.
5. **Isolated events.** The circumstance where a single client is resident in the building during the earthquake. There is no algorithm in the current system to identify such kind of situation when the building collapses as we define that all the single events as an isolated event which will be excluded from the server’s consideration.

### Constraints

**Demanding floating point calculations.** One constraint that the phone plugin has is that the floating point calculations needed to calculate the vector sum of the three accelerometer axes are very calculation-heavy operations. When the calculations are made several times in a second it is very demanding on the smartphones processor. Increased processor usage consequently shortens the battery lifetime in smartphones.

# evaluation process



**Figure 1**. Evaluation phase process

In the evaluation phase, we follow the process as displayed in Figure 1. It contains four tests in the evaluation phase for the plugin: finding the accelerometer range, accelerometer delay tests, accuracy tests, battery consumption tests and server stress test. We later analyses all the gathered data and make our conclusions.

## Finding the accelerometer range

Before we could start the accuracy tests we had to come up with the accelerometer delay we would use to test the threshold values. The objective was to find what is the range of the numerical values that the accelerometer sensor detects during a fall event. The range could be found by making a series of falls and monitoring the vector sum values during said falls.

## Testing accelerometer delay mode

Each accelerometer mode was planned to be tested by group of participants for approximately one week per mode. The Android API provided the four different delay modes that could be applied to the accelerometer sensor [5]:

* Fastest (0 microsecond delay)
* Game (20,000 microsecond delay)
* UI (60,000 microsecond delay)
* Normal (200,000 microseconds delay)

By testing each mode several days we could gather lots of valuable data on how many times phone falls gets triggered during normal activities.

## Accuracy tests

Fall detection algorithms accuracy was tested with a series of phone drops on a soft surface. By this method we would able define how the amount of detected falls depends on the different thresholds and accelerometer delays.

During these tests three different model phones would be dropped several times for each delay-threshold combination. We also included two different heights for the test: 15 cm and 70 cm. The amount of different combination added up to 48 different cases (6 thresholds x 4 delays x 2 heights = 48 cases). The amount of 40 drops should suffice for the test results to be statistically significant.

## Battery consumption tests

For the system’s user friendliness, the battery consumption would need to be tested. The hypothesis for this test is that the faster accelerometer modes will use more battery than the slower ones. This will be tested by using one phone with all the different accelerometer modes and see how much of battery they use and what are the differences between the modes.

## Stress testing on the server

Under circumstance of limited physical resources for the server to conduct an evaluation where it requires hundreds, thousands of testers joining all together within less than a minute of time, we have decided to write a module instead which generates similar types of data in the same format and send them to the server as many number as we want at a time.

# set up

This chapter describes in detail the practical plans for data collection and preparation.

## Test types and objectives

Accelerometer delay and threshold tests included passive testing where users launched the plugin and continued their daily activities normally. The plugin ran in the background and we studied how many times did the plugin would trigger a fall and for what reasons. After these tests the test subjects were able to answer a questionnaire where they could elaborate the reasons why the fall might have happened. Additionally the accuracy was tested with different threshold-delay pairs to find the optimal settings for both variables.

The evaluation had little heuristic evaluation in a free form question on general opinions of the user interface. The usability of the user interface was not the main objective for this evaluation phase. Evaluation consisted of mostly numerical and statistical analysis of the results.

## Test procedures

### accelerometer range tests

Accelerometer threshold scale was easily found using planned method. During the test a few phone drops were completed with threshold 0.8, and majority of those drops repeated values in the range of 0.2 to 0.6.

### accelerometer delay tests

The delay test, being similar, followed the previously discussed plan. For this test, a group of 7 people were recruited and they were given the plugin software. The users were told to install and launch the plugin for it to monitor fall events. The test started with the fastest accelerometer delay option. The mode was changed approximately once in a week. All fall data were sent to the server and saved in a database for further analyzing. The test durations for each delay tested were:

* Fastest mode: 6 days 7 hours (From 2015-04-16-2:30pm to 2015-04-22-10:22pm)
* Game mode: 5 days 11 hours (From 2015-04-25-2pm to 2015-05-01-0am)
* UI mode 6 days (From 2015-05-11-2:00pm to 2015-05-17-2:00pm)
* Normal mode 10 days (From 2015-05-1-1:30pm to 2015-05-11-1:30pm)

During this time the users would send feedback of the plugin’s functionality. Also with the implemented questionnaire pop-up the users sent some reasons why they thought the fall was triggered. Lastly an e-mail questionnaire was sent to the participants to reflect on the test.

