IMPLEMENTATION OF A PROBABILISTIC INTERRUPTION MANAGEMENT SYSTEM FOR SMARTPHONES

by

William Vilwock

Honors Thesis

Carroll College Department of Mathematics, Engineering, and Computer Science

Helena, Montana

April 2014

SIGNATURE PAGE

This thesis for honors recognition has been approved for the

Department of Mathematics, Engineering, and Computer Science.

[illegible] 3/27/14

Director Date

[illegible] 02/31/2014

Reader Date

[illegible] 3/26/14

Reader Date

Abstract

As the number of worldwide cellular subscriptions approaches the world's population, the negative effects of cell phone disruption have become increasingly apparent. In recent years, mobile phones have become advanced enough to moderate interruptions based on whether or not the user would want an interruption. Research into the area of interruption management has provided models and architecture for the creation of such an application. However, to our knowledge, there are no interruption management systems currently available in the Android or iPhone app stores using a probabilistic model to moderate cell phone interruptions. A probabilistic model would be an improvement over current binary decision models, which either lack accuracy, or require the user to predetermine every possible outcome.

In this project, we have employed a probabilistic model to implement an interruption management system for Android OS 4.0 that uses five contexts: schedule, time of day, location, caller relationship and driving. Our system intercepts the call, calculates the probability an interruption is desirable, and then changes the phone's audio profile to vibrate, silent, or ring based on our model. A missed notification service has also been implemented to alert the user, at an appropriate time, that he has missed notifications.

Acknowledgements

I would like to thank Dr. Praveen Madiraju for being my mentor through my research experience and providing me with advice and encourage along the way. I would also like to thank Dave Marshall, my thesis director, for his suggestions on areas to expand upon my summer research, as well as for his advice on improving my paper. Finally, I would like to thank Dr. Kay Satre for reading and providing grammatical corrections to my thesis. This work was supported by the National Science Foundation grant CNS-REU-1063041, and by Marquette University's MSCS REU program.

TABLE of CONTENTS

1. Introduction ... 1
2. Motivation ... 3
  2.1 Limitations of Binary Decisions ... 4
3. Previous Research ... 6
4. Determining Important Contexts ... 6
  4.1 Similar Applications ... 7
5. Creating a POI model ... 10
  5.1 The Process of Determining the Weighted Sum Model ... 11
    5.1.1 Weighted Average ... 11
    5.1.2 Weighted Average with Default Values ... 12
    5.1.3 Weighted Sum (Using a Ranking Method) ... 13
    5.1.4 Weighted Sum with Adjusted Weights ... 14
6. Implementation of the Application ... 15
  6.1 The Mobile Platform ... 15
  6.2 Application System Flow ... 15
  6.3 Determining the Preferred Method of Interruption (PMI) ... 18
  6.4 Obtaining and Setting Preferences for the Contexts and Weights ... 18
    6.4.1 Weight Preferences ... 20
    6.4.2 Location Preferences ... 20
    6.4.3 Schedule Preferences ... 21
    6.4.4 Contact Preferences ... 22
    6.4.5 Time of Day Preferences ... 22
    6.4.6 Driving Preference ... 23
7. Missed Notification Feature ... 24
  7.1 What the Missed Notification Feature Does ... 24
  7.3 Missed Notification Implementation ... 25
  7.3 Benefits of the Missed Notification Service ... 29
8. Application Status and Future Work ... 30
  8.1 Status ... 30
  8.2 Current Application Issues ... 30

8.3 System Usage 31
8.4 Future Work 31
9. Limitations of Our Application and Current Technology 32
9.1 Limitations of our Application 32
9.2 Current limitations of technology 33
10. Future Research 35
11. Conclusion 36
References 38
Appendix 40
The Fifteen Scenarios 40
Results of the various weighted sum/weighted average models 44

1. Introduction

Interruptions are a common part of everyday life. They are often necessary, such as a fire alarm going off. At the same time, unwanted interruptions can prove to be detrimental, such as a co-worker barging in on a meeting. While unwanted interruptions can be caused by an infinite number of factors, one device has opened the gateway like never before: the cell phone.

Cell phones have become ubiquitous within our society. The International Telecommunication Union predicts the global number of cell phones subscriptions will hit 7 billion in 2014, closing in on the number of people who inhabit the planet [8]. The services and capabilities of cell phones are immense. Yet, at the same time, this level of connectedness leaves users more vulnerable to harmful distractions than ever before. Research has provided quantitative evidence to display the negative effect of cell phone interruptions on society. Commonly affected spheres include business, education, and driving. As cellular devices become even more common in everyday life, the need to manage their interruptions becomes increasingly apparent.

Producing a system to manage interruptions has been the aim of a moderate amount of research. This previous research provides ample information on the benefits and harms of interruptions, models to measure the impacts of interruptions, and methods of producing an interruption management application. An important aspect of many of these research projects is Cost of Interruption, or COI, which measures the cost an interruption has on a user. Related to COI is POI, or Probability of Interruption. POI measures the probability an individual would want to be interrupted, and is inversely related to COI. For example, a high probability of interruption would indicate the cost to the individual is low, while a low probability of interruption would indicate the cost to the individual is high. While both POI and COI can be used to determine whether an individual would desire an interruption, POI was used for this research.

Most modern smartphones have been embedded with sensors capable of comprehending the user's surroundings. By using these sensors with other features on the phone, it is possible to understand much about the state of the user and, from that, calculate the POI. Despite the amount of research done on interruption management, to our knowledge, a widely available application has not been released that uses a probabilistic method to manage interruptions. Currently, most interruption management applications use a binary model, which either lack accuracy, or require the user to predetermine every possible outcome. A probabilistic model could solve these issues, and our research looks to be the first accomplish this task.

Our application was developed in several steps. The first step was determining what factors, or contexts, go into deciding whether a user would want an interruption. To determine the necessary contexts we evaluated practical scenarios, looked at examples from previous research, and evaluated current interruption management applications not using a probabilistic approach. Once the contexts had been determined, we created and evaluated a mathematical model to determine POI. Upon building a model, we were able to design, code, and test the application.

Currently, we have produced an application that can take an incoming call/text as input, calculate the POI using our contexts and mathematical model, and then change the audio profile (silent, vibrate, or ring) of the phone accordingly. Our research up to this point formed the subject of an IEEE conference paper [12]. Since the conference, a missed notification service has been implemented to notify users, at an appropriate time, they have missed notifications. While significant progress has been made towards developing our application, additional testing, debugging and UI changes need to be made before launch.

Our research hopes to provide several additions to the area of interruption management. First, our base level application can be compared to any latter installations of an interruption management application. Second, future research could use or modify our design. Third, by publically distributing our application, we can gain feedback from actual users. Feedback is important as it can be used to enhance the mathematic model and the application. Finally, we have found challenges and limitations that go along with building an interruption management application.

2. Motivation

In a society filled with cell phones, it is easy to see the motivation behind managing their interruptions. We have all experienced a situation when an untimely cell phone interruption has had a negative impact. Research has provided statistics of the impacts cell phones have on society as a whole. Several spheres are particularly influenced by cell phone disruption: business, driving, and education.

In the business sphere, cell phones have been shown to be a terrible disruption. Undesirable interruptions, such as cell phones ringing, take up 28 percent of a knowledge worker's day [10]. In a year, that totals to 28 billion hours wasted [10] and results in a total loss of over $650 billion, considering the average labor rate of $24 an hour [2]. Productivity is clearly a casualty of unwanted cellular interruption. In addition to productivity, safety can be jeopardized due to cell phone distraction, especially while driving.

The National Highway Traffic Safety Administration did a study on 100 vehicles for one year. They collected information on the vehicles, such as how many crashes, near crashes, and critical incidents occurred. They found around 80 percent of crashes and 65 percent of near-crashes involved distraction of the driver within three seconds of the crash. The most common distraction for drivers was found to be cell phones [9]. The distraction cell phones cause is so great twelve states prohibit all drivers from using hand-held cell phones while driving [6].

