Journal of Universal Computer Science, vol. 16, no. 15 (2010), 2060-2080
submitted: 31/1/10, accepted: 28/7/10, appeared: 1/8/10 © J.UCS

A Mobile Intelligent Interruption Management System

Sina Zulkernain, Praveen Madiraju[1], Sheikh Iqbal Ahamed, Karl Stamm
(Marquette University, Milwaukee, Wisconsin, USA
{sina.zulkernain, praveen.madiraju, sheikh.ahamed, karl.stamm}@marquette.edu)

Abstract: Mobile phones have become the most hated device that people cannot live without. For its primary usage as a communication device, it has surpassed any other medium. But it comes with a high price, interruption, anywhere anytime. These unwanted interruptions cause loss of productivity and also mostly not beneficial to the immediate task at hand, and moving them few minutes into the future can increase productivity. Considering receiver's unavailability, it is possible to manage cell phone disruptions using advanced features like sensing capability, ubiquitous computing and context aware systems. This paper proposes the architecture of a system named Mobile Intelligent Interruptions Management (MIIM), created for the automated administration of personal unavailability with regard to cell phones. We provide the problem description of interruption and its impact. Next, we state the desirable characteristics and architecture of the MIIM system. We also provide a case study implementation of MIIM system on the Android platform. Simulation and evaluation results show that its computational volumes are low enough for a mobile device. The analysis of the system also successfully satisfies all the characteristics requirements.

Keywords: Context Aware System, Interruption, Ubiquitous Computing, Unavailability
Categories: H.5.m

1 Introduction

A cell phone is an electronic device used for mobile telecommunication i.e. telephony, text messaging and data transfer over a cellular network. Mobile phones have gained importance in the sector of information and communication technologies only in the 2000s. However, according to an estimate by the International Telecommunication Union, mobile cellular subscriptions worldwide would reach approximately 4.6 billion by the end of 2009 [International Telecommunication Union]. Never ever before in history did a device reach so many people in such a quick time! There also has been a tremendous growth in smart phone applications. Recent statistics indicate there are over 100,000 active iPhone applications [148 Apps.biz]. Cell phone has become a necessity in our daily life. A University of Michigan study [University of Michigan News Service] shows that 83% people think cell phones make life easier and they choose it over the Internet. Mobile phones have the obvious benefit of all the moment communication. Irrespective of time and place, we can expect a phone to ring. But a ringing phone interrupting at an inopportune moment can be very disruptive to the current task or social situation [Guzman, 07]. In a survey of 1000 senior executives, it was reported that undesirable interruptions constitute 28 percent of the knowledge worker's day, which translates to 28 billion wasted hours to companies in the United States alone [Spira, 05]. It results in a loss of

[1] Corresponding author

700 billion dollars per year, considering an average labor rate of $25 per hour for information workers [Bureau of Labor Statistics]. A University of Oxford experiment suggests that in cognitively demanding situations, the advantage that 18-21 year olds enjoy over 35-39 year olds is reduced by an interruption caused by electronic communication technology [Disruptive communication and attentive productivity]. Interruptions are mostly not beneficial to the immediate task and moving them a few minutes into the future could greatly benefit many users [Adamczyk, 05]. Undesired disruption causes interrupted users to take up to 30% longer to complete and commit up to twice the number of errors [Bailey, 06]. Hence we need a mobile interruption management system that will decide in real-time if the user should be interrupted or not.

1.1 Similar researches

The system we propose uses the capabilities from ubiquitous computing and context aware systems to programmatically learn about the environment and achieve our goals. Modelling context information and software engineering framework for context aware pervasive computing is already built by [Henricksen, 04]. We also have location and environment aware handheld systems [Jiang, 04] and frameworks in development for generalizing the sensor interfaces [Dey, 02]. We have distributed resource discovery [Sharmin, 06] and trust models for anonymous sensors [Ahamed, 08]. Altogether, the avenue is clear for revolutionary system development awaiting only the sensor deployments.

Dekel [Dekel, 09] built an application named Smart Profile Management that minimizes mobile phone interruptions by changing profile settings intelligently. It does so by taking into account time of day and phone's database. Savioja [Savioja, 04] addressed different kinds of alarms for different types of interruptions in control room environments. Khalil [Khalil, 05] used calendar information of the phone to minimize disruptions. Marti [Marti, 05] devised an application for a group setting where a phone had to get all of the members' votes before ring. However, a distinguishing aspect of our work is that we have identified desirable characteristics of an interruption management system, and proposed a system architecture that satisfies these characteristics. A similar research like ours is done in [Godbole, 06], which proposes a methodology and design process for the development of interruption aware systems in general. But it neither provides the solution for any specific context nor any performance evaluation of the system.

1.2 Contribution of this paper

In this paper, we propose an intelligent interruption management system for mobile phones. The system is intelligent because of its adaptability, awareness for both context and unavailability, and also automatic decision making capability. Currently the prototype is user installable on the Google Android platform. It installs a service which monitors the ubiquitous pervasive computing environment for new sensory data and builds automatic decision algorithms inspired by machine learning systems to decide in real-time, if an incoming call is going to be a costly interruption and therefore prevented or a valuable call and hence allowed through to the user. The user

is a first-class entity who has the power in our system to define all of the rules and behaviors, such as who to let through and what situations constitute private moments.

In our earlier poster paper [Stamm, 10], we proposed a preliminary system architecture for an interruption management system with initial results. In this paper, we are extending on our previous work and provide the following contributions.

- Identified unique features/characteristics of an interruption management system.
- Designed and developed system architecture for the system.
- Carefully surveyed the state of the art.
- Implemented a case study application using the developed system architecture.
- Evaluated the prototype application.
- Simulated the system for scalability.
- Gathered user's feedback regarding the usability of the case study application.

The rest of the paper is organized as follows. Section 2 provides the motivation and desirable characteristics of the system. Section 3 focuses on the related works. System architecture is described in Section 4, followed by the evaluation of the system in Section 5. Finally, Section 6 concludes our findings and paves the way for future works.