### Accurary test



**Figure 2**. Testing in process

Accuracy was tested with the three differing phone models. The height, accelerometer delay, and threshold were changed between tests. This added up to 48 different test cases and for each case the phones were dropped 40 times. The phones were dropped on a soft surface so the phone wouldn’t bounce on impact. The drops happened from approximately 15 cm and 70 cm heights. The tasks were assigned to all three project members who then used their phones during the tests. Phone models used were:

* OnePlus One
* Galaxy Nexus
* Galaxy S4 mini

During these tests data was saved manually into a table for further analyzing. The tests took a couple of hours to complete.

### battery consumption tests

A good way to test battery consumption is to run the plugin with all the different accelerometer modes: Fastest, Game, UI and Normal. To get comparable results the test would need to be completed with same device. For this test Galaxy Nexus phone was used. Each of the modes were used for 3 hours and the percentage of the battery was monitored before and after the test. The modes were tested as follows:

* Fastest mode 6:02 pm - 9:02 pm
* Game mode 9:04 pm - 12:04 pm
* Normal mode 10:00 am - 1:00 pm
* UI mode 1:02 pm - 4:02 pm

## Environment

### Users environment

The users which are involved into our plugin test use the environment of smartphones.

#### Smartphones environment

The systems for the client should be deployed for Android. These phones are able to use the wifi-connection and GPS. The phone should have Google Maps and Aware installed. The environment items are listed as table 1.

|  |  |
| --- | --- |
| Operating system | Android 2.3 or higher |
| Pre-installed software | Aware  Google Maps |
| Phone features | GPS  Accelerometer  Wi-Fi |

**Table 1**. Smartphone environment

### Developers environment

Since the testing is an iterative process, developers need to fix bugs and make new features for testing purposes. The environment for developers’ is as follows:

#### Server environment

The server is keeping running in the background for all the users. It is used to supervise the client activity, and collect the data from the clients. The environment for server needed is listed as table 2.

|  |  |
| --- | --- |
| server account | [vm0101.virtues.fi](http://vm0101.virtues.fi/) |
| Linux version | 2.6.32-504.8.1.e16.x86\_64 |
| GCC version | 4.4.7 20120313(Red Hat 4.4.7-11) |
| Python version | 2.6.6 |
| Server version | apache/2.2.15(Unix),the server which is built in October 16 2014 14:48:21 |

**Table 2**. Server environment

#### Developer for android environment

Developer who needs to make application for the plugin should have android studio, and Aware on the Android phone. Google Map API is used in our plugin. The environment is explained in table 3.

|  |
| --- |
| Android studio |
| AWARE version 3.3.3 |
| Android version is 2.3.3 or higher |
| Google Map API |

**Table 3**. Developer for android environment

#### External libraries

The system is putting a requirement on some libraries. The libraries for developers in our project are listed as table 4.

|  |
| --- |
| Mysql version 5.1.73 |
| Pycrypto version is 2.6.1 |
| Python default libraries: time, math, socket |
| numpy |
| base64 |
| AWARE libraries |
| Android libraries |
| Java default libraries |

**Table 4**. Developer for libraries environment

#### Environment for the server stress tests

The system requires to know the capacity to stand stress on the server when hundreds or thousands of testers joining together within less than a minute. Table 5 lists the hardware that is needed for the developer who is testing stress on server for the plugin.

|  |
| --- |
| Intel(R) Core(™) i7-2679QM CPU @ 2.20GHz |
| 8.00 GB RAM |

**Table 5**. Developer for testing server stress environment

## Participants

We have 7 users: 1 girl and 6 boys. The age of users ranges from 22 to 31.All users are living near Oulu University.4 out of 7 persons are used to ride bicycle.2 out of 7 people often go to gym. They are all smartphone users.

## Data collection

Data including accelerometer value, location, phone id, timestamp is collected by passive testing where users launched the plugin and continued their daily activities normally. We also make some preparation on making the questionnaire.