Cell phone distractions also have negative impacts on school settings. A pilot study of graduate students indicated 85.1% of students believed ringing phones were a distraction and 72.3% stated they have had their phone go off in class [3]. The high percent of students believing cell phones to be disruptive, along with the high percent of students admitting to having their phone go off in class, clearly indicates cell phones create disruptions in educational settings.

While cell phone disruptions affect more than just these areas, these three spheres supply enough evidence to display the importance of an interruption management system. Further support comes from the limitations of currently available applications looking to manage cell phone interruptions. While quite a few applications manage the user’s audio profile, all of these applications use binary decisions to determine the final outcome.

2.1 Limitations of Binary Decisions

A binary decision has two possible outcomes: yes or no. In terms of interruption management, a series of binary decisions can be used to determine whether or not a user would want an interruption. One methodology is to determine if the user would want to be interrupted based on various contexts, and then go with the majority. Another way is to have no interruption if even one context evaluates to no interruption. Both these methods suffer accuracy as they do not consider how much each context matters to the user. One more method is to navigate a binary tree displaying all possible combinations. A binary tree should produce an accurate result every time, but it comes at a cost.

Applications that rely on using binary trees to determine the audio profile are limited by the user having to predetermine every possible outcome. For example, say one of these currently available applications makes decisions based on three contexts: location, schedule, and contact. To function correctly, the individual must specify how the phone profile should act given each context and every combination of the contexts. What really makes this process implausible is each context can have multiple instances, such as home and school being used for locations. Having to enter data for all possible outcomes can become tedious, and it is not logical for the user to predetermine every possible situation she will encounter. Here is where the advantage of using a POI model can be seen.

A POI model does not require the user to predetermine every possible outcome. Instead, it judges relevant criteria and produces a probability of whether or not the user would want an interruption. Different POI models use different contexts and have different ways of evaluating them. The creation of the POI model we used is explained in Section 5. How we obtain the values relevant to each context can be found in Section 6.4.

3. Previous Research

The main goal of our research is to produce the first widely available interruption management system using a probabilistic model. At this time, our research does not involve the comparison of our model or implementation of the application with other research. However, this research was particularly influenced by Sina Zulkernain's Master's Thesis *Modeling Cost Of Interruption (COI) To Manage Unwanted Interruptions For Mobile Devices* [13]. In his thesis, Zulkernain presents a compilation of the research done on COI, as well as the POI work done on mobile devices.

A common set of characteristics emerges from Zulkernain's compilation of research on the topic of interruption management. We used these characteristics to determine the important components that go into producing an intelligent interruption management system. The steps include:

1. Specifying factors and contexts,
2. Building a mathematical model to measure the cost of an interruption (or probability of interruption),
3. Simulating and evaluating the model,
4. Producing a prototype application.

4. Determining Important Contexts

Before any work could commence on a mathematical model or the application, we needed to decide the contexts we would measure. As we determined we would use the Android platform due to its open source nature and user base (See Section 6.1), we first determined what contexts could be measured with a smartphone running the Android OS. Google simplified this task by providing a list of sensors supported by the software development kit. In addition, Android also offers access to features and services of their native applications, such as calendar and contact information. Below is a list of the features we had access to on the Android platform:

Accelerometer
Ambient Temperature
Gravity
Gyroscope
Light Sensor
Linear Acceleration
Magnetic Field
Pressure
Proximity
Relative Humidity
Rotation Vector
GPS
Microphone
Camera
Native and Third Party Applications

Figure 1: Accessible Features of the Android Phone

While these features allow us to measure a wide range of different contexts, many of them are not relevant to whether or not an individual would want an interruption. To determine important contexts, we brainstormed different scenarios as well as looked at similar applications already on the market.

4.1 Similar Applications

Many currently available applications provide an interruption management service. From these applications, we could see the common components they take into account when determining whether the user would or would not want an interruption.

While many applications on the Android or iPhone app store do interruption management, to the best of our knowledge, none of them use a probabilistic method to make the decision. Instead, they rely on binary decisions based on user input beforehand. Binary decisions provide steep limitations, either providing low accuracy, or requiring the user to predetermine every situation (See Section 2.1). While the list below is not a comprehensive list of interruption management applications, it provides a good sample of available applications and the contexts they incorporate.

Name	**Platform**	**Schedule/ Time of Day**	**Contact**	**Location**	**Driving**
Auto Ring	Android		X		
Auto Silence	Android	X			
AutoSilent	iOS	X		X	
AutoSilent	Android	X			
Busy Me	Android	X			
Husher	Android	X			
Llama	Android	X	X	X	
PhoneGuard	Android				X
Ring Scheduler	Android	X	X		
Selective Silence	Android		X		
Silence	Android	X			
Silencify: Smart Silence	Android	X	X	X	
Silent Sleep	Android	X			

Smart Silencer	Android	X	X		
Smart Silent Beta	Android		X		
Smart Volume Control	Android	X	X	X	
Smart Volume Profile Manager	Android	X		X	
SuperSmartPhone	Android	X			
Tasker	Android	X	X	X	
TextArrest	Android			X	X
Timeriffic	Android	X			
Our Application	**Android**	**X**	**X**	**X**	**X**

Figure 2: Current Interruption Management Applications

Figure 2 displays a good selection of currently available applications that provide interruption management using binary decisions. The list displays the application's name, the platform it runs on, and the contexts it can measure. From Figure 2 we can see the contexts most commonly used. Most of these applications only look into one or two of these contexts. While none of the applications in our sample measures all of the contexts, our application looked to incorporate all of them. From the resources we have, the existing applications, and previous research, we decided our model would take into account five contexts: location, schedule, contact, time of day, and driving.

5. Creating a POI model

In previous research, Bayesian Probabilities models were commonly used for interruption management [7, 11]. However, Zulkernain points out that Bayesian models have several limitations. The first is requiring complete knowledge of a system, including all the a priori and conditional probabilities. This information can be difficult to determine beforehand. Another issue is a priori probabilities are traditionally measured from empirical data or from a uniform distribution, which is not always available. For these reasons, we did not pursue a Bayesian Probability method.

The solution Zulkernain offered implemented the Dempster-Shafer theory. Using this theory, it is unnecessary to know the a priori and the conditional probabilities. Instead, each context is evaluated and considered a piece of evidence. All the evidence is combined to make a final decision about where or not to interrupt the user. In addition, the Dempster-Shafer theory model takes into account uncertainty. Uncertainty arises from not knowing, or not having access to, all the information about a user. Part of the reason for uncertainty is smartphone's sensors not being 100% accurate in their measurements (See Section 9.2 for Current Limitation of Technology). As many factors can go into whether a user would want an interruption, and they can differ for each person, this method is highly relevant to cellular interruption management. However, for our project, we assume the probabilities for each context can be obtained, and if not, a default value is assigned (See Section 5.1.4).

One suggestion made in Zulkernain's thesis for determining POI is to do a weighted sum/average. Using a weighted sum or a weighted average is a simplistic way to make a decision based on multiple criteria. With this thought in mind, we experimented with weighted sums and weighted averages to determine a model.

5.1 The Process of Determining the Weighted Sum Model

A weighted sum/average provides either a sum or an average for a group of values not contributing equally to the whole. This is exactly the case for a POI model. Each person is affected differently by each context. Person A, for example, may not want an interruption while he is driving, but Person B may be fine with this. In this sense, each context can be assigned its own POI. However, these contexts cannot simply be considered in isolation from one another. Instead, each context influences an individual’s overall POI to a different degree. In other words, a weight could be assigned to each context. The overall POI is a sum of the POI of each context, times its respective weight. Initially, we started out with a weighted average.