2 Motivation

The problem we seek to solve has had a real cost on the aggregate. We want to stop phones from interrupting when someone is not actually ready to receive a call, because it is those instances that incur social cost with zero gain. In order to effectively understand desirable characteristics of the research solution, we first analyze relevant scenarios where an interruption management system will be useful. To illustrate the motivation for an intelligent unavailability system, we show a system environment, both before the integration of our system and then after, and consider three circumstances related to it. This scenario considers the primary user with her phone turned on and set to loud ring. Our hypothetical user is Alice and she is presenting at a meeting and her cell phone is inside her purse. Her friend Bob calls her without having any idea of her current state.

2.1 Scenario 1: Featureless Phone

Alice is using a featureless phone. So the cell phone rings loudly and she has to pick it up from her purse and turn it off. But already the interruption has been instantaneous and complete. Alice is derailed from her train of thoughts and everyone else in the meeting is annoyed and an aroma of social stigma is clearly in the air. Now, let us enable the ubiquitous context supplier and see the differences. This phone can only receive data from its vendor supplied cell phone towers and knows nothing about the local environment. So no real improvement can be achieved here.

2.2 Scenario 2: Smartphone with interruption management system

Alice is using a smartphone with the proposed interruption management system installed. Her phone knows from her calendar schedule that she is supposed to be present at a meeting during those hours. So the cell decides to go to the silent mode or

takes the caller to the voice message option. Later, when the meeting is over, Alice has no idea that there had been a call. Now, this could have been an important one or an emergency call. If Alice does not check her phone for any call or make a call herself within a certain time period, she may miss something very important. Here, although the cost of interruption is mitigated, a potentially aiding call is missed.

2.3 Scenario 3: Smartphone with context aware interruption management system

Alice is using a smartphone with the proposed interruption management system installed in a context aware environment. At the time of call, the phone behaves same as it did in Scenario 2. But when the meeting is over, the phone can sense that it is out of the office area, and then interrupts Alice that there had been a call. Now, she not only avoids any interruption during the meeting but is also notified in a timely manner about the all important call.

Here we have seen the value of an unavailability manager in the generic scene. While the ubiquitous computing system is an importance piece of the puzzle, much can be done only with the handheld on a smart platform. Even though context aware sensor networks may not be deployed in every situation, our system software installed in the cell phone can provide most of the services. We derive the following characteristics for the proposed system from the above mentioned scenarios.

2.4 Characteristics

We have a set of constraints and goals that could be considered system requirements, feature requirements or program goals. We define six necessary characteristics labelled C1 through C6.

Mobility (C1): The system must be installable on a small handheld device like a cell phone. Furthermore, the system must be mindful of data transference costs, memory limitations, disk space limitations and CPU limitations. It has to cater to a mobile environment.

Customizable (C2): The user will not accept a system wherein he/she cannot change the rules and outcomes of the system in many ways.

Adaptable (C3): The system must change itself to different environments. Mobile devices have differing feature sets from CPU power, to screen size, to input methods. Secondly, the environments differ in scope.

Context Aware (C4): The system will need to know about its own environment to know about the user's environment, to know what situation it is in and which social rules it must be following.

Automated (C5): We need the device to make the decision of interruption in real-time, without user interaction.

Unavailability Aware (C6): The system needs to take into account different modes of unavailability: vision, hearing and touch. It also needs to respond to user specific interruption modes: ring, vibrate or silence.

3 Related work

We are building a system for managing interruption using context aware devices. When a call is made to a phone, the system decides beforehand to not the let the call go through if it is costly interruption, The cost of interruption (COI) is a function of immediate task and the user's state of mind, which can also be seen as a function of the task at hand. A proper ubiquitous computing system can theoretically understand the task at hand and infer the user's state of mind and therein get a measure of COI. Hence the survey of literature spans into areas related to COI, interruptions and context aware systems.

3.1 Cost of Interruption (COI)

Adamczyk [Adamczyk, 04] measures the effects of interruption in terms of task performance, emotional state and social attribution. The study suggests that the relationship between interruption and task is crucial in certain conditions. It also aims to find the most suitable time to interrupt the user. Several researchers have addressed the issue of cost of interruption [(Bailey, 08), (Iqbal, 06), (Grandhi, 09)]. Mark [Mark, 08] measures the COI based on additional time required to reorient back to the primary task and mental stress brought upon the interruptee. The authors suggest that COI differs in various individuals. The results show that openness to interruptions and quickness in handling them can lower the cost of disruption. Bailey [Bailey, 08] approaches interruption by measuring the value of delivering information against the cost of interrupting the primary task. To compute COI, the authors use non task specific cues e.g. desktop activity, visual and acoustical analysis of the environment and user's scheduled activity. To understand work load changes during task execution, the authors used pupil size. A user's pupil size increases due to the mental processing efforts and there is an upper bound on how much it can grow. Bailey [Bailey, 08] shows this could be a possible way to measure a user's mental stress and hence decide whether interruption could be detrimental or a bit refreshing anyway. The bottom line is to defer interruption when COI is high. This has been shown to not only increase worker efficiency, but also benefit morale [Adamczyk, 05]. Our approach in dealing with COI is to design context enabled rules which form a tree based data structure (see Section 4, Figure 3). The system evaluates these rules and decides if the user needs to be interrupted.

3.2 Interruption Management

To manage interruption, first we need to specify the factors that make interruption a burden. Horvitz [Horvitz, 04] describes a system that builds decision-theoretic models by asking users about their perceived interruptibility during a training phase. Ho [Ho, 05] considers 11 factors that impact the perceived burden of interruption namely activity, emotional state and social engagement of the user, social expectation in a group environment, user's control over the device and task efficiency rate, importance of the message to the user, medium and frequency of interruption and, user's previous and future activities and also history and likelihood of responses. The authors further suggest that an exhaustive model of interruptibility should include a weighted sum of these factors.

