A System Implementation of Interruption Management for Mobile Devices

William Vilwock
Department of Computer Science,
Carroll College, Helena, MT, USA
wvilwock@carroll.edu

Praveen Madiraju and Sheikh Iqbal Ahamed
Department of Mathematics, Statistics and Computer Science
Marquette University
Milwaukee, Wisconsin, USA
{praveen.madiraju, sheikh.ahamed}@marquette.edu

***Abstract*— As the number of worldwide cellular subscriptions approaches the world's population, the negative effects of cell phone disruption have become increasingly apparent. With advances in mobile phones, specifically their sensor technology, mobile phones are now capable of moderating interruptions based on whether or not the user would want an interruption. Research into the area of interruption management has provided models and architectures 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 that utilize a probabilistic model to moderate cell phone interruptions. A probabilistic model would be an improvement over current binary decision models as the user would not need to predetermine every possible outcome. In this project, we have used a probabilistic model to implement an interruption management system for Android OS 4.0 which utilizes five contexts: schedule, time of day, location, caller relationship, and driving. Our system intercepts the call, calculates the probability of interruption, and then changes the phone's audio profile to vibrate, silent, or ring based on our model. Our performance evaluations indicate minor application foot print size, reasonable battery consumption, and very little time overhead for the application.**

Keywords- interruption management; Probability of Interruption (POI); mobile application

I. INTRODUCTION

Interruptions are a common part of everyday life. They can be necessary, such as a fire alarm going off to warn of danger. 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 things, 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 that by 2014 the global number of cell phones subscriptions will hit 7 billion, closing in on the number of people who inhabit the planet [10]. 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, it is apparent of the need to manage their interruptions.

The goal to produce an interruption management system has been the center 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 application to facilitate interruption management. An important aspect of many of these research projects is the ability to measure Cost of Interruption (COI) [1,7,9,11]. COI is a measurement of the cost an interruption has on a user. Related to COI is POI, or Probability of Interruption. POI is the probability an individual would want to be interrupted. Both can be used to determine whether or not an interruption would be desired by the individual. For this research we will be using POI.

As cell phones have advanced, they have been embedded with sensors that can comprehend their surroundings. By using these sensors with other features on the phone it is possible to understand the state of the user, which can be used to calculate POI. Despite the amount of research done on interruption management, to our knowledge, a widely available application has not been released that utilizes a POI or probabilistic method. Our application looks to be the first to accomplish this task.

In previous work [18], we have proposed a mathematical model for calculating COI based on Dempster-Shafer [5, 13] theory, taking into account uncertainty of context information. In [17], we proposed a system architecture which takes user preferences and relevant context as input, and then produces whether or not the phone should be allowed to ring as output. The system used binary decision tree based implementation of different scenarios of contexts. In this paper, we propose a probabilistic model for calculating the POI. The system takes an incoming call as input, calculates the probability of interruption, and based on POI alters the state of the phone to vibrate, ring, or silent. We report on our experience with implementing the application on the Android platform.

The rest of the paper is organized as follows. Section II provides motivation for interruption management research. Section III provides a survey of related researches. Section IV provides our model for calculating probability of interruption

(POI). In Section V we discuss the implementation and performance evaluation of the system. The last section concludes our results and paves the way for future work.

II. MOTIVATION

In a society filled with cell phones, it is easy to see the motivation behind managing their interruptions. We have all experienced a time when an untimely cell phone interruption has had a negative impact. It becomes even clearer when seeing statistical impacts cell phones have on society as a whole. To display their negative impact we explore several spheres that are particularly influences by cell phone disruption: business, education, and driving.

In the business sphere, cell phones have been shown to be a significant disruption. Undesirable interruptions, such as a cell phone ringing, take up 28 percent of a knowledge worker's day [14]. In a year, that totals to 28 billion hours wasted and results in a total loss of over 650 billion dollars, considering the average labor rate of $24 an hour [2]. Productivity is clearly a victim 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 that around 80 percent of crashes, and 65 percent of near-crashes, involved distraction of the driver within 3 seconds of the crash. The most common distraction for drivers was found to be cell phones [12]. The distraction cell phones cause is so great that eleven states in the USA 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 that 85.1% of students believed ringing phones were a distraction, and 72.3% stated that they have had their phone go off in class [3]. With a high percent of students believing cell phones to be disruptive, and a high percent of students who admit to having their phone go off in class, it is clear to see the problems of cell phone disruptions in an educational setting.