5.1.1 Weighted Average

$$P(I) = \frac{P(I)_L * W_L + P(I)_S * W_S + P(I)_C * W_C + P(I)_T * W_T + P(I)_D * W_D}{W_L + W_S + W_C + W_T + W_D}$$

In this model, $P(I)$ is the probability of interruption. $P(I)_{\{L,S,C,T,D\}}$ are the probabilities of interruption based on location, schedule, contact, time of day, and driving. $W_{\{L,S,C,T,D\}}$ are the weights assigned to each respective context. For this model, only the measurable contexts were evaluated. The term measurable means the POI for the context is set beforehand by the user and the context is relevant to the user’s situation. If the user is not driving, for example, the driving context is considered not measurable. In addition, if no POI is set, then the model cannot evaluate the context and must consider it not measurable.

As this model is a weighted average, the value is obtained by dividing by the sum of the weights attributed to each context. This function also requires the POI of each context, as well as the weights, to be between 0 and 1. Both the weights and the POI are in increments of 0.1. If $P(I)$ is determined to be greater than 0.5, then the app permits an interruption to occur; otherwise the audio profile will go to silent. When using a weighted average, the weights do not need to sum to one, as the denominator will normalize the function. A weighted sum, on the other hand, requires the weights to sum to one. This is the major difference between a weighted average and a weighted sum.

Having created this model, we then produced a set of scenarios to test the equation. In total, we created fifteen scenarios using different contexts based around different individuals. The scenarios we used, the POI of each context, the corresponding weights, and the desired results can be found in the Appendix. When using this method to evaluate the fifteen scenarios, the model produced the intended result 10 out of 15 times.

5.1.2 Weighted Average with Default Values

Upon receiving the results from the weighted average, we decided to see what would happen if default values were used for non-measurable contexts. This way, all the contexts would be evaluated even if a value was not assigned, or the context was not pertinent to the situation. From looking at the scenarios, measurable contexts generally had a low POI assigned. This led us to believe if the context was not measurable, then its absence indicated a slightly favorable

POI. To test our theory, we assigned each context a default value of 0.6. In the end, this model also produced the intended result 10 out of the 15 times.

5.1.3 Weighted Sum (Using a Ranking Method)

Learning a default value was not the solution to producing a more accurate POI, we decided to attempt a different approach. We noticed undesired results often occurred when different contexts had the same weight values. Instead of allowing for weights to have the same value, we decided we would make the weights sum to one. This changed the model from a weighted average to a weighted sum, as the denominator now summed to one. The model can be seen below.

$$P(I) = P(I)_L * W_L + P(I)_S * W_S + P(I)_C * W_C + P(I)_T * W_T + P(I)_D * W_D$$

To determine the values of the weights we devised a ranking system. As there are five contexts, the most important context to the user when determining whether or not she would want an interruption gets a value of 5. The next most important context gets a value of 4, and so on. To make sure the weights sum to one, these values are divided by 15 (the sum of the integers between 1 and 5, inclusive). In our new model, we also used default values. As all the values now have a direct impact on each other due to the weighting scheme, it only makes sense to use default values if a context is not measurable. However, as it is unknown how a person would react if a context is not measured, the default values were initially set to 0.5. This way the default values will not influence the result one way or the other, unless they are changed by the user.

Evaluating this model, we found it produced the intended results 11 out of the 15 times. While the result was better than the previous model, it still was not optimal.

5.1.4 Weighted Sum with Adjusted Weights

The final variation on this model, and the one we proceeded to implement, incorporates an intuitive change. As the model previously stood, a non-measurable context could have a larger impact than a measurable context depending on the rankings and default values. However, from our scenarios, we determined non-measurable contexts often do not have an effect. To try and solve this problem, we introduced adjusted ranks.

The weights are adjusted by first checking whether or not the context is measurable. Any non-measurable context, regardless of the weight attributed by the user, cannot be allowed to have a higher weight than a measurable context. The weights are readjusted so all the measurable contexts have a higher ranking than the non-measurable contexts. The exact ranks from here are determined by the ranks set by the user.

For further clarification, say the user has ordered the contexts (in order of most important to least important) as follows: driving, contact, location, schedule, time of day. A call comes in and the only measurable contexts are contact and time of day. The model is then readjusted so contact and time of day receive the greatest weights. As contact initially had a higher weight, it would receive the greatest weight. Time of day would receive the next greatest weight. Finally, the remaining context, in order of importance, would be driving, location, and schedule

Evaluating the scenarios with this model produced the correct results 12 out of 15 times, or 80%. These results show progress, but could still be improved. Looking at the scenarios this model evaluated incorrectly provided insight into a couple of situations where the model fails. The first situation is when one context has a considerably higher weight to the user than all the other contexts. An example of this is a business meeting where location, time of day, and schedule indicate to no interruption, but the caller is the boss, making the interruption desirable. A second shortcoming is when a context's POI, besides time of day, is directly dependent on time. For example, a college student may not want their phone to ring in a building during school hours, but after school hours it may be fine.

6. Implementation of the Application

6.1 The Mobile Platform

To implement this application, we needed to first choose a mobile platform for development. The Android platform was chosen for several reasons. For starters, Android is open source. This allowed us access to many of the phone's features, including sensors, native applications, and the audio profile manager. As our application requires access to all of these features, Android was the preferred platform. In addition, as of May 2013, there were over 900 million Android devices activated [1]. The popularity of Android devices provides a large community for feedback on the application.

6.2 Application System Flow

The system flow for this application was based on a general system architecture laid out by Sina Zulkernain et al. [14]. Their work depicted a three tiered model. The first tier continuously collects contextual information, while the second tier stores user preferences. The final tier uses the information from the first two tiers and a tree generator to produce the appropriate audio profile. The system flow of our application closely resembles this model. However, there are a few key differences:

1. Our application only evaluates contextual information upon a call coming in. Prior to a call, no decisions are made about the audio profile.
2. Instead of a Tree Generator to determine the audio profile, our Weighted Sum model is used to determine a POI, which in turn decides the audio profile.

The System Flow of the application can be seen below in Figure 3, with the orange boxes representing the contexts. In this case, the term calendar is used to represent the schedule context.