Next we need to use context aware services to manage interruption. Abundant body of literature has studied the issue of context management for personal computers [(Dey, 01), (Hull, 97), (Brown, 96)]. Baladauf et al. [Baldauf, 07] presents a survey of context aware systems. The typical contexts included are: location, time, day, and proximity. In relation to interruption management, several researchers have proposed other meaningful contexts. Petersen [Petersen, 08] mentions the challenges to face when pervasive computing becomes a reality and a part of our everyday life. It discusses the problem of interruption in pervasive aware systems and identifies different roles that software agents can play supporting it. Godbole [Godbole, 06] considers three types of contexts to solve the interruption problem. The authors consider interruptor-interruptee relationship, interruptor's context and interruption content such as relational context, interruptee's local environment factors e.g. place, people around as social context and, interruptee's cognitive level of involvement in tasks and interruption's effects on task performance as cognitive context.

3.3 Context Aware Systems

A context aware system is a computing resource with knowledge of its environment and its user's situation. Research in autonomic computing [Ziebart, 05] recognizes the complexity involved with applying or interfacing such a system with human users. The problem arises when we expect the context aware system and the pervasive environment to combine into one intelligent environment. Ziebart's [Ziebart, 05] work on Learning Automation Policies serves to solve this problem. Our work also falls in this area, where the personal mobile device is the decision making computer. It too needs to support multiple inputs and outputs from as yet unforeseen sensors. But the restricted needs of a mobile device discretion system do not merit the use of machine learning, nor are the computing resources required for machine learning available to the mobile device. We cannot simply apply heavy expensive algorithms and hope them to run on a battery and otherwise constrained device. So we have provided here the solution to the many inputs problem as a semantic tree heuristic. In a context aware system, information will come in from many sources rather than only one or two streams of input. The Context Toolkit is a java based library that facilitates development and deployment of context-aware systems [Dey, 00].

With context aware systems assumed and available, the next step is aware and adaptive services. In [Conlan, 03], context awareness is extended into the service oriented architecture. After configuration, a service oriented system is accessible to any sort of request and it also provides extensibility and scalability in the enterprise setting. Our unavailability system will require such rich information sources to properly diagnose a given situation. Privacy is a topic that is closely related to personal unavailability. In [Lederer, 03], interpersonal relationships in regard to data privacy preferences have been addressed. A system called Lilsys, which reads motion, sound and door-closed-state has been constructed in [Begole, 04] to build a qualitative measure of user unavailability. Also, automated preference control on mobile devices has been tackled in [Bayley, 08].

3.4 Interruption associated with Mobile Devices

There have been several works on how to manage interruption at inopportune moments using smartphones. Yu [Yu, 06] defines user preference, terminal capability, location, time, activity and so on as context dimensions for smartphones. In [Picard, 07], the authors suggest that an interruption technology adapting its response considering person's feelings is likely to improve people's experience with that technology. Godbole [Godbole, 06] surveys the type and extent of desired information about the incoming cell phone calling. Their findings show that the desired information is highly unknown and often misattributed by the user. Guzman [Guzman, 07] studied the context information users consider when they make a call and also the context information they wish others consider when they receive a call. The study shows that the key points are: Location, Time, Physical Ability, Social Availability, Task Status and Emotional Availability. Also the authors show that some users do not even consider any of these issues whereas some others consider something else. In [Toninelli, 08], the authors group the strategies for interruption management by filtering calls based on caller's identity, situation and time, and, status message sharing e.g. current location, activity etc. As users tackle interruption by taking some actions themselves, Toninelli [Toninelli, 08] suggests that the intelligent system should learn how the users act in some situations, learn from them and later take actions like them.

Characteristics → **Research Works ↓**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**
Toninelli et al. [Toninelli, 08]	X	X	-	X	X	-
Godbole [Godbole, 06]	-	-	-	X	X	-
Picard et al. [Picard, 07]	-	-	X	-	X	-
Bailey et al. [Bailey, 08]	-	-	X	X	X	-
Ho et al. [Ho, 05]	-	-	X	X	X	X
Mark et al. [Mark, 08]	-	-	-	X	X	-
Dekel et al. [Dekel, 09]	X	X	-	X	X	-
Guzman et al. [Guzman, 07]	X	-	-	X	X	-
Khalil et al. [Khalil, 05]	X	X	-	-	X	X
Marti et al. [Marti, 05]	X	X	-	-	-	X
Our System	X	X	X	X	X	X

Table 1: Comparison of different interruption management systems

The aforementioned research works give us a solid basis for (i) which context needs to be considered, and (ii) how to evaluate such context. However, the chief distinguishing aspects of our work are (i) system architecture and prototype implementation with performance evaluations, and (ii) identification of desirable characteristics of the problem solution. Although not always explicitly stated, we analyzed the above mentioned interruption management systems and identified the characteristics (Section 2.4) we defined before (see Table 1).

4 General System Architecture

Our objective is to design a generic system architecture that will have components which can be adapted for either a mobile or personal computer interruption management system. In this section, we present our proposed general system architecture for mobile interruption management system. The general components of the system are shown below in Figure 1.