* Make sure that the purpose of study is clear
* Promise anonymity
* Ensure a questionnaire was well designed
* Offer a short version for those who do not have enough time
* Follow-up with emails and phone calls

After testing session with our users, we asked each user to fill out a questionnaire regarding the plugin. The questionnaire consisted of following questions:

1. What kind of phone do you have (model)
2. What version of Android do you have? Instructions: <http://www.wikihow.com/Check-What-Android-Version-You-Have>
3. What kind of activity triggers the fall pop up message the most (bicycling, running, dropped the phone, inside backpack, place it on a table…)
4. How often do you drop your mobile?
5. What do you think of the battery consumption, did you notice any difference with different accelerometer delays (Fastest, normal, game, UI) which would you prefer?
6. Do you think the accelerometer delay affected the amount of detected falls?
7. Do you think the used threshold affected the amount of detected falls?
8. How often you turn on the app
9. Have you ever turned off the app that was already running, if yes of what specific reason?
10. If you live/lived in high earthquake risk area would you voluntarily install this app?
11. How could the app be improved to make it more attractive?
12. Did you have any problems during the installation?
13. Did any crashes happen during the app usage and for what reason?
14. Any other comments?

# data and results

## Overview of data and finding

For most of the time the data and survey answers received matched the expected results. The test subjects thought that the plugin filled its purpose well although the fastest accelerometer modes consumed a lot of the phones’ battery which was very uncomfortable. The accuracy tests fulfilled the expectations of the thresholds and accelerometer delay’s effect on the results.

## Statistics

#### Questionnaire data

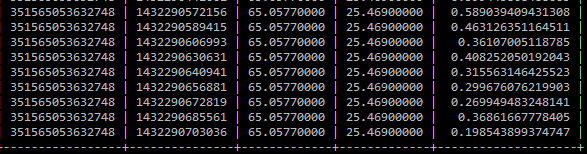
After user filled out the questionnaire by email, we received answers from 7 users in total and got the following results:

* The pop up message is triggered by following reasons:

1. Dropping phone on bed
2. Dropping phone on table
3. Dropping phone on floor
4. While dismounting from bicycle
5. While riding bicycle

* Accidentally dropping the phone from longer heights is pretty rare. The frequency of accidental phone drops ranged from once in a week to once in a month.
* 5 out of 7 people surveyed disliked the fastest accelerometer delay mode because it lets the phone battery survive within 6 hours.
* Most of the people did not know whether accelerometer delay and the threshold affected the amount of detected falls.
* 6 out of 7 people launched the plugin at least once a week.
* All people were glad to install the app if they lived in a high earthquake risk area. People show a big interest in this app, they think it fulfills its purpose.
* Some people met problem when installing the app because their phone was old or aware did not recognize the plugin.

#### Range of fall event accelerometer data



**Figure 3**. Range of the accelerometer values during a fall (Accelerometer vector in the rightmost column)

By setting the threshold value as 0.8, we got the accelerometer value from dropping phone on purpose. From figure 3, we get to know the range value for accelerometer when the phone is dropping is approximately from 0.1 to 0.6.

#### Amount of fall events per day

When 7 people carried their phones during their normal activities, the average of fall event number is around 3 per day. This data may also be affected by the activity of the test users, which cannot be predicted. Although this data gave some good idea of how many accidental fall events happen when there are no earthquakes.

Figure 4 shows the data gathered with the fastest accelerometer mode. The average occurrence of a fall is 6 falls per day.

**Figure 4**. Fastest mode

Figure 5 shows that the game mode causes averagely around 3 fall events per day. In theory the average fall amount should be close to the fastest mode. Although change in the users’ activity may have had an effect on the data.

**Figure 5**. Game mode

As the figure 6 shown, the fall event number is around 1 per day in the normal mode.

**Figure 6**. Normal mode

People who use the UI mode get around 2 fall events per day as shown in figure 7.