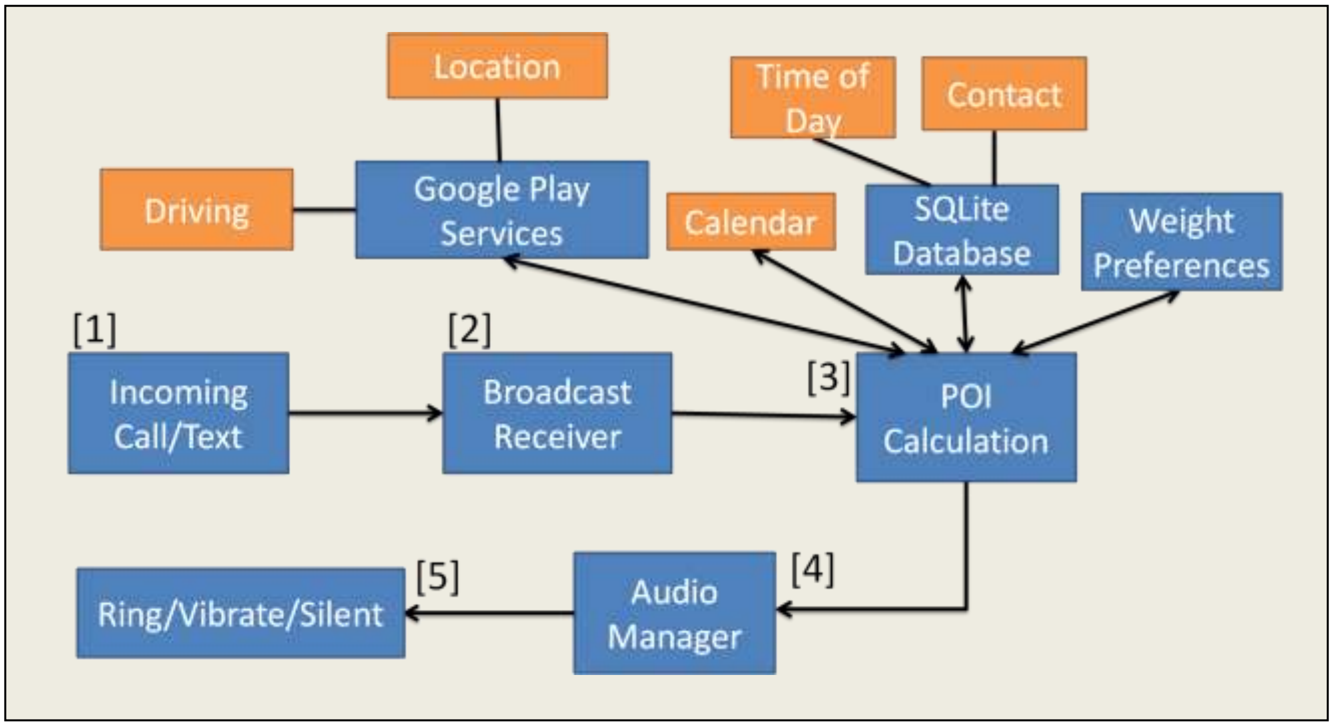


Figure 3: Application System Flow

The application can be divided into five main sections: 1) Incoming Call/Text, 2) Broadcast Receiver, 3) POI Calculation, 4) Audio Manager, 5) Ring/Vibrate/Silent.

1. Incoming Call/Text:

The first stage occurs when the phone receives an incoming call/text, changing the phone state. Android devices allow for three different phone states: idle, off-the-hook, and ringing. A message known as an intent is broadcast when the phone state of an Android device changes. This intent can be received by a component known as a broadcast receiver, assuming the receiver is looking for that particular intent.

2. Broadcast Receiver

The broadcast receiver looks for the intent launched when the phone state is changed. Our broadcast receiver waits to hear an intent advertising the phone is ringing. A ringing phone state indicates a call or text is coming in and launches the POI Calculation.

3. Probability of Interruption (POI) Calculation

The POI calculation uses our Weighted Sum with Adjusted Weights model (see Section 5.1.4). The calculation is a background service and is launched by the broadcast receiver. Evaluation of our model requires obtaining all of the context information. This includes determining whether the context is measurable, the POI for each context, the weight assigned to each context, and the preferred method of interruption (PMI). These values are obtained by querying a database, as well as using saved user preferences. The service then runs the model and determines the audio mode the phone should be in. Changing the audio mode to represent the audio profile is handled by the audio manager.

4. Audio Manager

The Audio Manager provides access to changing the audio mode of the phone (silent, vibrate, ring). After determining the mode the phone should be, the audio manager places the audio profile of the phone into the mode deemed best by the POI Calculation.

5. Ring/Vibrate/Silent

At this point, the phone is now in the audio mode determined by the POI model, and the call or text is presented to the user accordingly.

6.3 Determining the Preferred Method of Interruption (PMI)

If the POI is above 0.5, the application must decide whether it should alert the user with a ring or with a vibration. As a ring is generally more obtrusive than a vibration, we make the assumption if it is acceptable for the phone to ring, then it is also acceptable for the phone to vibrate. However, if a vibration is acceptable, a ring is not necessarily also acceptable. From this logic, we devised a way to determine the user's preferred method of interruption (PMI).

We allow the user to specify a PMI for each context, which by default is set to vibrate. Based on our logic, vibrate takes precedence over ring. This means if all of the measurable contexts have a PMI of ring and, if an interruption is deemed acceptable, the phone will ring. However, if even one of the measurable contexts has a PMI of vibrate, the phone will vibrate.

6.4 Obtaining and Setting Preferences for the Contexts and Weights

Our model requires complete knowledge of all contexts. This means determining if the context is measurable, the probability of interruption for each context, the weight assigned to each context, and the PMI. If the context is not measurable, a default POI is used. Upon installation, all the default probabilities are set to 0.5, though they can be changed later by the user. To acquire this information, the user must specify their preference for all of the contexts. The UI is designed to accommodate quickly adding or altering preferences. An image of the main UI is seen below.

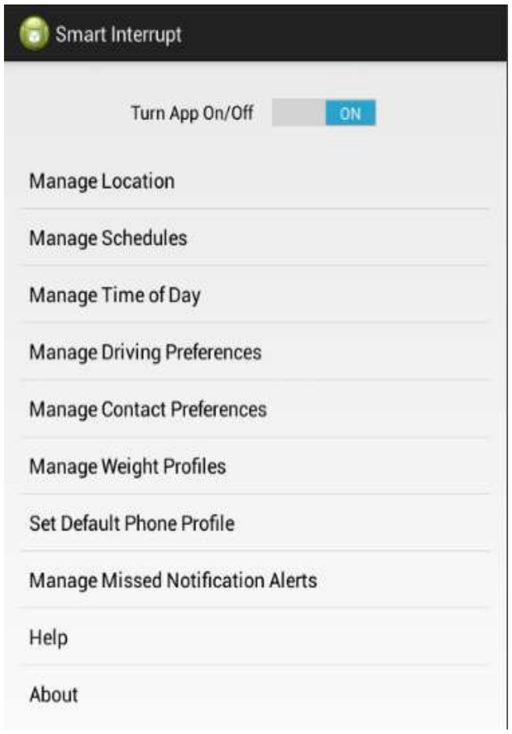


Figure 4: Main Menu

From this main menu, the user can navigate and enter the required information for the six main sections: weight, location, schedule, contact, time of day, and driving. In addition, the user can also set the default phone profile, and configure missed notification settings (See Section 7.2). If none of the contexts are measurable, a default audio profile is used. For the default phone profile, the user is able to choose between ring, vibrate, and silent.

6.4.1 Weight Preferences

The Weight Preferences UI is used to set the ranks of each context. As our model uses a ranking system, the user simply has to list the weights in order from most important to least important. This is done by using the arrow controls to the side of each of the contexts. The selected order is then saved as a user preference. When the POI Calculation service is launched, it retrieves the rank from the saved preferences, and then determines corresponding weight (See Section 5.1.3 for determining weight from rank).

6.4.2 Location Preferences

The Location UI allows the user to create location profiles, and each profile gets its own set of preferences. For example, a user could create a location profile for his work. From here he can specify his work address to be checked against a database of known street addresses for possible matches. If any matches are found, the UI presents these addresses to the user, and the user can select the correct one. For each profile, the user is also able to specify a POI and PMI. A SQLite database is used to store the user's preferences as well as latitude and longitude of the location specified.

To determine the user's proximity to a specified location, our application relies on Google Play Services. In May of 2013, Google released an update to the Google Play Services providing additional APIs for functionality. One of these APIs, the Location Service API, automatically keeps track of the user's current location. The Location Service handles maintaining user location updates without any additional configuration.

Upon a call coming in, the last known location of the phone is obtained. The POI Calculation compares the user's last known location with the locations stored in the user's location profiles. Current technology does not always provide the most accurate location (See Section 9.2). With this in mind, when the POI Calculation compares the two locations, it allows the measurements to be off by a certain margin. This margin is the radius of accuracy, represented in meters, provided by the Location Services when it makes its location prediction. If the two locations are off by a distance less than the radius of accuracy, the context is considered measurable, and the user's preferences are used in the calculation. Otherwise, the context is considered non-measurable and the default values are used.

6.4.3 Schedule Preferences

For the user's Schedule Preferences, our application uses the native calendar application. Several possible options arise when evaluating the schedule context. The first possible outcome is the user has nothing scheduled at the time of the call/text. If this is the case, the context is considered non-measurable and the default probability is assigned. The next outcome is the user has something scheduled, but she does not have a POI or a PMI assigned for that specific event. In this case, the user can set a default POI and PMI for events with no specified preferences. The third and final outcome is if an event occurs and the user has specified both POI and a PMI in the native calendar application.