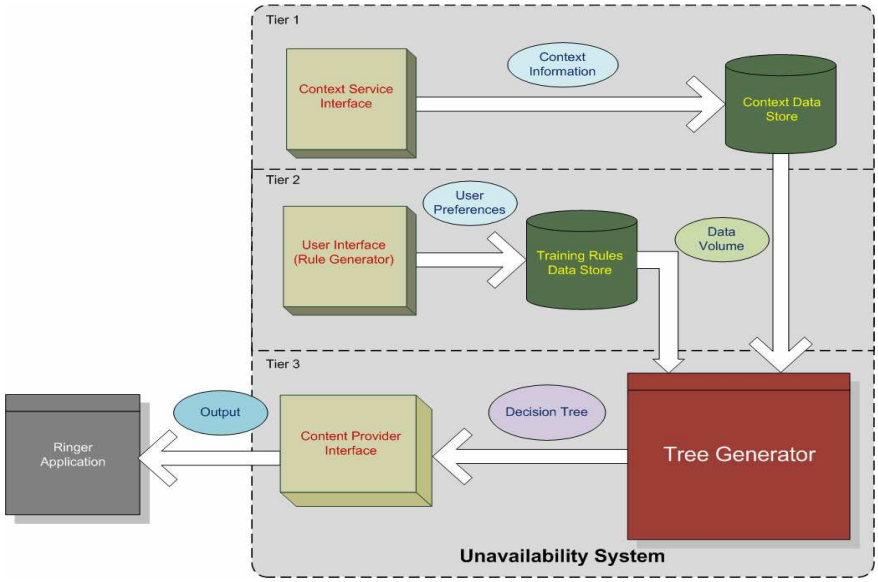


Figure 1: General System Architecture

The large unit on the right, labelled Unavailability System is the system installed on the handheld. The system is divided into three tiers. The first tier includes the Context Service Interface in the upper left and Context Data Store in the upper right of the main unit. Upon reception of data, this tier collects and stores them in a continual process. In the middle, we have the User Interface Component that saves user preferred configurations in the Training Rules Data Store. The main component of the third tier is the Tree Generator. The tree generator collects the available context information from the first tier and user preferences from the second tier. This is the

processing stage whereupon the decision structures are made. Tree Generator provides its decisions to the Content Provider Interface. The Content Provider Interface then instructs the Ringer Application on the phone to either ring or not ring. Finally, Ringer Application on the left is external to our system, but internal to the handheld's operating system.

Tier 1: Context Information

Context Service Interface in Figure 1 is a persistent service on the mobile device that actively searches for context information from publicly available sensors. It aggregates information from the internal sources and generates a Context object to be stored in the Context Data Store. This object is passed to the Tree Generator (Decision Tree tier) for processing.

Each sensor has different data. Hence, the context object will change for each of the different sensors. If there are no sensors for a data category, then the context object is missing this information. The sensor data changes from time to time as the platform is mobile and will move into and out of range of sensor's viability.

The Context Data point can be internally represented as a tuple of {data type, value, semantic, expiration}. It has a recognizable data type defined in itself, because incoming context could be string, or integer, or floating point. It comes with a string representation of the semantic set that the data represents, to better fit itself into the hierarchy of context. It also comes with an expiration date, to keep the data store from growing stale, because all data from sensors age and fall away. If the sensors do not supply an expiration time, the context reader service generates a value for this field.

Tier 2: User Preferences

Each user has a set of preferences about when to let calls through or when to deny them in any circumstances. These rules appear contradictory between users. So we need some sort of training data that needs to be collected for example using a survey. These might be served by a device with a larger interface, as they are not going to change rapidly and do not need to be generated on the mobile phone. The User Interface Component in Figure 1 is the primary driver for generating these preferences. A sample user interface is shown in Figure 2.

We can generate some information in advance; if there can be found a set of training data that is true in all circumstances across all users. The user will want to input rules that come to mind as priority rules. Presumably, each user who installs the system will have a set of situations s/he wants taken care of by the unavailability sensor. We must give a method of inputting specific rules. We can supply an interface right on the handheld with dropdowns for each sensor showing possible values, and the resulting classification. This lets the user spell out each sensor set. A survey has the benefit of covering other situations that the user does not immediately think of. Although there are myriad permutations of sensor data to consider, the training data set can be minimized through generalization of these data.

The two methods (manual and survey) can be combined in a randomly generated manual entry method. This seems to be the best choice. When the user opens the

manual entry activity, the default values populated on the control will be generated randomly in effort to spur the user's creativity and let them classify random instances. They can change the values to match the preferences they intended to insert, or continue filling in random sets. A few minutes of random set generation should manage to cover a very wide range of training data successfully, and in an entertaining manner. A sample user interface for the unavailability system is shown below in Figure 2.

Tier 3: Decision Tree

The Tree Generator receives inputs of Context Data Point from Tier 1 and user preferences from Tier 2 and then generates a decision tree structure. A decision tree is a structure of conditional code that classifies a data set into one or more categories. A decision making piece of code is one that takes in sensory data and then activates the correspondingly correct action.
A decision tree is generally regarded as the cheapest in terms of computational complexity at runtime. It is also well suited to a discrete set of a small number of outcomes. The structure is a graph in tree formation, each branching point is representative of a conditional decision, and each leaf node is a final answer.

Figure 2: Sample User Interface

The tree need not be flat on the ends, i.e. not all paths must have the same length. We could quickly have a decision if the circumstances are correct that skips most of the checks. To make a decision, the tree is traversed. At the root node, all of the data is held that will make a decision. The first branching will split the classification of final outcomes as evenly as possible, so as to have an efficient tree. Learning algorithms in the data mining literature exist to construct the tree to match the data available, and ensure it is efficient and compact. Some popular examples are C4.5 [Quinlan, 93], SLIQ[Mehta, 96], and Rainforest[Gehrke, 98]. The premise is that if a decision can be

made quickly, it will be. The structure of the tree will be generated when the data set structure is known. Outside the ubiquitous computing environment, the only data available for the decision maker is that known inside the phone. For our hypothetical tree construction (see Figure 3), we have a set of available data and a set of outcomes. The biggest decision is the one that divides the most instances. In our case, the tree is constructed using a top down recursive divide and conquer method (greedy method).

