International Research Symposium
Rajarata University of Sri Lanka

IRSyRUSl 2015

Integrating Context-Awareness with Reminder Tools

Yohani SR[1]*, Malaka JW[2]

Department of Physical Sciences, Faculty of Applied Sciences, Rajarata University of Sri Lanka[1].

Department of Computer Science and Engineering, University of Moratuwa[2].

ABSTRACT

Busy schedules in life lead people to set reminders to recall tasks to-do. But no sophisticated enough tools are provided to support users for handling reminders. Current reminder tools are triggered considering only the time factor. The main reason is the lack of using rich context that specify when a reminder should be presented to its recipient. Since Context-Awareness in mobile computing is more engaging area in gathering information about the user's current situation it would be ideal to integrate context awareness with the reminders to generate context-aware reminder where rich context is used that specifies when a reminder should be presented to its recipient. The aim of this research is to propose a model to integrate context-awareness to reminder decision making process and develop a Context-Aware reminder. In this research more focus has put on capturing user context using technologies like IoT. Context is captured under five categories; Location, User Activity, User Preference, Identity of the user and Date and Time. Focused on developing a context representing model and processing context data to infer valuable information in a mobile environment. Using above methods, a conceptual framework for a context-aware reminder has presented and finally "RemindMe", an android app has developed as the proof of concept. As the conclusion it can be stated that if more rich context can be captured, by integrating rich context for decision making more sophisticated context reminders can be generated, which makes the reminder process much more sophisticated

KEYWORDS: *Context-Awareness,Context-Capturing,Context-Representation, Mobile Computing, Reminder.*

[1] * Corresponding Author: Yohani SR: yohani.ysr@gmail.com

1. Introduction

Still we are not fit for the rapid pace we are living. Our brains can transfer information very rapidly, but what we experience with these busy schedules is procrastinate always. This is an era of over-booked schedules. When we became able to measure time exactly, work schedules became precise. But reminding the day's schedule by brain is not precise; the reason behind most of the laments of people complaining is forgetting some important matters in life. Which may lead to an unsatisfied day at the end? Simply put, life became so fast, so busy, and so pressured that, for many, it became stressful depriving us of pleasure. Stress is becoming an assumption of modern life and can lead to anxiety. That's why eventually people tend to go for technological solutions.

Every day we use special messages in order to help us remember future tasks. These messages, known as reminders, take many forms, such as memos, notes, emailing one-self, to-do lists, alarms and electronic calendar alerts. For example, a person may create a calendar alert to remind himself to buy some grocery on the way home. Most task-lists are time-sensitive with due dates, interval reminders and alerts. But, reminders can be more helpful when time based effort is supplemented with rich contextual information to present them at appropriate times in appropriate places considering user and his preferences. For example, a grocery list reminder is more helpful while passing his favorite supermarket on the way to home from work, rather than while at work or after getting home. In that case it is clear having context awareness included into reminders make it much supportive to people make life easier. There is growing interest in the use of context-awareness as a technique for developing pervasive computing applications that are flexible, adaptable, and capable of acting autonomously on behalf of users. It tries to offer a personalized experience based on situation aware and understands the needs of the user.

Although there are so many reminders available, still there is no context aware reminder which uses rich context of the user to predict the situation and trigger the reminders. Through this research our intension is to identify whether we can integrate context awareness with the reminder applications, if so develop a conceptual framework for a context aware reminder.

2. Literature Review and Conceptual Framework for the Context Aware Reminder

This part contains the conceptual design of the context aware reminder. Through literature it is clear context includes everything about the user that can be the place you are at, current activity you engaged in, your mood, with whom you are with etc. To understand the user it is required to capture as many information as possible. Context can be anything about user but for the easiness of design it has categorized into 5 major categories. They are user's identity, current location, the date and time, user's current activity and his preferences. Then context is captured under these categories.

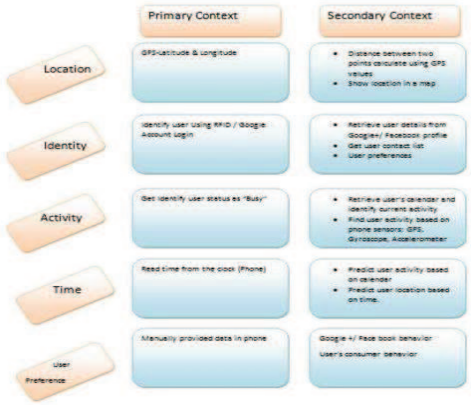

Fig. 1: Context Categorization.