III. RELATED WORKS

Figure 1 displays a comprehensive selection of applications (searchable by their name in the Android/iPhone app store) currently available that do interruption management. The list displays the application's name, the platform it runs on, and the contexts it can measure. 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 incorporates all of them.

In addition, none of these applications utilize a probabilistic method to make the decision. Instead, they are all binary decisions based on user input beforehand, which requires the user to predetermine all possible situations. Previous research also provides interruption models based on Bayesian Probability and Dempster-Shafter theory.

Limitations of Binary Decisions

The limitation of using binary decisions to determine the interruption profile is the fact that the user must predetermine every possible outcome. For example, say one of these currently available applications makes decisions based on three contexts: location, schedule, and contact. In order to function correctly, the individual must specify how the phone profile should act given each context, and every combination of the contexts. This can become tedious, and it is not logical for the user to predetermine every possible situation they will encounter. This is the advantage of using a Probability of Interruption (POI) model. A POI model does not require the user to predetermine every possible outcome. Instead, it judges the relevant criteria to produce the probability a 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 the next Section.

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	Android	X	X		
Silent Driver	Android				X
Silent Sleep	Android	X			
Smart Silencer	Android	X	X		
Smart Silent Beta	Android		X		
Smart Volume Profile Manager	Android	X		X	
SuperSmartPhone	Android	X			
Tasker	Android	X	X	X	
TextArrest	Android	X			
Timeriffic	Android	X			
Our Application	Android	X	X	X	X

Figure 1: Current Interruption Management Applications

Bayesian and Dempster-Shafer Model

In previous research, Bayesian Probability models were often used for interruption management [8,15]. However, as we discussed in our earlier works [17], Bayesian models have several limitations. The first is they require having complete knowledge of a system, including all the a priori and conditional probabilities. This information can be very difficult to determine beforehand. Also, a priori probabilities are traditionally measured from empirical data or uniform distribution, which is not always available. For these reasons a Bayesian Probability method was not pursued.

The solution that was offered in [18] was an implementation of the Dempster-Shafer theory [5, 13]. Using this theory, it is not necessary to know the a priori and the conditional probabilities. In addition, this model takes into account uncertainty. Uncertainty arises from not knowing or not having access to all the information about a user or context. As there are many factors that 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, in this current work, we assume that probability of contexts is available, and if they are not, we simply assign a default value (see Section 4.4).

IV. OUR MODEL FOR PROBABILITY OF INTERRUPTION (POI)

Here we use a weighted sum of the probabilities of different contexts to calculate the probability of interruption.

The Process of Determining the Weighted Sum Model

A weighted sum/average provides either a sum or an average for a group of values that do not contribute 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 they are 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 has a different degree with which they influence the overall POI for the individual. In other words, a weight could be assigned to each context. The overall POI is then a sum of the POI of each context, times its respective weight. To get to our weighted sum model, we first started out with a weighted average.

4.1 Weighted Average

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

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 that the information for the particular context is obtainable. This requires the POI for the context to be set before hand by the user, and for the context to be relevant to the situation the user is in. If the user is not driving, for example, the driving context is considered not measurable.

As this is a weighted average, the function is divided by a sum of the weights attributed to each context. This function also requires the POI and weight of each context 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, than it is okay for an interruption to occur. If it is less than 0.5, then the interruption profile will go to silent.

A set of scenarios was produced to test this model. In total, we created 15 real life scenarios that looked to utilize different contexts, as well as incorporate values from different types of individuals. The scenarios we used, the POI of each context, the corresponding weights, and the results can be found in the extended version of the paper available online at [16] (we are not presenting here, because of space constraints). When using this method to evaluate the 15 scenarios, the model produced the intended result 10 out of the 15 times. While a good start, we believed a better model could be created.

4.2 Weighted Average with Default Values

Upon receiving the results from this method we decided to see what would happen if a default value of 0.5 was used for non-measurable contexts. This way all the contexts would be evaluated regardless of being assigned a value, or being pertinent to the situation. In the end, this model also produced the intended result 10 out of the 15 times.