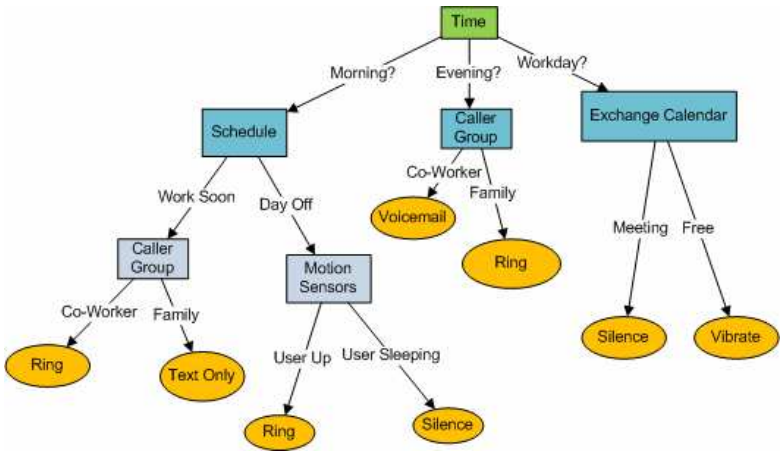


Figure 3: A Decision Tree

We need an entire training data set to calculate these things. The training dataset can be generated by permutation of possible results from each input vector.

5 Case Study Implementation

Here we provide a case study implementation where a company (anonymous for privacy reasons) wants to send all its sales employees performance metrics each hour. These metrics represent their production, sales and the employee's current rank compared to others. They use it to make their sales force competitive within the company. Each sales person has an area specified as their work area and the company wants to send the metrics to a sales person whenever they are in the work area. Some sales people work indoors and they are mostly in the positions of managers. Now even though a sales person may be in his/her work area, s/he might be busy in a meeting with their superiors. Whenever the sales person is in some scheduled event the company does not want to send the metrics to him/her. Based on these requirements, we developed our prototype application. Whenever the sales person is in their designated area, not busy in a scheduled event and also when the time is between their work hours, we show the metrics on the device. But when any of the cases fail, we do not show him/her the metrics.

For now, the prototype is developed on the Android, the operating system of a new class of smartphones which was designed primarily at Google in participation with the Open Handset Alliance. The reason for choosing Android is that it is Linux to the core and entirely open sourced. Most importantly when there is no call (in this case when no data is sent from the server), our application can run as a background process using minimal CPU or battery resources. The application needs to be installed in the receiver phone and Android is the only platform that allows full control of the ringer actions i.e. the interrupter. In the future, we also plan to implement it on other platforms as well. For the prototype, we used three contexts: location, schedule, and day of week along with time of day. We used Google Calendar as our scheduler. So our assumption is whoever uses our system will have some sort of scheduler where the application can query into. To find the location, we used the GPS service provided by Google. So it had to be a smartphone. And we used the system clock to find the day of week and time of day. .Now we show step by step screen shots to explain how our application is working.

Figure T1

Figure T2

Figure T3

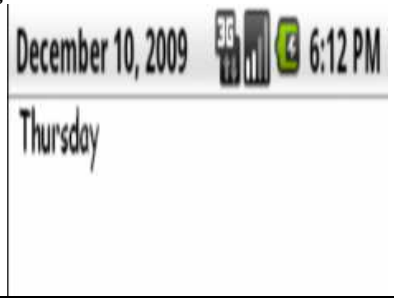


Figure T4

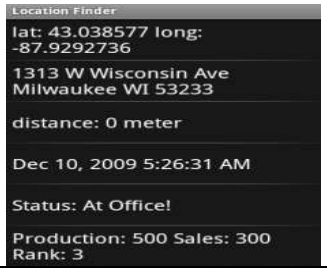
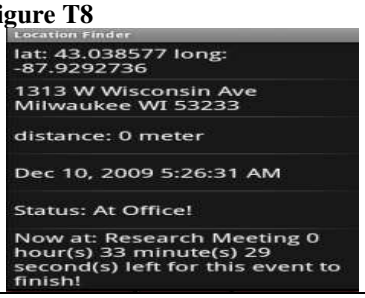
Figure T5	**Figure T6** 
Figure T7	**Figure T8** 

Table 2: Screenshots of the application

Figure T1: The application first looks for its office location. It knows from its data storage what user's office address is and pinpoints that location using Reverse GPS service. Figure T1 in Table 2 shows user's office address.

Figure T2: The application then looks for user's current location. It uses GPS service and shows a region of radius 1000 meters where user could be. Figure T2 in the above table shows that region by a black circle. Measuring the distance from the center of this circle to user's office location, the application decides that user is at office.

Figure T3: Now the application uses the system clock to get today's day of week and current time. From its data storage the application determines that it is user's working hour.

Figure T4: Figure T4 in Table 2 shows user's scheduler, in this case the Google Calendar. The application now queries the calendar to get user's current schedule.

Figure T5: The application sees that user has no event specified at the current time (Figure T5 in Table 2). So it decides to show the salesperson metrics sent from the office.

Figure T6: This figure shows the things the application considered before showing the metrics. The first line shows user's current location in latitude and longitude and user's current address in next. Then it shows the distance between user's current location and his/her office location. Next line shows current day and time. The application then shows user's status. The last line in the figure shows the metrics.

Figure T7: Now we make a change in the calendar and put an event there. Now the user is supposed to be busy. So the application sees that user is busy and now takes a decision not to show the metrics to the user.
Figure T8: This figure shows in the last line the event the user is currently attending. Also it shows the time left for this event to finish.

6 Evaluation

Here we provide a detailed evaluation of our proposed system. First we analyze the system by simulations, prototype evaluation and cognitive walkthroughs of the application.

6.1 Simulation

The first component of the system, gathering and organizing data, is a process that depends computationally on the sources of information and the data volume and the data structure. Therefore, the major need is a simulation of data volumes and data structures in typical use. For feasibility simulation, we choose the count of the context suppliers to be between one and ten sources, with ten highly unlikely and one or two quite likely. This assumption stems from the belief that commonly travelled areas are more likely to have fewer sensors than more. We estimate this number with an exponentially random variable with parameter of 0.350. This gives a distribution such that 50% of cases have between 1 and 3 sensors, 25% between 3 and 5, 10% between 5 and 8, and so on with ever decreasing probability. The simulated distribution is shown below in Figure 4.