Context has captured in two ways as primary context which get data directly from the sensors and secondary context information which acquired through processing of primary data from sensors. Both

has been used here to capture rich context about the user. Context widgets has introduced for each context category. Inside the widget context capturing method can be vary time to time with the advancement in the technology. Final goal is to get rich and more accurate context data from each widget to identify user context more precisely.

1.1. Capturing Context

1.1.1. Location

Due to increasing popularity of mobile devices context systems based location awareness is increasingly popular. GPS is the most popular method of capturing user's current location using a mobile phone but GSM network (cell towers) can be used to find location in finer granularity. That is for acquiring primary context data.

There are more secondary context capturing methods. For example in one application each entity (person /objects) which need to detect position is attached an ultrasonic wave emitter called "bat". "Bat" periodically emits signals which receivers detect and calculate proximity and then the position. Also we can capture user's logged in information such as to a PC and identify now the user is available at that place. Also using of network printers, card readers we can identify users current location. Floor embedded sensors can be used to identify the user is available there and smart objects as mentioned in [1], [2] also can be used to capture location. Those are secondary context capturing methods which need explicit settings in the environment proposed for IoT paradigm which is not possible to implement for mobile applications which the user is changing the location continuously. Digital image and video processing is another method to capture user location as mentioned in [3]. Voice recognition also can be used as a location detection technique according to [4]. GPS and GSM network have identified as the best method to detect user location considering the available infrastructure. GPS is identified as the best method after evaluating advantages of GPS vs. GSM. We can detect GPS coordinates or the location name and set it as the location widget output. Same method as in Context Toolkit [5].

1.1.2. Identity

Identity context is mainly focus on identifying the user. Detecting the electronic NIC with microchip which includes the personal information of the owner is possible within the country, driving license also can be used. Retina detection, RFID technology, voice recognition, video processing are some other methods as mentioned in [6]. Since most people are using social networking sites and they use smart phones to log in to them we can use social networking sites for capturing information about the user and identify who he/she is. For example Facebook can be used to get name, gender, preferences , date of birth information , Google + can be used to get name, gender, image, date of birth like all the information set as public. We can use one or more of these techniques to detect user identity, no matter what method is used finally the identity widget should give user identity using any method possible.

1.1.3. Activity

Activity detection is intended for identify the activities user engaged in and then can identify user's availability to complete a to-do task in reminders' perspective. Video image analysis is one method as mentioned in [7] but this needs more processing and intelligence to identify what user is doing. So that will be possible when IoT implement in the world. Using phone sensors also used to detect six positions of user activity such as sitting, standing, driving etc. according to [8]. According to [6] sensors that are easily available in office environment are used to generate information on the users' situations. This information is abstracted to higher-level contextual data by using a Bayesian machine learning approach. Gesture spotting with body worn sensors is another method as mentioned in [9]. But in mobile environment this is not possible, we want to detect users' activity with minimum or no interruption to their work. We can use social aware computing [10] and crowd sensing [11], but those are high level methods which need to establish sensors in the environment. OmniSense in [12] uses multiple phones in the environment to concurrent sensing of same location using different angles and Hidden Markov model to identify activity. In UV-point [13] also suggest collaborative sensing for social event coverage. But all these need an additional systems to establish which is not practical. One of the easiest method is to get user's calendar data, if we can get user calendar schedules by integrating it with the current time we can have a rough idea about user's current activity. Google calendar or inbuilt calendar in the phone can be used. Here also any available technology can be used inside the activity widget. Activity widgets act

as the source for getting user activity context.

1.1.4. Date and Time

Date and the time is easy to get using the phone clock. Some other technologies we can use are capturing light intensity of the environment, noise level and predict the time of the day. Seasonal based whether information can be used to identify what season is this. But in our country seasons are not visible therefore that technique is inappropriate. The easiest way is to get phone clock. Time widget provide current date and time.