The POI and PMI are entered into the native calendar application. A specific format must be used when entering the event to allow the application to determine the POI and PMI. This format requires using dollar signs as delimiters within the title of the event. Within the two dollar signs the user must specify the POI, and then the PMI. The probability must be between 0 and 1, and be in increments of 0.1 (e.g. 0.0, 0.1). This is then followed by the PMI, with R representing ring, and V representing vibrate (it is not case sensitive). One example of properly formatted preferences for an event is “Studying $0.1V$”. When a call or text comes in, the application uses the current time to decide if an event is currently scheduled, and evaluates the schedule preferences accordingly.

6.4.4 Contact Preferences

The Contact Preferences UI is very similar to the Location Preferences UI. The user is able to create group profiles, assign contacts to each group, and finally specify preferences for each of these profiles. Each contact is only allowed to be in one group at a time. The contacts name, phone number, and group are stored in a SQLite database, along with the POI and PMI of each group. When the application needs to do its calculations, it queries the database, using the incoming phone number to determine whether or not the contact is in a group. If nothing is returned by the query, the default POI is assigned and the context is considered non-measurable. If the contact is found to be in a group, the POI and PMI are retrieved to be used in the calculation.

6.4.5 Time of Day Preferences

Like both the Location and Contact Preferences, the Time of Day Preference allows the user to create multiple profiles. Each profile gets a start time, an end time, a POI, a PMI and the days of the week the preference is applicable. As the profiles are specified for each day, the user cannot span between one day and the next using the same profile. If this result is desired, the user must specify two profiles. All this information is stored in a SQLite database.

To check whether or not the Time of Day context is measurable, the POI Calculation service does a query of the database. It is looking for a return value where the current time is after the start time but before the end time. The day of the week must also match the systems day of the week. If a result is returned, then the Time of Day context is measurable, and the necessary values are retrieved. Otherwise the context is considered non-measurable and the default value is used.

6.4.6 Driving Preference

The update in May of 2013 to the Google Play Services provided the addition of the Activity Recognition API. Through this API it is possible to request the user's current activity. Among walking, standing still, and other activities, Google Play Services is also able to determine if the user is driving. As it takes several seconds to request an activity update and to get a response, it is not feasible to do this upon the call/text coming in. Instead, this service runs in the background and receives location updates every 30 seconds. The last five activity updates are kept on record, and if driving was one of the last five activities, a shared Boolean value is saved as true. While this may not be the most accurate method, the application is trying to balance accuracy with battery usage.

When the POI Calculation is run, it determines whether or not the user is driving from the saved Boolean value. The POI for driving, as well as the PMI, can be set from the Driving Preferences UI. If the driving context is evaluated as non-measurable, a default POI is used. As the user may not always desire the phone to continuously track their activity and use battery resources, it is possible to stop the activity recognition. Stopping the updates from activity recognition means the driving context will always be evaluated as non-measurable and get the default value.

7. Missed Notification Feature

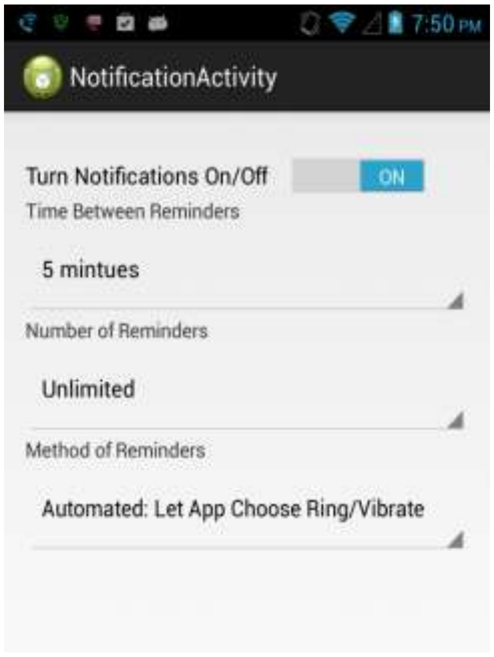
An application that prevents unwanted interruptions increases the possibility a user will miss attempted notifications. If, for example, the phone is placed in silent mode, it may be awhile before the user checks their phone again. A missed notification service was built into the application to alert the user, at an appropriate time, he has missed a notification. The following section describes what this feature does, how it works, and the benefits of the service.

7.1 What the Missed Notification Feature Does

The missed notification feature alerts the user to missed messages (calls/texts). The feature does not require the phone to have been in silent mode when the notification came in, only that the user has yet to recognize the missed message(s). To alert the user, the phone can either make a sound or vibrate. Of course, as any alert can be considered an unwanted interruption, the phone uses the POI calculation to determine whether or not an interruption would be appropriate, and how to interrupt the user. In addition, a message is displayed on the screen to convey visually the user has missed a call or text.

7.2 Missed Notification Configuration

Various settings can be configured by the user to specify how often she wants to be notified, and how many times she wants the alert to continue. In addition, the user can choose if he wants the application to determine the method of interruption, or if he only wants to be interrupted through vibration. There is also the option of turning off the missed notification feature (Figure 5 shows the missed notification feature UI).



Turn Notifications On/Off

Time Between Reminders

5 mintues

Number of Reminders

Unlimited

Method of Reminders

Automated: Let App Choose Ring/Vibrate

Figure 5: Missed Notification UI

7.3 Missed Notification Implementation

As the notification feature was added to the already implemented application, much of the structure and code is the same (a basic activity flow for the missed notification service can be seen in Figure 6). As can be seen, a strong similarity exists between the notification feature's activity flow, and the application's system flow (depicted in Figure 3).

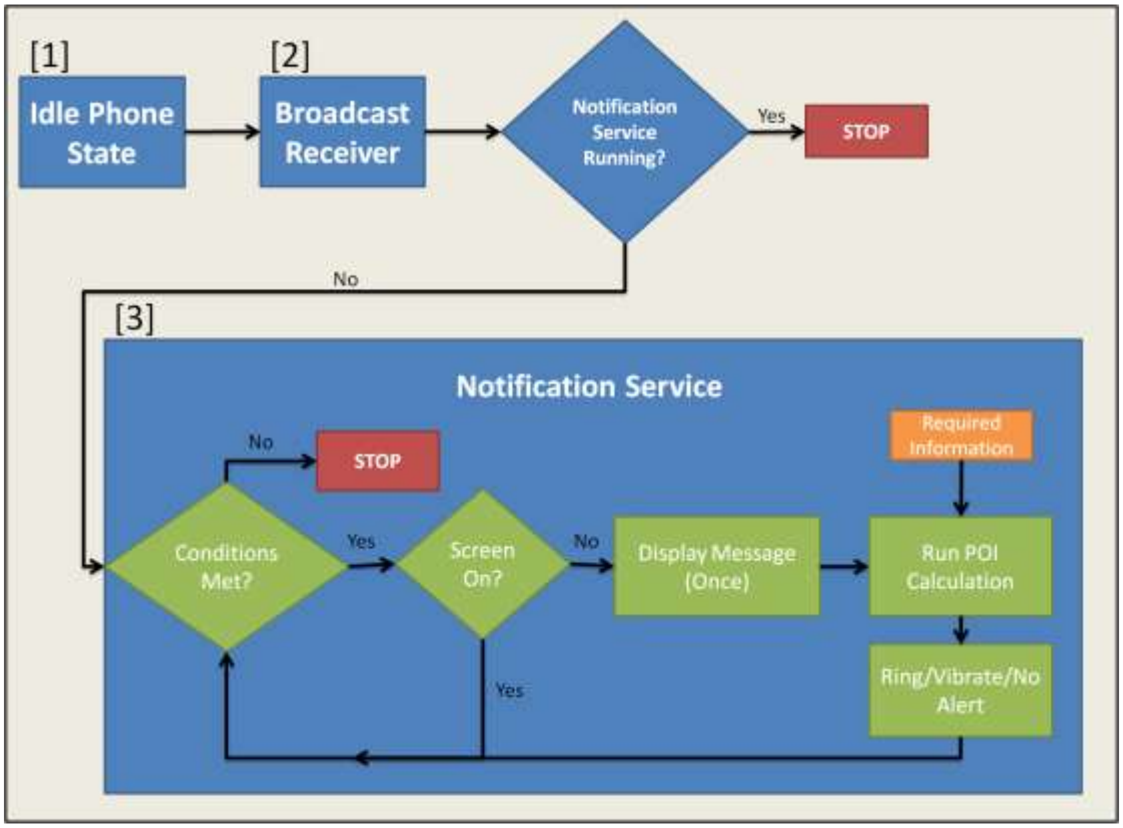


Figure 6: Notification Service Activity Flow

The activity flow for the notification service can be divided into three main sections: 1) Idle Phone State, 2) Broadcast Receiver, 3) Notification Service. In the diagram above, the logic of the Notification Service is expanded to better demonstrate how it serves its function.