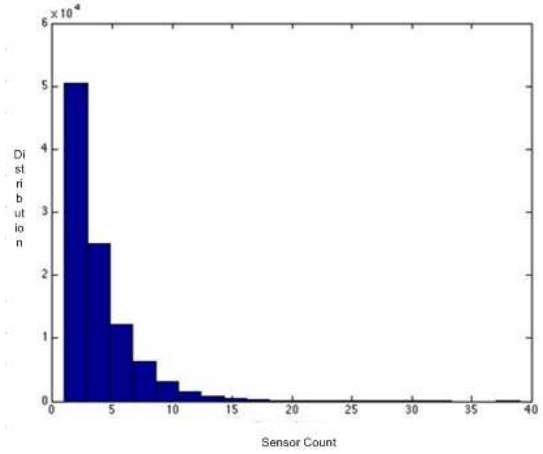


Figure 4: Histogram of 100,000 total instances

Similarly, the number of data points included from a context source is going to be an exponentially distributed random variable. Generating with the parameter 0.100 gives a distribution such that 50% are less than five data points, 20% are between five and

ten, 14% between 10 and 15, all the way to 1% giving fifty data points. This seems like a reasonable distribution of data from the ubiquitous sensor network's data supplier.

This leads us to the size of each data point. We are interested in computation to parse in the context data, so only the length of the data is relevant. Measuring in bytes, we are interested in a distribution such that the higher percent of results are below 15 bytes, the next portion below 30 bytes, and the next below 60 bytes. This is best generated as an exponential random variable with parameter 0.050 and an offset of five bytes overhead.

We combine these results to generate a distribution of data volume. Each context source is simulated to bring with it an overhead of fifty bytes xml header. In truth, this is an insignificant distinction. Multiplying the results of exponential random variables yields a new exponential random variable. The new one should be preserved as an exponential random variable with parameter $(0.350 \times 0.100 \times 0.050) = 7/400$. This is a distribution with 50% of the instances below one kilobyte of data, and 90% of the instances below five kilobytes of data. These values are well within the operational parameters of the mobile device's available memory.

The next step is selecting unique pieces of data and consolidating the received information. If each piece of data is matched against all previous pieces, then the amount of calculation required is an order n squared operation. We could sort the items and check line by line for duplication, this uses somewhere between linear time and another n squared operation. So n2 + n are the number of checks for the filtration step. This gives a conservative estimate for the number of operations the mobile device will need to do.

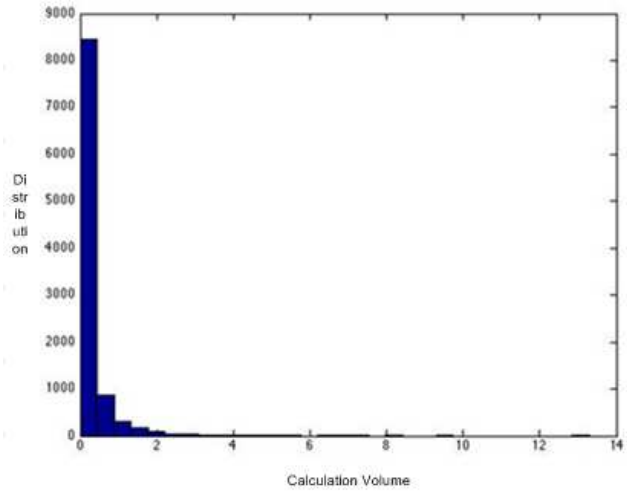


Figure 5: Histogram of filter step

From our construction so far, the distribution of the number of data points is as follows. 50% of instances have below 20 data points, 25% are 21-40, and 10% are 41-60 with a long tail of the remaining 15% through 200 data points. The filtration step

therefore gives a distribution that would have at maximum 2002+200=40200 checks, and the majority requiring below 202+20=420 checks. Most of these instances are calculated in the blink of an eye on modern hardware.

In Figure 5 we show the distribution of data volumes that can be expected at any polling time. The numbers were first transformed with the n2+n function to show the number of operations required to filter the data. Here the vertical axis is probability and the horizontal is the processing requirement in thousands of comparisons. We have shown that the calculations involved to control the context information are on an order of magnitude small enough for mobile devices to handle.

6.2 Prototype Evaluation

In our MIIM system, data volume received is calculated below 50 kilobytes at any given environmental change. We have over 64MB to use and the memory usage requires a fraction of that. With respect to data transference, this is a perfect conformity to characteristic C1 (mobility). Mobile Intelligent Interruption Management (MIIM) system is installed on a cell phone with dimensions only 117.7 mm × 55.7 mm ×17.1 mm and in respect to size and portability, it conforms to C1 (mobility). The three main components of the system are data gathering and organizing, preference configurations through user interface and decision tree traversal. To acquire information, our system collects context from the pervasive middleware and sensors that are network aware. It also uses sensors built into the device to gather data as to the local context. So it is C4 (context aware). Also, the unavailability system accepts varying sensor sets in differing situations and environments. The application cannot require a certain kind of sensor data, because many situations will have little or no sensors of any particular datatype. A few sensors can be assumed statically existing, namely those built into the device. Our prototype platform has built in GPS for instance, and this is one of the only data sources that can be assumed to exist for all deployments and use cases. Thus the system is C3 (adaptable). Decision tree traversal is a linear process. So the CPU power usage is very low along with the battery concerns. MIIM system keeps a ready to use decision tree so that when a call comes in, it can immediately make a decision and prevent the interruption without interference by the user. This prevents interruption and satisfies C5 (automated). With each decision tree having a different structure due to the per user customization, we have easily satisfied C2 (customizable). The user interface component is a constant time computation, as its structure needs not be generated on the fly beyond device restrictions; again less CPU and memory usage. This system also utilizes the varying modes of unavailability; vision, hearing and touch. These levels of permissions for incoming communication attempts make the system smarter and more user friendly, providing appropriate attention to the different sorts of unavailability modes and thus satisfies C6 (unavailability aware). Thus, all the characteristics outlined in Section 2.4 have been realized in the solution we have proposed.