1.1.5. User Preferences

User preference is an important context information when taking decisions, because the decision may vary based on the user preference. For example the book store a person A may not like may like by the person B, then when generating reminders we need to consider that as well. [14] Suggest a method to detect user preference based on the context using users' feedbacks. Some other advanced technologies are "Proem" suggested in [15] which is a wearable system that can write simple rules about user's preferences over nearby people. Most of the techniques provided in previous work suggest to observe user behavior and activity logs and create a knowledgebase. Using Facebook and Google predictions can be a possible method to detect users' preference, for example for the moment eBay like commercial sites use this knowledge to promote advertisements. Social media networking is the most successful way to detect user preference , since most people are using social media , data mining can be used to understand users mood, preference using status updates, search results, location present etc. Also we can let users to manually provide their preference, but in context aware paradigm it is better if we can get without user involvement. Using any one or more methods we can build up a knowledge base on user preference in each context and context widget is responsible for getting user preference, then it provide user preference based on the situation.

1.1.6. Representing Context

According to the literature there are multiple methods used for context representation, but most researchers [16], [17], [18], [19] provide proof that the ontology based context representation is the most successful. Therefore I have used ontology based context representation model for the conceptual design. To get a rich user context the data captured should be clearly stored in a precise manner using well understood, easy to implement ontology model.

Context information can be stored in the local memory of the phone or in the cloud. If store in the phone only, since the phone memory is low comparatively and need high processing that will drain the memory and battery power quickly. There is a cost of sending data to the cloud but it is inconsiderable when compared to user experience we can provide. Because any mobile user do not want to drain their battery in terms of having reminders on time. We can store data like location in the phone and send data need to process such as user preference in the cloud. Ontology is based on context reasoning. Given below is the ontology developed for the reminder system. Ontology is provide a vocabulary to represent knowledge about domain and describe specific situation in a domain. In this ontology model I have considered; Property, Relationship, Accuracy, Confidence Level of Decisions.

It has two parts high level which store general context knowledge and domain specific part which describe the properties in each sub domain. Domain specific part can vary when environment change. Therefore we can plug / unplug it according to the requirement. Context has classified into 5 categories and confidence level and dependence use parameters like Accuracy, Freshness of information, Certainty, Resolution as quality metrics of context. Given below is the design of the context vocabulary. Processing burden, domain specific ontology can vary when the environment changes. Therefore we can plug / unplug it according to the requirement. Context as classified and confidence level and dependency also associated with it. Parameters like Accuracy, Freshness of information, and Resolution has used as quality metrics of context.

TABLE I: User Context Representation using Ontology

Context Type	Value
User:Identity:Name	{String}
User:Identity:Email	{String :%@%.%}
User:Location:Coordinate:Latitude	{Double}
User:Location:Coordinate:Longitude	{Double}
User:Location:Interest	{High,Medium,Low}
User:BusyState	{Idle,CheckAvailability,Busy}
User:Action	{Walk,Sit,Drive}

TABLE II: Environment Context Representation using Ontology

Context Type	Value
Environment:Sound:Intensity	{Silent,Moderate,Loud}
Environment:Temperature	{Cold,Normal,Hot}
Environment:State	{Rainy,Cloudy,Windy,Normal}
Environment: Location: Building	{In,Out}

Using the ontology, the context reasoning happens in the following order.

Ex: Location (Sam, Car) ^ Activity (Sam, Driving) ^ Time (>, 05:00h) → Activity (Sam, GoingHome)
Whether (Rainy) ^ Location (Sam, PlayGround) → ~Do (Play)
Activity (Sam, Available) ^ Location (Grocery, FoodCity) → ReminderCalculator (On)

As a conclusion in designing of context representation I choose ontology representation, only the location data is sent from the app whenever the location changes , other data stored in the cloud and processed whenever location changes or periodically. Since most of the data does not change with the mobility that is easier to process data in the cloud and less costly to send data to cloud in terms of processing within the mobile phone and draining the battery.

1.1.7. Processing Context Information

Context processing is twofold. First we need to process context to identify user activity, then we need to process the context to match context with the reminder requirements to decide whether any reminder should trigger.

1.1.8. Context Processing to Trigger Reminders

1st Method

Assign each to-do task to an agent who is responsible for identifying matching context. Then each agent should check all the five context categories classified in the previous section in the research. Then check the context matching percentage for each category. If all are not above certain threshold, no reminding.

Ex: Fill the fuel to

TABLE IV: Context Requirement for Fuel Filling Task

Context	Matching %
Identity	100%
Location	95%
Time	100%
Preference	20%
Current_Activity	80%

Here if the user is near to the filling station and time is within open hours, just because user does not like the filling station don't giving reminder is useless. Because things like fuel stations, user preference is not important if the fuel level is very low. Therefore that method does not provide best context matching

mechanism.