**Figure 7**. Ul mode

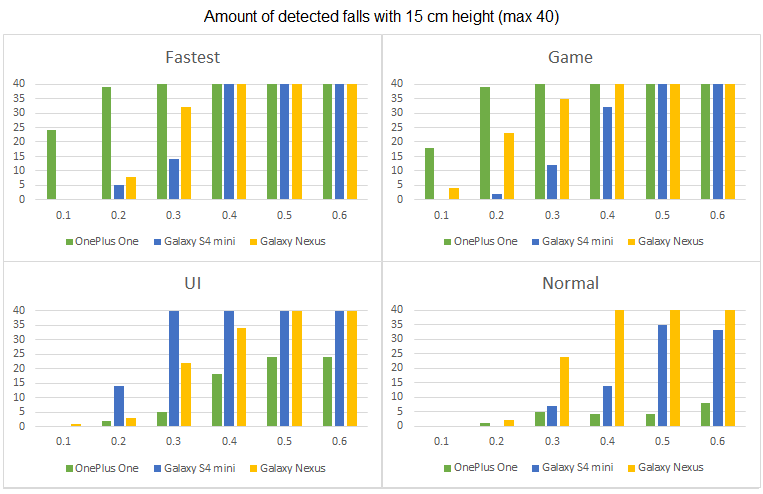
#### Accuracy data

During the accuracy tests good amount of data was gathered. The data was divided to two cases: the heights 15 cm and 70 cm. Used thresholds ranged from 0.1 to 0.6. The maximum amount of detected falls is 40 which is the total amount of the phone drops made for each threshold. The data answered well to the initial questions whether phone model, fall height, accelerometer mode or threshold affected the results. Generally all the changed variables have their effect on the outcome.

**15CM**

For 15 cm height drops, differences between different cases can be seen. For 15 cm height, the threshold of 0.4 seems to detect majority of the falls when used the fastest and game modes. On the other hand this threshold does not work as well with lower frequencies, detecting smaller amount of the fall events the more the frequency is lowered. Also here can be seen quite big differences between the phone models. For example with the normal mode the Galaxy Nexus detects 100 % of the falls whereas the OnePlus One and Galaxy S4 mini detects only a small fraction of the actual falls.

For small fall heights such as 15 cm the accelerometer delay has a big impact on the outcome. For this height the fastest and game modes seem to be the only ones that have reliable results with all tested phones.



**Figure 8.** Statistics for 15 cm height phone drops

**70CM**

Whereas with lower height falls are highly dependent of the accelerometer frequency, the higher fall events are predictably easier to detect. With higher falls the phone has more time to update the sensor values and thus has higher probability of fall detection. Like in the case of 15 cm fall, the lower thresholds have worse fall detection capability. The data shows how higher fall heights are easier to detect with lower accelerometer delays. For example with the Normal delay and threshold 0.5 the 70 cm fall seems to be detected much better than the 15 cm ones. Therefore the UI and normal modes perform better during these higher falls and the performance of Fastest and Game modes stays almost the same.

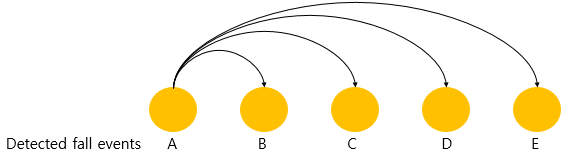
Also with the higher falls some differences can be seen with the used phone models. For example the OnePlus One phone performs much better with higher fall heights using the lowest accelerometer delay modes UI and Normal.

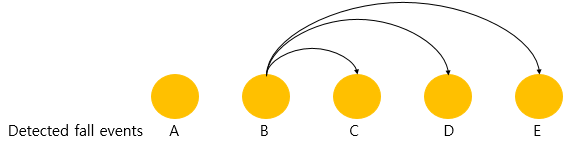
****

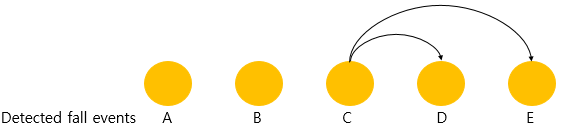
**Figure 9.** Statistics for 70 cm height phone drops

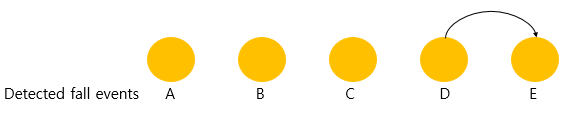
#### Computed time data

Here we have encountered an issue while testing on capacity of the server. The current algorithm calculating distance between locations of each event is shown Figure 10. However, this has very slow performance as the number of components to be calculated increases. It requires 10 times of computing until selecting locations where are suspected to be collapsed buildings. Thus the computing time reaches up to 209 seconds already when 2000 clients sending fall events at the same time, see in Figure 11. While several hundred thousand of victims are reported within a single earthquake the capacity of system is rather too scanty. The capacity varies depending on the size of the time window. Once received data is being calculated, it has time of size of a time window, which means the calculation has to be done before the next time window is closed and another set of data arrived.