6.3 Cognitive Walkthrough

To get the proper assessment of our application, we used the cognitive walkthrough strategy. We did a survey on a group of 30 people on the usability and usefulness of

our application. First we explained the problem, briefly went over some of the issues we addressed and then showed the prototype application demo. The distribution of the participants is as follows : 17 undergraduate students, 8 graduate students, 2 faculty members, 2 entrepreneurs, and 1 other.

We handed 5 questions about the application over to each participant and requested them to answer them on a scale of 1 to 5. The questionnaire for the survey is given below:

1. Overall, how would you rate the services? (*1 = Very Poor, 5 = Excellent*)
2. What is the effectiveness of this application? (*1 = Not Effective at all, 5 = Very Useful*)
3. How easy is it to give the input? (*1 = Very Hard, 5 = Very Easy*)
4. Will you pay to use this application? (*1 = Definitely Not, 5 = Definitely Yes*)
5. Would you recommend this application to a friend? (*1 = Surely Not, 5 = Surely Yes*)

The survey results are shown in a chart below:

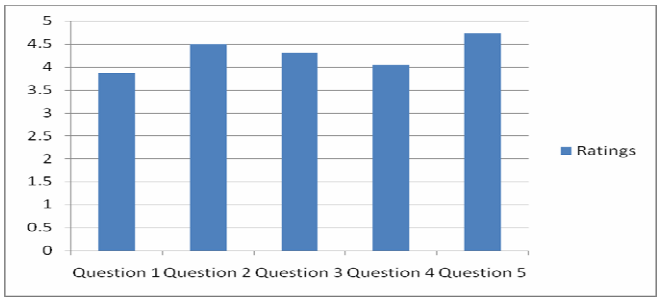


Figure 6: Survey results

From the graph, it is evident that participants were enthusiastic about the application and its usability.

7 Conclusion

In this paper, we have presented the design, development and evaluation of a Mobile Intelligent Interruption Management (MIIM) system. The system architecture considers context information and user preferences and automatically filters out interruptions for mobile devices. We also presented a prototype case study that implements the system architecture. The system is fully analysed and the performance evaluations indicate that it is efficient to run within the constraints of a handheld device.

We plan to extend our work with additional features. The caller can be notified of the receiver's current state if s/he is not picking up. The receiver may not want to

disclose this information to everyone. In some cases, s/he might just want the caller to know that s/he is "busy", wherein the other cases such as to a spouse s/he would like to inform the caller specifically of his/her current state. Again, this information can be passed to the caller in a simple text message or there can be a user interface for the caller in our application where this information is viewed. Secondly, receiver can inform the caller when to try calling again. Acquiring information from the user's task scheduler, our system can know when the current task is going to finish and notify the caller accordingly. In some cases, the receiver may just fail to notice that there is a call. In that case, the system can encourage the caller to try again instantaneously.

We also plan to formalise the model for unavailability which takes into account context-aware services such as location based services. As a part of our goal, we are currently working to mathematically formalise Cost of Interruption (COI). We will also plan to expand the usability evaluations of the system by having more questions regarding font, color, contrast, and screen alignment. We also like to explore possible applications of our system in different application domains from cell phones to instant messaging, email clients, and social networking. These are some areas which operate by interrupting a user and we plan to incorporate our unavailability feature to them so that the cost of interruption is kept to a minimum.

References

[Adamczyk, 04] Adamczyk, P. D., Bailey B. P.: If not now, when? the effects of interruption at different moments within task execution. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Vienna, Austria, April 24 - 29, 2004, 271-278.

[Adamczyk, 05] Adamczyk, P. D., Iqbal, S. T., Bailey, B. P.: A method, system, and tools for intelligent interruption management. In Proceedings of the 4th international Workshop on Task Models and Diagrams, 2005, 123-126.

[Ahamed, 08] Ahamed, S. I., Sharmin, M., Ahmed, S.: A Risk-aware Trust Based Secure Resource Discovery (RTSRD) Model for Pervasive Computing. In Proceedings of the 2008 Sixth Annual IEEE international Conference on Pervasive Computing and Communications, March 17 - 21, 2008, 590-595.

[Bailey, 06] Bailey, B. P., Konstan, J. A.: On the need for attention aware systems: Measuring effects of interruption on task performance, error rate, and affective state, Journal of Computers in Human Behavior, Vol.22, No.4, 709–732.

[Bailey, 08] Bailey, B. P., 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.

[Baldauf, 07] Baldauf, M., Dustdar, S., Rosenberg, F.: A Survey on Context Aware Systems. International Journal of Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, 2007, 263-277.

[Bayley, 08] Bayley, C., Jernigan, C., Lin, J., Shu, J., Wright, C.: Talk Android, February 11, 2008, http://www.talkandroid.com/android-forums/android-market-reviews/495-locale.html.

[Begole, 04] Begole, J., Matsakis, N. E., and Tang, J. C.: Lilsys: Sensing Unavailability. In Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, 2004, 511-514.

[Brown, 96] Brown, P. J.: The Stick-e Document: a framework for creating context-aware applications. In Electronic Publishing, Palo Alto, 1996.

[Bureau of Labor Statistics] http://www.bls.gov/

[Conlan, 03] Conlan, O., Power, R., Higel, S., O'Sullivan, D., Barrett, K.: Next generation context aware adaptive services. In Proceedings of the 1st international Symposium on information and Communication Technologies, September 24 - 26, 2003, 205-212.

[Dekel, 09] Dekel, A., Nacht, D., Kirkpatrick, S.: Minimizing mobile phone disruption via smart profile management. In Proceedings of the 11th international Conference on Human-Computer interaction with Mobile Devices and Services, Bonn, Germany, September 15 - 18, 2009, 1-5.

[Dey, 00] Dey, A. K.: Enabling the use of context in interactive applications. In CHI '00 Extended Abstracts on Human Factors in Computing Systems, the Hague, the Netherlands, April 01 - 06, 2000, 79-80.

[Dey, 01] Dey, A. K., Salber, D., Abowd,G. D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. In the Human-Computer Interaction (HCI) Journal, Volume 16 (2-4), 2001, 97-166.