2nd Method

Assign a weight for each context category take probability value for each category and if it is greater than certain threshold then generate reminder.
TABLE V: Context Requirement for Buy a Book and Fill Fuel Tasks

Context	Buy Books	Fill Fuel
Location	1	1
Time	1	0
Preference	1	0
Current_Activity	1	0

This approach is better but what should calculate first and when to start checking the context should be included.

3rd Method
First match the location, if the location is matching then check for other context factors. If the combination context matching probability is above a threshold then trigger the reminder. Here context is divided into 2 levels. The problem in this approach is, it consider other factors if and only if the location context match. In a scenario where user is free and no tasks to do, then just because the location not match the reminder does not trigger, but if the reminder triggers, may be user can decide to complete that task as he has no other work to complete.

4th Method

First check the user's activity and availability status; If

- Idle → Provide the to-do list then user can select any task to complete at his choice and also user can change the status of the reminder to complete.
- Busy → No action is taken
- Available → Get the location, then time and suggest a work and trigger the reminder.

Therefore 2nd, 3rd and 4th methods have used in a combined manner to run the context matching algorithm.
Operation of the Context Matching Algorithm depending on two situations.

- Whenever user current location changed considerably. The distance consider as a location change can be vary.

c_lat→current latitude c_lon→current longitude
p_lat→previous latitude p_lon→previous
longitude
v→ minimum distance to consider as a location change(variable). Op→operator(>,<,<=,>=,=)

LocationChange({c_lat,c_lon},{p_lat,p_lon},op,v) → true/false

If LocationChange="True" calculate other context factors. If

LocationChange="False" no further actions.

LocationListener: Location change can be identified using Google Location API's LocationListener method. LocationListener is used for receiving notifications from the LocationManager when the location has changed. Then when ever location change notification appears the app can run the context matching algorithm.

- Periodically. Time interval can be defined as required.

If we only based on location change when the location change does not happen for a long time and user wait "idle" without doing any work, then time wastes. There should be a mechanism to identify reminders fit for such situations as well. For that we run context matching algorithm periodically.

CurrentTime→t2

Last location changed time→t1 Time interval→t
If (t2-t1)> t then run the context matching algorithm.

Or we can set context matching algorithm to run after every 10min after previous run. This is more suitable since no need to do processing to calculate the time interval.

1.1.9. Context Matching Algorithm

Context matching algorithm which has developed consider several factors to identify best situation to trigger reminder.
Location Manager calls LocationListener and then location updated. Then
Set LocationChange="true"
IfLocationChange=="true"

then send location (lat,lon) to the cloud.

Cloud contains information related to other context factors such as user identity, Activity of the used identified using Google Calendar and the availability of user based on the activity. Can get the current time, and also the preference of the user. Since these factors do not change frequently, and no effect from the mobility of the user there is no problem in storing them in the cloud. Thus reduce the data transfer cost from the phone to cloud significantly.

Appropriateness of the context factors to the task should be considered. But the better approach is to identify the context factors that must be considered for each task by the system itself. This need a knowledge base and artificial intelligence to identify that way.
Task1 (c1,c2, c3….)

Let ci, c2, c3 be the context factors we should consider for identify the best context to trigger reminder. Weight to each context factor can be given to. If we are near to the Motor Vehicles and Motor Traffic Registrar office and it is during working hours better to get that. In such case priority should be given as follows.
GetDrivingLicence: priority (location:0.4, time:0.4, activity:0.2) Then should do the calculation and obtain a value that depicts the context matching factor.
Let CMF be the Context Matching Factor, C_i p be the priority value for the context factor C_i, *and m* be the matching factor. Matching value is calculated as follows. To calculate this value we should set variables that decide what "m" value is applicable to the situation. For example if we consider location we can set;
m = 1, if difference between CurrentLocation and PreferredLocation is < 1m

m = 0.75, if difference between CurrentLocation and PreferredLocation is < 5m m = 0.5, if difference between CurrentLocation and PreferredLocation is < 10m m = 0, if difference between CurrentLocation and PreferredLocation is >10m