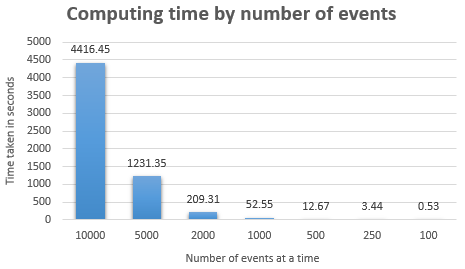








**Figure 10**. Current algorithm has been running in the server



**Figure 11**. Taken computing time by number of events detected within a time window

#### Battery consumption data

When the battery tests were completed quite surprising data was gathered. The data didn’t match the expectations and it seems that the different modes have consistent battery usage.

|  |  |  |  |
| --- | --- | --- | --- |
| Mode | Percentage battery consumption | Date | Time range |
| Fastest mode | 14% | 2015-5-30 | 6:02 pm-9:02 pm |
| Game mode | 13% | 2015-5-30 | 9:04 pm-12:04 pm |
| Normal mode | 13% | 2015-5-31 | 10:00 am-1:00 pm |
| UI mode | 17% | 2015-5-31 | 1:02 pm-4:02 pm |

**Table 6**. Battery consumption

# analysis

## Overview

### Threshold tests

In theory the accelerometer should have the value zero during the free fall. Although our tests proved that in real life the range of the accelerometer varies from 0.2 to 0.6 during a fall event. For this reason the threshold value should be in this range. However the choice of threshold causes problems with other aspects of the system. In the drop test results more falls seems to occur the higher the threshold is.

### Accuracy test results

While testing the accuracy of the plugin’s capability of detecting fall events, the most prominent findings were how large effect the accelerometer delay, threshold and phone model have on the plugins performance. The higher height falls were also easier to detect with lower delays. It is difficult to predict what kind of fall heights or trajectories the phone will experience during an actual building collapse. For this reason the assumption of free fall was made.

In free fall situations the phones were capable of detecting the fall if the threshold were set right. For thresholds higher than 0.3, the Fastest and Game modes performed the best in both lower and higher drop heights.

### Server computing time results

First alternative to solve the computing time issue would be generating multiple threads that’s going to be assigned one for each set of data to be calculated. This might seem to sort the problem out as then the server just need to wait until all calculations are done in parallel by multiple threads and printing out. However this cannot be a permanent solution as shown atfigure 11,the rate of increasing computing time is beyond imagination as number of events reaches over a thousand. Moreover the fact that this system is based on real-time warning notification, time issue is critical. One hour delayed information is certainly not useful at all.

Altogether we need a total new feature of algorithm for better performance. We have only come to an idea to reduce the computing time so far by adding assumptions. This primitive solution might significantly reduce number of events to be calculated. For example if we are able to figure out a cluster of majority data that’s closely located each other before the completion of calculating all targeted events, we assume that the remaining can be then ignored and announce the suspected location much earlier.

### battery consumption test results

The battery consumption test result did not match the expected results. The gathered data seems quite the opposite of the expected results: the slower accelerometer delays seems to consume more battery than the faster ones. This may be a result of an unstable battery or some other processes using the battery. All in all more tests would be needed to be sure of the results.

From the user tests some feedback was given about high battery consumption with faster accelerometer modes, which also contradicts the results we got from the tests. For better results more tests with several different phones would need to be completed, which couldn’t be done during this phase because of the lack of time.

### Usability

When researching the subject an idea of usability came across. The results prove that users dislike high battery consumption caused by high accelerometer frequencies. For this reason the users wanted to use the Normal or UI options for the accelerometer delay. On the other hand if we concentrate on user friendliness by using the slower modes we will need to prepare for less accurate results, and that some of the falls will not be registered.