[Dey, 02] Dey, A., Mankoff, J., Abowd, G., and Carter, S.: Distributed mediation of ambiguous context in aware environments. In Proceedings of the 15th Annual ACM Symposium on User interface Software and Technology, Paris, France, October 27 - 30, 2002, 121-130.

[Disruptive communication and attentive productivity] http://www.iii-p.org/research/disrupt_comm_report_v2.pdf

[Gehrke, 98] Gehrke, R. Ramakrishnan, and V. Ganti: Rainforest: A framework for fast decision tree construction of large datasets. In Proc. 1998 Int. Conf. Very Large Data Bases, pages 416-427, New York, NY, August 1998.

[Godbole, 06] Godbole, A.; Smari, W.W.: A Methodology and Design Process for System Generated User Interruption based on Context, Preferences, and Situation Awareness. Information Reuse and Integration, IEEE International Conference, 2006, 608-616.

[Grandhi, 09] 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.

[Guzman, 07] De Guzman, E. S., Sharmin, M., and Bailey, B. P.: Should I call now? Understanding what context is considered when deciding whether to initiate remote communication via mobile devices. In Proceedings of Graphics interface 2007, Montreal, Canada, May 28 - 30, 2007, 143-150.

[Henricksen, 04] Henricksen, K., Indulska, J.: A software engineering framework for context-aware pervasive computing; Pervasive Computing and Communications (PerCom) 2004, Proceedings of the Second IEEE Annual Conference, 77 – 86.

[Ho, 05] Ho, J., Intille, S. S.: Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Portland, Oregon, USA, April 02 - 07, 2005, 909-918.

[Horvitz, 04] Horvitz, E., Koch, P., Apacible, J.: BusyBody: creating and fielding personalized models of the cost of interruption. In Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, November 06 - 10, 2004, 507-510.

[Hull, 97] Hull, R., Neaves, P., Bedford-Roberts, J.: Towards situated computing. In Proceedings of International Symposium on Wearable Computers, HP Laboratories Technical Report HPL, 1997.

[International Telecommunication Union]
http://www.itu.int/newsroom/press_releases/2009/39.html

[Iqbal, 06] Iqbal, S. T., 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.

[Jiang, 04] Jiang, X., Chen, N. Y., Hong, J. I., Wang, K., Takayama, L., Landay, J. A.: Siren: Context aware Computing for Firefighting. Lecture notes in computer science. In Proceedings of Second International Conference on Pervasive Computing, 2004, 87-105.

[Khalil, 05] Khalil, A., Connelly, K.: Improving cell phone awareness by using calendar information. In Proceedings of INTERACT, Rome, Italy, 2005.

[Lederer, 03] Lederer, S., Mankoff, J., Dey, A. K.: Who wants to know what when? Privacy preference determinants in ubiquitous computing. In CHI '03 Extended Abstracts on Human Factors in Computing Systems, Ft. Lauderdale, Florida, USA, April 05 - 10, 2003, 724-725.

[Mark, 08] Mark, G., Gudith, D., 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.

[Marti, 05] Marti, S., Schmandt, C.: Giving the caller the finger: collaborative responsibility for cellphone interruptions. In CHI '05 Extended Abstracts on Human Factors in Computing Systems, Portland, OR, USA, April 02 - 07, 2005, 1633-1636.

[Mehta, 96] Mehta, M., Agrawal, R., and Rissanen, J : SLIQ : A fast scalable classifier for data mining. (EDBT'96), Avignon, France, March 1996.

[Petersen, 08] Petersen, S. A., Cassens, J., Kofod-Petersen, A., Divitini, M.: To be or not to be aware: Reducing interruptions in pervasive awareness systems. In Proceedings of the Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies – UBICOMM, 2008, 327–332.

[Picard, 07] Picard W., Liu K.: Relative subjective count and assessment of interruptive technologies applied to mobile monitoring of stress; International Journal of Human-Computer Studies; volume 65, issue 4, 361 – 375.

[Quinlan, 93] Quinlan, J. R. : C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

[Savioja, 04] Savioja, P.: Alarms – Necessary Interruptions? Seminar on User Interfaces and Usability, HUT, Sober IT, 2004, 121-900.

[Sharmin, 06] Sharmin, M., Ahamed, S. I., Ahmed, S., Li, H.: SSRD+: A Privacy-aware Trust and Security Model for Resource Discovery in Pervasive Computing Environment. Computer Software and Systems Conference, 2006, 67-70.

[Spira, 05] Spira, J.B., Feintuch, J.B.: The Cost of Not Paying Attention: How Interruptions Impact Knowledge Worker Productivity, Basex, 2005.

[Stamm, 10] Stamm, K., Ahamed, S. I., Madiraju, P., Zulkernain, S.: Mobile Intelligent Interruption Management (MIIM): A Context Aware Unavailability System. In Proceedings of the ACM Symposium on Applied Computing, Switzerland, March 22 - 26, 2010, 599 - 600.

[Toninelli, 08] Toninelli A., Khushraj D., Lassila O., Montanari R.: Towards Socially Aware Mobile Phones. 7th International Semantic Web Conference, 2008.

[University of Michigan News Service] http://www.ur.umich.edu/0607/Apr02_07/02.shtml

[Yu, 06] Yu Z., Zhou X., Zhang D., Chin C., Wang X., Men J.: Supporting context-aware media recommendations for smart phones; Pervasive Computing, IEEE, Volume 5, Issue 3, July-Sept. 2006, 68 – 75.

[Ziebart, 05] Ziebart, B. D., Roth, D., Campbell, R. H., Dey, A. K.: Learning Automation Policies for Pervasive Computing Environments. In Proceedings of the Second international Conference on Automatic Computing. International Conference on Autonomic Computing. IEEE Computer Society, Washington, DC, June 13 - 16, 2005, 193-203.

[148 Apps.biz] http://148apps.biz/app-store-metrics/