4.3 Weighted Sum (Using a Ranking Method)

Learning that a default value was not the solution to determining POI, we decided to attempt a different approach. One thing we notice was that undesired results often occurred when different contexts had the same weight values. Instead of allowing weights to have the same value, we decided we would make the weights sum to one. This then changed the model from a weighted average to a weighted sum, as the denominator now summed to one. The modified 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 5 contexts, the most important context to the user (when determining whether or not they would want an interruption) will get a value of 5. The next most important context will get a value of 4, and so on and so forth. To make sure the weights sum to one we divide these values by 15 (the sum of the integers between 1 and 5, inclusive). In our new model, we also use 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 are initially set to 0.5. This means unless the user changes the default values, the values will not influence the result one way or the other. Evaluating this model, we found it produced the intended result 11 out of the 15 times. This is an improvement on the previous model, yet could still be improved.

4.4 Weighted Sum with Adjusted Weights

The final variation on this model, and the one we proceeded to implement, is an intuitive change. As the model previously stood, if you had a context with a particularly high or low POI, and it had a low rank, it would not have much of an impact. This could throw off the model, especially if the individual had programmed default values. It did not make sense for a non-

measurable context to receive more impact than a measurable one. This led to the inclusion of 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 first readjusted so all the measurable contexts have a higher ranking than the non-measurable contexts. The exact ranks from here are then 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 for which the 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 was initially ranked higher, it would receive the greatest weight. Time of day would then receive the next greatest weight. Finally, the remaining contexts, 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 are not bad, but could still be improved. The scenarios this model evaluated incorrectly provide 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 suggest no interruption, but the caller is the boss, making the interruption desirable. The 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.

V. SYSTEM IMPLEMENTATION

We implemented the application on the Android platform, which was chosen for several reasons. For starters, Android is open source. This allowed us access to a lot of the features of the phone, such as the sensors, the native applications, and the audio profile manager. As our application requires access to all of these features, this was the preferred platform to develop it on. In addition, as of May 2013 there were over 900 million Android devices activated [4]. The popularity of Android devices provides a large community for feedback on the application.

Application System Flow

Our application evaluates contextual information upon a call coming in. Prior to a call, no decisions are made about the interruption profile (vibrate, ring or silence).

The System Flow of the application can be seen in Figure 2, with the orange boxes representing the contexts. The term calendar is used to represent the schedule context.

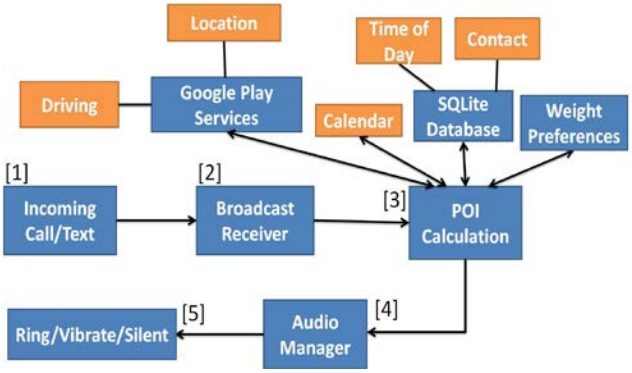

Figure 2: Application System Flow

The application can be divided into 5 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, which changes the phone state. When the phone state of the Android devices changes, a message known as an intent is broadcast. This intent can be received by what is called a Broadcast Receiver, assuming the receiver is looking for that particular intent.

2. Broadcast Receiver

The Broadcast Receiver looks for the intent that is launched when the phone state is changed. Examples of phone states include being idle, off the hook, or ringing. To determine when a call or text is coming in, the application waits to hear an intent that advertises the phone is ringing. This indicates a call or text is coming in and launches the POI Calculation.

3. Probability of Interruption (POI) Calculation

The POI calculation utilizes the Weighted Sum with Adjusted Weights model we produced (see Section 4.4). It is a service that runs in the background and is launched by the broadcast receiver. In order to evaluate this model, it 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). To obtain these values this component queries databases, as well as uses saved user preferences. By these means the POI Calculation service has all the required values. It then runs the model and determines the audio mode the phone should be in. The changing of the audio mode is handled by the audio manager.

4. Audio Manager

The Audio Manager provides access to changing the mode of interruption. After using the POI calculation to determine the mode the phone should be in, the audio manager proceeds to change the phone to that mode.

5. Ring/Vibrate/Silent

At this point the phone is now in the mode the application has determined it should be in. The call or text now is presented to the user in the form deemed best by the model.