The second option for optimizing the accuracy it would be sensible to use the phones as stationary collapse detectors fastened inside the building. The phones would be then connected to a continuous power source so the battery usage would not be an issue. Even with power outages which may happen as a result of an earthquake, the phones would still have their batteries to supply power for the plugin for many hours.

When deploying the system one possible way to solve this problem could be to use both options parallel. Some of the phones would be static with higher accelerometer update rate and some could be with the users set up with lower settings. For making the installation easier the system could have pre-defined thresholds for each accelerometer mode. These thresholds could also be calibrated for each phone model separately.

## Scalability

Currently we have limit in capacity of system on the server that because of the way it calculates incoming events. About 1000 clients can fit in at a time to avoid delayed information provision. However this can be improved by employing a novel algorithm. Next step will be coming up with the algorithm that is able to handle about 10,000 - 20,000 events within a minute. Once we have that, distributed processing system could reduce even more the amount of data to be calculated by processor which would help to avoid sort of bottleneck phenomenon.   
Samsung has released their smartwatch last year and Apple also released it this year. Apparently the smartwatch is latest trend in mobile industry thus speaking about scalability. We could also extend our system into smartwatch platform where it does the similar jobs as now it does on mobile but more compacted and informative interface to play in a smaller device.

Another potential field is web. This has been already discussed during the design phase of the system building up a webpage for some of our stakeholders such as researchers or emergency organizations whoever finds the information useful. We could serve a webpage with visualized information in real-time however this may require extra computing power and advanced fault tolerance to keep multiple applications up and running. For this service we would wish to collect other various data as well from clients. For example additional temperature data collection may allow the system being able to define in which place or area has been on fire during earthquake and then the system visualizes the data onto the webpage so that would eventually help firefighters making a better decision in their services. Furthermore, usage of offline big data and combination of it could leverage the ability of entire system to the limit.

Recruiting new clients wouldn’t be a technical issue though it still can be a major issue in other term of scalability.

We would also need multiple server centers over targeted area or a country to minimize data travel time and stable, high speed and reliable network infrastructure.

## Fault tolerance

The fall detection algorithm implemented in the plugin may trigger a fall event even if the phone is placed on a surface or while riding a bicycle. Although the original planned purpose of the plugin was to monitor whether the phone has fallen or not, and the plugin seems to fill its purpose quite well. The fall detection algorithm may need some refining, but while the displacement of objects during a building collapse is highly unpredictable, it would need more careful research what actually happens during the collapse.

As discussed before falls can be triggered in the plugin quite easily. If happened inside a small area, such events can be able to trigger a building collapse in the server software. The risk of this happening is tried to minimise by setting the server a time window for collapse detection. It is more unlikely for the phone falls get triggered both inside same radius and time window. The length of the time window has not been discussed during this evaluation process because of the lack of time and resources.

The android API also notifies the user if the plugin crashes for some reason. The user then has to start the plugin again, which in theory could be automated to ensure that most of the phones would continue monitoring even if the user forgets to launch the plugin again. To prevent this kind of crashes the bugs found during the tests were tried to fix as soon as possible.

If for some reason some of the devices will go offline during the time the system is in use the heartbeat message has been implemented as a part of the system. The heartbeat message means that the client will send its device id number once in a minute to the server so the server will know which devices are still online and ready to send and receive data.

When the server is down the back-up server is prepared to take up its task until the main server is back and running normally. However during a month of evaluation procedure the server has been kept up running yet no unexpected crash or error found.

## Security

The current security level is rather basic. During evaluation phase we have come to an idea to improve the security structure regarding to the shared key management. In current system it has followed two methods: First, both server and all clients share one single key value. This had been implemented in that way only for the purpose of testing if encryption and decryption works. Meanwhile the second work which could be implemented in the future work is that server and each client will have a shared public key as well as a private key. The client will send data and private key value encrypted with public key to the server, server is aware of public key value so as of that moment both server and the client communicate by using the private key value which at the very first transmission the client created and sent it to server.