1. Idle Phone State

Android devices allow for three different phone states: idle, off-the-hook, and ringing. The only time a call or text could have been missed is when the phone state changes to idle. As described in Section 6.2, when an Android device changes states, it broadcasts an intent. A broadcast receiver is listening for this specific intent.

2. Broadcast Receiver

The broadcast receiver used to receive the idle phone intent is the same receiver used to launch the POI calculation when a call comes in. This receiver is listening for any phone state change, and through conditional statements, determines the phone state the device was changed to. In the event the phone state went to idle, the broadcast receiver checks if the notification service is already running. If the notification service is running, the service does not need to be launched again. However, if the service is not running, the broadcast receiver launches the missed notifications service.

3. Notification Service

The diagram in Figure 6 expands the notification service to show its logical flow. The notification service is comprised of five main steps (represented by the green boxes) to appropriately alert the user of a missed notification.

Step 1: Before the Notification Service can be run, five conditions must be met:

1. The application must be turned on,
2. the notification service must be turned on,
3. a call or text must be marked unread since the last time the user acknowledged the missed notification message,
4. the service must not have run more times than the number specified by the user, and
5. the notification message must not have been okayed/cancelled.

While these five conditions remain true, the notification service will continually run at intervals specified by the user. As soon as one of these conditions no longer holds, the service terminates.

Step 2: Check to see if the screen is on.

If the screen is on, the notification service will not alert the user of missed notifications, as the Android OS already has a status bar indicating missed notifications. If the screen is on, the service does not terminate, but instead is run again after the specified interval.

Step 3: Display missed notification message (once)

If our application's missed notification message is not already displayed, the notification service will display a dialog message on the screen. The message visually alerts the user about a missed notification, and will terminate upon the user selecting an okay button on the dialog message. Accepting the message will terminate the notification service, and the time the message was acknowledged will be used in the future to see if missed messages have occurred since the user last recognized she had missed notifications.

Step 4: Run the POI calculation

After checking to see if the necessary conditions hold, and if the message has been displayed, the POI calculation is run. The orange box title "Required Information" in Figure 6 represents the information needed to run the POI calculation, and is equivalent to the information depicted in the orange boxes in Figure 3. Running the POI calculation determines whether or not it is acceptable to notify the user with an interruption, and if so, how to interrupt the user.

Step 5: Ring/Vibrate/No Alert

After running the POI calculation, the phone can ring or vibrate to alert the user of a missed notification, or it can decide not the alert them at all. If the phone alerts the user, then it will decrement the number of times it will continue to alert them. After this step, the entire notification service is run again at the interval set by the user.

7.3 Benefits of the Missed Notification Service

As our application deals with preventing unwanted interruptions, a notification service seems like a logical addition. A user may be unaware of a missed notification, especially if the audio profile was placed to silent. Our application also has the benefit of using our POI calculation to first try to determine whether it is an appropriate time to alert the user of missed notifications.

By using our POI calculation to determine if it is an appropriate time to interrupt the user, our application has a clear benefit over missed notification features not providing interruption management (i.e. native Android OS missed notification features and third party missed notification applications) . Currently, very few applications on the Android app store provide notification services paired with interruption management. Of the interruption management applications we analyzed in Section 4.1 (all of which used binary decisions), only two provided a missed notification service: “Silencify: Smart Silence”, and “Tasker”.

8. Application Status and Future Work

8.1 Status

Currently, the application is in a functioning state; it can obtain all the information needed to determine the POI, and from the POI, it can appropriately change the audio profile. The application also used the derived POI calculation to issue appropriate missed notification alerts. However, the application still has some functionality issues that will need to be taken care of before public release. In addition, before public release, it would be best to change some of the ways the application obtains user information. The schedule context, for example, could use key words to obtain POI instead of delimiters. The location context could be improved by using a map system instead of a street address. These changes would not impact the model, but simply improve the interaction with the user.

8.2 Current Application Issues

Currently, the application's biggest issue occurs when the audio profile is changed from vibrate to ring. When a call comes, in instead of changing instantaneously from vibrate to ring, it vibrates a couple of times before starting to ring. Then it continues to vibrate while it rings. In addition, when changing from vibrate to silent the phone makes a small vibration before changing profiles. This issue only occurs when a call comes in. We believe the issue stems from the call being put through at the same time our application is being run.

8.3 System Usage

Only preliminary testing has been done on the application, and the only device we have used for testing is the Sony Xperia U. In addition, this testing was done prior to the implementation of the missed notification service, though we believe the additional service will not have a significant impact on the system. In the future, we look to do future testing to further verify our results.

Currently, the application is 2.54 MB, with the source code being 174KB. The application runs constantly and generally uses 7MB-18MB of RAM. The application does not have an impact on the CPU when running in the background. Upon initially being launched for the user to enter values, the application uses 2-3% of the CPU. When doing calculations the application uses 1-2% of the CPU.

While we do not have any concrete statistics regarding battery usage, our application appears to significantly drain the battery. The constant connection to the Google Play Services to retrieve location updates and to get activity updates requires a lot of energy. However, if a different application were already using Google Play Services for constant location and activities updates, our application would cause no additional overhead.

8.4 Future Work

The most important future work in respect to this research is finishing the application and getting it published. The main goal of this research was to produce the first widely available interruption management application using a probabilistic method. While great progress has been made towards this goal, it still has yet to be reached. Production of this application has the potential to be extremely valuable in the realm of interruption management.

To date, besides feedback from small simulations and trials, no interruption management application using a probabilistic model has been tested by a large population. Our research is meant to decrease the burden cell phone interruptions have on society. The ultimate test is the degree of success it has on a variety of users. With this feedback, the application can be re-evaluated and modified to produce an even better application.

Another area of future work for this particular research is to compare the effectiveness of our model to other COI models. The comparison could show possible weaknesses within our model. In addition, using our application as the framework, we could see how easy or difficult it would be to implement other COI models.

9. Limitations of Our Application and Current Technology

9.1 Limitations of our Application

Our current application has several limitations. The first is the limited success of our model during testing. While it correctly predicted the desired outcome of 12 of 15 scenarios, this still leaves a large margin for error. While this basic model provided an effective foundation for building an application, it will probably need to be modified or replaced by a more accurate model.

Another limitation of the current model is the necessity of the user to enter all the information needed for the model to function properly. While weights only have to be entered once, our model requires the user to enter all of the other information for each context. This can become especially tedious for contexts with multiple profiles. While the amount of information required is less than an application using binary trees for a decision (See Section 2.1), the user still must specify all the judging criteria for the final decision. In addition, the final decision is made by the phone and not the individual. This leads to the next limitation.

As the application stands, the user is unable to make manual profile changes on the fly. Say, for instance, the user is about to go into a business meeting and remember to silence their phone. As our application makes its decision upon the call coming in, it could potentially change the profile to ring. For the user to manually change the profile, he must turn off our application and then chose the desired profile. While this is not a difficult task, it could lead to the user forgetting to turn the application back on (analogous to users forgetting to go to the correct mode in the first place), rendering our application useless.