Determining the Preferred Method of Interruption (PMI)

If the POI is above 0.5, the application then has a decision to make: ring or vibrate. In some situations it is okay to receive a vibration, but it is not okay for the phone to ring. An assumption we make is that if it is okay for the phone to ring then it is okay for the phone to vibrate. From this logic we devised a way to determine the user's preferred method of interruption, or PMI.

To accommodate for the PMI, we allow the user to specify a PMI for each context. By default it is set to vibrate. If all of the measurable context have a PMI of ring, and if an interruption is deemed as okay, then the phone will ring. However, if even one of the measurable contexts has a PMI of vibrate, then the phone will vibrate instead of ring.

Obtaining and Setting Preferences for the Contexts and Weights

Our model requires complete knowledge of all contexts. This means determining whether or not the contexts are 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.

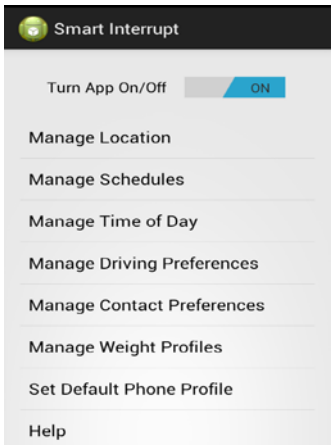

Figure 3: Main Menu

From this main menu the user can then navigate and enter the required information for the six main sections: weight, location, schedule, contact, time of day, and driving. Most of the preferences (e.g. POI and PMI) can be set from drop down menus. In addition, the user can also set the default phone profile. If none of the contexts are measurable, a default profile is used. For the default phone profile the user is able to choose between ring, vibrate, and silent.

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 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 values from the saved preferences.

Location Preferences

The Location UI allows the user to create location profiles. Each of these profiles gets its own set of preferences. For example, the user could create a profile called work. From here they can specify an address of where their work is located. The address that is entered is 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 able to specify a POI and PMI. A SQLite database is used to store the user's preferences, as well 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 that provides 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 that of the user's location profiles. Current technology does not always provide the most accurate location. With this in mind, when the POI Calculation compares the two locations it allows them to be off by a certain margin. This margin is the radius of accuracy, represented in meters, which the Location Services provides when making its location prediction. If the two locations are off by a distance less than the radius of accuracy, then the situation is considered measurable, and the user's preferences are used in the calculation. Otherwise, the situation is considered non-measurable and the default values are used.

Schedule Preferences

For the user's Schedule Preferences this application utilizes the native calendar application. There are several possible options 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, then the context is considered non-measurable and the default probability is assigned. The next outcome is the user has something scheduled, but they do not have a POI or a PMI assigned for that specific event. In this case the user can set a default event POI and PMI. The final outcome is an event that occurs where the user has specified both POI and a PMI in the native calendar application.

Currently, to specify POI and PMI in the native calendar application, the user must place the POI (in increments of 0.1), and the PMI (R for ring, V for vibrate) between dollar sign

delimiters. 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.

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 specify preferences for each of these profiles. Each contact is only allowed to be in one group at a time. The contact's 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, then the default POI is assigned and the context is considered non-measurable. If the contact is found to be in a group, then the POI and PMI are retrieved to be used in the calculation.

Time of Day Preferences

Like both the Location and Contact Preferences, the Time of Day Preferences allows the user to create multiple profiles. Each profile gets a start time, an end time, a POI, a PMI, and the day(s) of the week for which this 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 then 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 system's day of the week. A returned result indicates a measurable Time of Day context, and the necessary values are retrieved. Otherwise, the context is considered non-measurable and the default value is used.

Driving Preference

The Google Play Services update introduced the Activity Recognition API. Through this API it is possible to request the user's current activity. As it takes several seconds to request an activity update and 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 5 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 stop the activity recognition. Stopping the updates from activity recognition means the driving context will always get the default value.

Evaluation

Currently the application's footprint is 2.30 MB, with the source code being 147KB. The application is constantly running, and generally uses 7MB-9MB of RAM. For CPU usage, it does not have an impact 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.

For the location services, the constant connection to the Google Play Services to retrieve location and activity updates places a drain on the battery, as expected. However, if a different application were already using constant location and activities updates, then our application would cause no additional overhead for battery usage.

VI. CONCLUSIONS

In this research, we have implemented an interruption management application that utilizes a probabilistic approach. The system uses an adjustable weighted average of probabilities of interruption for each of the five contexts, namely, schedule, time of day, location, caller relationship, and driving. With this model we then succeeded in designing and implementing the application, such that when a call comes in it uses the model to determine a cell phone's appropriate interruption profile.