The server also may need to be able to endure Distributed Denial of Service attacks (DDoS). In which multiple sources will overwhelm the server by sending huge amounts of data to try and make the server to crash. These attacks could be prevented by keeping track of the exact number of devices the system has and allow only so many separate connections. With this technique harmful outside connections cannot be made. There are also multiple other techniques on preventing DDoS attacks [6].

# References

|  |  |
| --- | --- |
| [1] | H. T. Clayton R, "Community seismic network," 2012. |
| [2] | Q. V. Wald D, "Utilization of the Internet for rapid community intensity maps," 1999. |
| [3] | "About Community Internet Intensity Maps," 2000. |
| [4] | "Sensors Overview | Android Developers," 2015. [Online]. Available: http://developer.android.com/guide/topics/sensors/sensors\_overview.html. |
| [5] | "SensorManager," 5 2015 . [Online]. Available: http://developer.android.com/reference/android/hardware/SensorManager.html. |
| [6] | G. M, "A Summary of DoS/DDoS Prevention, Monitoring and Mitigation Techniques in a ServiceProvider Environment," 2003. |

# Contributions

All members equally participated in the certain team meetings and helped each other during the within-team deliverables. Below is a breakdown of the contribution of each team member.

## Team Meetings:

4 weeks: 2 hours per week - 8 hrs. All team members were present during the meeting.

1 weeks: 4 hours -4 hrs. All team members were present during the meeting.

## Individual Contributions

### Zeyun Zhu

I am the project manager of the team and have been entrusted with some additional project management duties. It was my duty to ensure that we have proper team meetings every week. I also charted our plan for the evaluation stage, having provided ideas on how to evaluation the project ideation. As a team member, I contributed through participant recruiting, making questionnaire, doing questionnaire data collection, finding the accelerometer range, recording the accelerometer delay mode data, analyzing the accelerometer delay mode data, participating the accelerometer delay mode testing, participating the accuracy testing, participating the battery consumption testing, recording the battery consumption data. Below is a breakdown of my total effort during this stage.

|  |  |
| --- | --- |
| Topic | Hours |
| Project Management | 2 |
| Meeting | 12 |
| Participant recruiting | 1 |
| Making questionnaire | 1 |
| Doing questionnaire data collection | 2 |
| Finding the accelerometer range | 4 |
| Recording the accelerometer delay mode data | 23 |
| Analyzing the accelerometer delay mode data | 3 |
| Participating the accuracy testing | 4 |
| Participating the battery consumption testing | 1 |
| Recording the battery consumption data | 1 |
| Writing Report | 16 |
| Making presentation | 3 |
| **Total** | **73** |

Table 7. Zeyun Zhu working hours

### Haejong Dong

I have contributed on system wide evaluation tasks as listed in the table below.

|  |  |
| --- | --- |
| Topic | Hours |
| Meeting | 12 |
| Writing a testing module | 8 |
| Module testing | 2 |
| Implementing features for testing client | 3 |
| Participating the accuracy testing | 4 |
| Analyzing server stress | 5 |
| Questionnaire | 1 |
| Writing report | 8 |
| Preparing presentation | 4 |
| **Total** | **47** |

Table 8. Haejong Dong working hours

### Perttu Pitkänen

I contributed to the project by participating the accelerometer delay mode testing, participating the accuracy testing, implementing new features and doing bug fixes to the plugin. I did a good amount of report writing as well.

|  |  |
| --- | --- |
| Topic | Hours |
| Meeting | 12 |
| Participating the accuracy testing | 4 |
| Implementing security | 6 |
| Implementing new types of data to be sent for server | 4.5 |
| Implementing plugin settings | 5 |
| Writing report | 13 |
| **Total** | **44.5** |

Table 9. Perttu Pitkänen working hours

## Summary

The table summarizes the contribution of each of the team members to this design phase.

|  |  |  |
| --- | --- | --- |
| Name | Total Work in Hours | Contribution to the Total (%) |
| Zeyun | 73 | 44 % |
| Haejong | 47 | 29 % |
| Perttu | 44.5 | 27 % |
| **Total** | **164.5** | **100%** |

Table 10. Summary table of total work load and individual contributions

# Appendix

The codes for the smartphone plugin can be found at GitHub repository: [Collapse detector](https://github.com/Perttu-/Collapse-detector)