Another limitation with the current design of our application, and perhaps mostly due to the UI, is its complexity. The way the application is run is hidden from the user. Generally this is fine: if the application does its job, why get into the complexities of it? However, a valid evaluation of the model relies on the user providing accurate information. Our methods of obtaining information might be confusing to many users, as they may not understand how different POI and PMI values affect the outcome.

9.2 Current limitations of technology

Technology provides us with cellular phones with the capability to act as personal computers. However, even with advances in cell phones, limitations still exist. The most noticeable limitations in regards to our application come from inaccuracies when retrieving sensor data. In our model, the location and driving contexts rely on sensor readings.

Our application relies on being able to determine the user’s locations. In order for the phone to determine the location, it relies on access on to several different sensors and features. These sensors and features include a cell tower connection, GPS, and Wi-Fi. However, each feature can only be so accurate. For example, GPS can be accurate up to several meters [5]. Wi-Fi accuracy can be around 200m or better [5]. Finally, if the user only has access to a cell tower, the accuracy may be approximated at distances up to several thousand meters [5]. While the user probably has a cell tower connect, it may be likely the user has the other features turned off. This can lead to inaccurate measurements.

Our application incorporates one other context reliant on sensors: whether or not the user is driving. From different sensors on the phone, the Activity Recognition service provides a best estimate of the user’s activity. However, it is only an estimate. Initial testing showed the Activity Recognitions service was able to correctly determine if the user was driving, though it took around five minutes of driving before the activity was recognized. In addition, the service cannot determine the user’s activity instantaneously. When determined at the fastest rate possible, it still takes around five seconds to update. Our application does not receive updates as quickly as possible to save on battery life. This leads to the next limitation.

While continuous access to sensors provides the best accuracy, it comes at the cost of battery life. The user may be able to frequently charge their phone, but it should be assumed she wants the best battery life possible. Continuously using sensors and requesting updates require a lot of energy, and can deplete the battery quickly. As technology advances, battery life may be improved to support continuous sensing, but currently our application tries to moderate between accuracy and battery life.

10. Future Research

One of the largest limitations of our application is the requirement for the user to enter all of the information for the model. The application would be much simpler to operate if these values could be obtained without constant user input. Future research could be done to determining the most accurate and efficient ways of finding the probabilistic values used in these probabilistic models.

One possible way of determining the probabilistic values could be through machine learning or data mining. [4] devised a learning based algorithm that uses previously acquired information and user input to make decision on the audio profile. By recording when the user changes their audio profile, or asking whether or not preventing an interruption was the correct decision, it may be possible to adjust the POI values of each context to better suit the individual.

Another area of possible research would be to incorporate other methods of interruption. While our research took into account ring, vibrate, and silent, other technology is currently available that expands upon these audio profiles. Pairing a device with the phone, such as a Bluetooth headset, or a smart watch, could change the user’s preference of whether or not he would want an interruption.

Modern mobile devices are also susceptible to interruptions caused by services other than calls and texts. E-mails and social networking updates are just a couple of examples of alerts vying for the user’s attention. Using the same model, our application could be expanded to incorporate notification management beyond calls and texts.

For this research, we focused on smartphones, though interruption management is not limited these devices. Interruption management can be implemented on any context aware device. Examples include laptops and tablets. The model we implemented, along with the system flow, could be ported to these devices. Instead of managing telephony, the app could handle different type of interruptions, such as an e-mail. By going through the same system flow and same framework, it may be possible to limit unwanted interruptions in other digital devices as well.

11. Conclusion

With this research, we have made significant progress on an application looking to be the first widely available interruption management application using a probabilistic approach. First, we determined which contexts are measurable and important to an interruption management model. After determining our contexts, we compared our proposed application with currently available applications. We explained the shortcoming of the currently available binary models, and show how our model looks to incorporate all the same contexts as these models. From the contexts, we produced a basic model capable of determining probability of interruption. We then succeeded in designing and implementing a bare-bones application that probabilistically determines a smartphones’s appropriate audio profile.

The production of this application has provided steps, measurable contexts, a system flow and an application that future interruption management researchers can use, modify or compare to new models. In addition, the production of this application has provided insight into its limitations; the most apparent is requiring the user to provide all the values needed for each context. Future research should look to implement applications capable of acquiring contextual values for the user without relying solely on user input. The limitations of modern day technology, such as inaccurate measurements and battery constraints, should also be considered when designing an interruption management application.

While this application is still in development, upon completion and publication it will it be possible to receive user feedback. This feedback could prove extremely valuable to the topic of interruption management as it will provide insight on the application’s effectiveness from its target audience. As no widely available application has been released, such information has not been available.

Finally, with this application, it is easy to see future work in the area of interruption management. Being able to determine probabilities of interruption for each context without relying on user input would make the application more reliable and user friendly. Currently, the complexities of the application could discourage users from actually using it. One possible way of obtaining these values could be through machine learning and data mining techniques. In addition, the steps, model and system flow for this smartphone application may be able to be used on other context aware devices, such as laptops and tablets. By preventing unwanted interruptions we can hopefully provide benefits to society, such as more efficient employees, safer roads, and a more productive school atmosphere.

References

[1] Bort J., “Google: There Are 900 Million Android Devices Activated,” May. 2013; http://www.businessinsider.com/900-million-android-devices-in-2013-2013-5

[2] Bureau of Labor Statistics, “Economic News Release,” March 2014; http://www.bls.gov/news.release/empsit.t19.htm

[3] Burns S.M and Lohenry K.(2010, Sept).Cellular phone use in class: implications for teaching and learning a pilot study. *College Student Journal* [Online]. 44(3), pp. 805-810.

[4] Dekel A., Nacht D. and Kirkpatrick S., “Minimizing Mobile Phone Disruption via Smart Profile Management”, presented at the MobileHCI Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services, Bonn, Germany, Sept 15-18, 2009, Article No. 43.

[5] Google, “Location source and accuracy,”; https://support.google.com/gmm/answer/81873?hl=en

[6] Governors Highway Safety Association, “Distracted Driving Laws,” Mar. 2014; http://www.ghsa.org/html/stateinfo/laws/cellphone_laws.html

[7] Horvitz E., Koch P. and Apacible J., “BusyBody: creating and fielding personalized models of the cost of interruption,” presented at the ACM Conference on Computer Supported Cooperative Work, Chicago, IL, Nov 6-10, 2004, pp. 507-510.

[8] International Telecommunication Union, “ITU relesease lastest global technology development figures,” Feb. 2013; http://www.itu.int/net/pressoffice/press_releases/2013/05.aspx#.UyfEN_ldWuI

[9] National Highway Traffic Safety Administration, “Breakthrough Research on Real-World Driver Behavior Released,” Apr. 2006; http://www.nhtsa.gov/Driving+Safety/Distracted+Driving/Breakthrough+Research+on+Real-World+Driver+Behavior+Released

[10] Spira J.B. and Feintuch J.B., “The Cost of Not Paying Attention: How Interruptions Impact Knowledge Worker Productivity”, Basex, New York, NY, 2005

[11] Turney P., “Robust Classification with Context-Sensitive Features,” presented at the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 1993.

[12] Vilwock W., Madiraju P. and Ahamed S.I. "A System Implementation of Interruption Management for Mobile Devices", Proceedings of the 2013 IEEE 16th International

Conference on Computational Science and Engineering (CSE 2013), Sydney, Australia, December 3-5, 2013 pp.181-187

[13] Zulkernain S., “Modeling Cost Of Interruption (COI) To Manage Unwanted Interruptions For Mobile Devices” M.S. thesis. Marquette University, Milwaulkee, WI, 2011.