The production of this application has provided steps, measurable contexts, a system flow, and an application that future interruption management research can use, modify, or be compared to. In addition, the production of this application has provided insight into its limitations. Most apparent is requiring the user to provide all the values needed for each context. Future research should look to implement applications that can acquire 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 much work still needs to be done on the application, upon completion and publication it will then be possible to receive user feedback. This feedback could prove to be extremely valuable to the topic of interruption management as it will provide insight and effectiveness from its target audience. As no widely available application has been released, this information has not been recorded. Also, the proposed implementation's computation module can be easily replaced with other models, such as the Dempster-Shafer model, and then subsequent comparisons can be carried out.

Finally, with this application it is easy to see the possible future work that can be done in the topic of interruption management. By being able to determine probabilities of interruption for each context without relying on user input, the application would be more reliable and user friendly. Currently, the complexities of the application could prevent 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

expanding beyond Smartphones we increase the ability to prevent unwanted interruptions.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation grant CNS-REU-1063041.

REFERENCES

[1] Bailey, B. P. and Iqbal, S. T.: Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management. ACM Trans. Comput.-Hum. Interact. 14, 4, Jan. 2008, 1-28.

[2] Bureau of Labor Statistics, http://www.bls.gov/news.release/empsit.t19.htm

[3] Burns, Shari M, Lohenry, Kevin., "Cellular phone use in class: implications for teaching and learning a pilot study", in College Student Journal, Sep2010, Vol. 44 Issue 3, pp. 805-810.

[4] Business Insider, http://www.businessinsider.com/900-million-android-devices-in-2013-2013-5

[5] Dempster, A. P., "Upper and Lower Probabilities Induced by a Multivalued Mapping," The Annals of Statistics 28, 1967, pp. 325-339.

[6] Governors Highway Safety Association, http://www.ghsa.org/html/stateinfo/laws/cellphone_laws.html

[7] Grandhi, S.A., Schuler, R.P., & Jones, Q.: To answer or not to answer: that is the question for the cell phone users. Proceedings of the 27th international conference extended abstracts on Human factors in computing systems, 2009, 4621-4626.

[8] Horvitz E., Koch P. and Apacible, J., "BusyBody: creating and fielding personalized models of the cost of interruption," in the ACM Conference on Computer Supported Cooperative Work, 2004, pp. 507-510.

[9] Iqbal, S. T. and Bailey, B. P.: Leveraging characteristics of task structure to predict the cost of interruption. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Montréal, Québec, Canada, April 22 - 27, 2006, 741-750.

[10] International Telecommunication Union, https://www.itu.int/net/pressoffice/press_releases/2013/05.aspx

[11] Mark, G., Gudith, D., and Klocke, U.: The cost of interrupted work: more speed and stress. In Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems, Florence, Italy, April 05 - 10, 2008, 107-110.

[12] National Highway Traffic Safety Administration, http://www.nhtsa.gov/Driving+Safety/Distracted+Driving/Breakthrough+Research+on+Real-World+Driver+Behavior+Released

[13] Shafer, G., "A Mathematical Theory of Evidence. Princeton," NJ, Princeton University Press, 1976.

[14] Spira J.B. and Feintuch J.B., "The Cost of Not Paying Attention: How Interruptions Impact Knowledge Worker Productivity", Basex, 2005

[15] Turney P., "Robust Classification with Context-Sensitive Features," in the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 1993.

[16] William, V., Madiraju, P., and Ahamed, S.I. unpublished report available from http://www.mscs.mu.edu/~praveen/Research/im]

[17] Zulkernain S., Stamm K., Madiraju P. and Ahamed S.I., "A Mobile Intelligent Interruption Management System", in Journal of Universal Computer Science, Vol. 16, No. 15, 2010, pp. 2060-2080.

[18] Sina Zulkernain, Praveen Madiraju and Sheikh Iqbal Ahamed. "A Context-aware Cost of Interruption Model for Mobile Devices", Proceedings of 8th IEEE Workshop on Context Modeling and Reasoning (CoMoRea 2011) in conjunction with the 9th IEEE International Conference on Pervasive Computing and Communication (PerCom'11), Seattle, USA, March 21-25, 2011, pp.421-425