[14] Zulkernain S., Stamm K., Madiraju P. and Ahamed S.I. (2010, Jan). A Mobile Intelligent Interruption Management System. *Journal of Universal Computer Science*. 16(15), pp. 2060-2080. Available: http://www.jucs.org/jucs_16_15/a_mobile_intelligent_interruption/jucs_16_15_2060_2080_zulkernain.pdf

Appendix

The Fifteen Scenarios

The 15 scenarios used to test the models are depicted here. Scenarios 1 and 2 are unique scenarios. Scenarios 3 and 15 use the same subject, but change who is calling them. Scenarios 4-7 represent a single subject, but the contexts surrounding the subject change in each scenario. Scenarios 8-14 represent a single subject as well. As in scenarios 4-7, the context surrounding the subject change with each scenario.

Each context has a weight and a probability of interruption assigned to it. The explanation of how weight and probability are used in our model is discussed in Section 5.1. As our model developed over time, we produced two ways of representing weight. The weight column values were used for the weighted average models (See Sections 5.1.1 and 5.1.2), and the values could be any value between 0.0 and 1.0. The values in the rank column were used in the weighted sum models (See Sections 5.1.3 and 5.1.4). The contexts are ranked in order of importance, with 1 being the most important, and 5 being the least. Each rank corresponds appropriately to a weight in the weight column (e.g. the context with the greatest value in the rank column would receive a rank of 1). To see how these ranks are translated into a weight to be evaluated in the weighted average models, see Section 5.1.3. A $P(I)$ value of N/A in the table means the context is not measurable.

Scenario 1: Henry is at school in class. It is in the afternoon and Henry's class is having a study session. Everyone is quiet, and that is the way they would like the room to stay. Right in the middle of class Henry receives a call his mother.

Context	**Rank**	**Weight**	**P(I)**
Location	3	0.7	0.3
Schedule	4	0.6	0.1
Contact	2	0.8	0.8
Time of Day	5	0.6	N/A
Driving	1	0.9	N/A

Desired Result: No Interruption

Scenario 2: Julia is a business executive. She is in the middle of a meeting when her boss calls. There is something very important he needs to tell her about the meeting.

Context	**Rank**	**Weight**	**P(I)**
Location	2	0.7	0.2
Schedule	3	0.6	0
Contact	1	1	1
Time of Day	4	0.6	N/A
Driving	5	0.5	N/A

Desired Results: Interruption

Scenario 3: Brock is driving along the highway, and unless the caller is important he prefers not to be disturbed while driving. An unknown caller calls.

Context	**Rank**	**Weight**	**P(I)**
Location	3	0.7	N/A
Schedule	4	0.6	N/A
Contact	2	0.8	0.6
Time of Day	5	0.6	N/A
Driving	1	1.0	0.2

Desired Result: No interruption

Scenarios 4-7: These scenarios revolve around one subject, Bill the college student. The weights and rank remain the same for these scenarios, though the $P(I)$ for the contexts change.

Scenario 4: Bill is in class. No other events are scheduled, besides him simply being in class. His mother calls him. It is 2 pm.

Context	**Rank**	**Weight**	**P(I)**
Location	3	0.8	0.1
Schedule	2	0.9	N/A
Contact	4	0.7	0.9
Time of Day	5	0.6	0.4
Driving	1	0.9	N/A

Desired Result: No Interruption

Scenario 5: It is 7 pm and Bill is studying in one of the campus buildings. He has nothing scheduled, and he does not mind being interrupted from his studies. His brother calls him.

Context	**Rank**	**Weight**	**P(I)**
Location	3	0.8	0.1
Schedule	2	0.9	N/A
Contact	4	0.7	0.9
Time of Day	5	0.6	N/A
Driving	1	0.9	N/A

Desired Result: Interruption

Scenario 6: It is 7 pm and Bill is studying in one of the campus buildings. He has nothing scheduled, and he does not mind being interrupted from his studies. His brother calls him. Bill also has considered this time of day, and after 5 p.m. he is okay with interruptions.

Context	Rank	Weight	P(I)
Location	3	0.8	0.1
Schedule	2	0.9	N/A
Contact	4	0.7	0.9
Time of Day	5	0.6	0.7
Driving	1	0.9	N/A

Desired Result: Interruption

Scenario 7: Bill is in his dorm, studying, it is 11:00 o'clock at night. Being interrupted is okay, as long as it is by someone he knows. An unknown caller calls

Context	Rank	Weight	P(I)
Location	3	0.8	N/A
Schedule	2	0.9	N/A
Contact	4	0.7	0.6
Time of Day	5	0.6	0.3
Driving	1	0.9	N/A

Desired Result: No interruption

Scenarios 8-14: These 7 scenarios revolve around John, a business man. As it is the same individual, all the weights and rank remain the same while the $P(I)$ for the contexts change.

Scenario 8: John gets a call at 5 a.m. from a co-worker, explaining to him the boss is in a bad mood and to please bring the report information not thought to be needed until Friday.

Context	Rank	Weight	P(I)
Location	3	1.0	N/A
Schedule	2	1.0	N/A
Contact	1	1.0	0.9
Time of Day	4	0.6	0.2
Driving	5	0.4	N/A

Desired Result: Interruption

Scenario 9: John gets to work. He is in his office, doing simple paper work. His wife calls and wants to know if he remembered to pack the kids' lunch. It is 9:00 a.m. and before he has any meetings.

Context	Rank	Weight	P(I)
Location	3	1.0	0.1
Schedule	2	1.0	N/A
Contact	1	1.0	0.9
Time of Day	4	0.6	0.4
Driving	5	0.4	N/A

Desired Result: Interruption

Scenario 10: John is in a meeting and a telemarketer with an unknown number calls.

Context	**Rank**	**Weight**	**P(I)**
Location	3	1.0	0.1
Schedule	2	1.0	0.0
Contact	1	1.0	0.7
Time of Day	4	0.6	0.4
Driving	5	0.4	N/A

Desired Result: No Interruption

Scenario 11: John is in the meeting and his boss sends him a text pertinent to the current meeting.

Context	**Rank**	**Weight**	**P(I)**
Location	3	1.0	0.1
Schedule	2	1.0	0.0
Contact	1	1.0	1.0
Time of Day	4	0.6	0.4
Driving	5	0.4	N/A

Desired Result: Interruption

Scenario 12: John is now in a 3 p.m. meeting and is working with co-workers. One of his fellow co-workers who could not make the meeting calls in to see how things are going.

Context	**Rank**	**Weight**	**P(I)**
Location	3	1.0	0.1
Schedule	2	1.0	0.4
Contact	1	1.0	0.9
Time of Day	4	0.6	0.4
Driving	5	0.4	N/A

Desired Result: Interruption

Scenario 13: John is now off from work and driving home. His wife calls and wants him to pick up a pizza.

Context	**Rank**	**Weight**	**P(I)**
Location	3	1.0	N/A
Schedule	2	1.0	N/A
Contact	1	1.0	0.9
Time of Day	4	0.6	0.7
Driving	5	0.4	0.4

Desired Result: Interruption

Scenario 14: Its 12:00 a.m. John is sleeping. A telemarketer calls.

Context	**Rank**	**Weight**	**P(I)**
Location	3	1.0	N/A
Schedule	2	1.0	N/A
Contact	1	1.0	0.6
Time of Day	4	0.6	0.2
Driving	5	0.4	N/A

Desired Result: No Interruption

Scenario 15: This is the same as #3. However, now Brock's boss calls instead of an unknown caller.

Context	**Rank**	**Weight**	**P(I)**
Location	3	0.7	N/A
Schedule	4	0.6	N/A
Contact	2	0.8	1.0
Time of Day	5	0.6	N/A
Driving	1	1.0	0.2

Desired Result: Interruption

Results of the various weighted sum/weighted average models

Scenario	**Standard**	**Default(0.6)**	**Weighted Sum**	**Weighted Sum w/Adjusted Weights (our model)**
1	X	X	X	X
2		X		
3	X	X	X	X
4	X		X	X
5		X		
6	X	X		X
7	X		X	X
8	X	X	X	X
9		X	X	X
10	X	X	X	X
11				
12			X	X
13	X	X	X	X
14	X		X	X
15	X	X	X	x
Total:	10	10	11	12