TABLE VI: Context Matching Factor

m Value	Definition
m=1	Best Match
m=0.75	Considerable equality in context
m=0.5	Can consider but not best
m=0	No match

$$CMF = \sum_{i=1} \quad (1)$$

Using (1) we calculate the context matching factor and if
CMF > t then that task is consider as ok to remind task. Let "t" be the threshold value. Finally all the ID's of selected tasks send to the phone via Google Cloud Messaging. Once the app gets the Task_ID that is ok to remind. The app should get the current location and remind the task with least difference between current and preferred location. These task should only be selected from the tasks send by context matching algorithm.
This algorithm is developed using critically evaluating the possible outcomes. Since there is no method to get user context from a knowledgebase or no previous decision success rate is stored implicitly, difficult to evaluate the success rate of the algorithm.
Where to process these context data is important due to less battery and memory inside the pfone. In "Clone Cloud"[20] researchers have proposed a mechanism to switch processing between the cloud and the phone. According to clone cloud they maintain a clone of the phone in the cloud then partition application automatically to optimize processing power and battery life. When necessary offload the

thread to clone and then merge the state after finish processing at cloud. This idea can be researched to integrate with this reminder as well because here also need large processing power when it comes to rich context.

1.1.10. Conceptual Framework Design

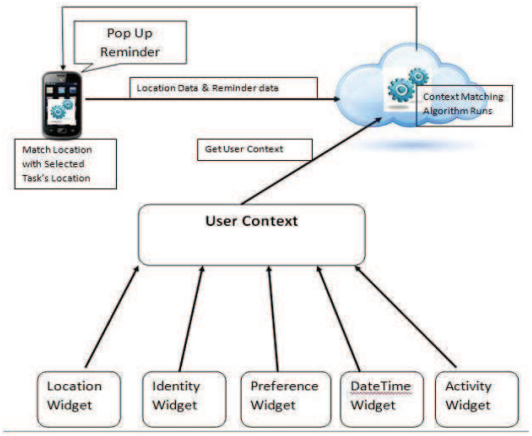

Fig. 3: High Level Conceptual Design of the Reminder.

3. Methodology

3.1. RemindMe: Proof of Concept Design

Proof of concept development solely depend on the available technologies to capture context data, no sensor implementation has done to acquire rich context information at this level. Simple mobile app called "RemindMe" has created as the proof of concept implementation. When the app is launched it directs to a screen where three options available.

Add to-do task → user can add to-do tasks. View
to-do task →User can vies to-do tasks.
About Me → this contains 6 options Location, DateTime, Activity, Identity, Preference and whether.
Users can click each option and view context capture under each category.

Add To-Do Tasks

User can input task name, and a description if required and select the category of the task using an

already defined task categories. Then set date of the reminder using a calendar widget. Set time using the time widget available in android. Location is the user's preferred location. Here Google Places API has used to get the location easily. User can select type of place such as hospitals, restaurants, book stores, when user click "Find" places will load in the Google map integrated and then user can select the location. Priority level is set to reminder. Priority can be High, Medium and Low. Then user can "Save" or "Delete" the task To-Do. When save it save in the phone memory and also data sent to Google Cloud.

View To-Do Tasks

Users can view the tasks To-Do, when viewing it appear as a list and for each list item there is a drop down list user can select the status of the To-Do task. When first create the task it is default set to "Pending" state which indicate the task has not completed yet. If task completed it set to "Completed" state. If the set time has already expired and context has not been met then set to "Expired" state. The intension of this view option is: whenever the user activity status is "Idle" user can view the tasks To- Do and, if user decided to do any task at that time , he can set the status using the drop down menu.

About Me

This option is used with the intension of displaying the context data captures about the user. Location: use both GPS and GSM technology to identify user current location. Location is given in latitude, longitude coordinates and also view in the Google Map. Google Geofencing API can be used to trigger reminders when enter or exit from a defined area. Identity: user information such as name, DoB, Gender, Phone no, and user image is captured using Google+. All the information publicly available can be viewed here. Activity: User Calendar details are captures using Google Calendar, if the users have scheduled work in the calendar the user Activity_Status is set to "Busy" otherwise "Available". Preference: Can be captured through Facebook and Twitter updates and Google search results, and also users can used manually set user preference. Weather: Is captured using Google Weather API. DateTime: captures through phone clock and calendar.

4. Test Results

Following test cases have used to test the accuracy of the ReminMe application developed as the proof of concept. There are three major events where reminders trigger.

- If the context matching percentage is above a given threshold value the reminder will generate automatically.
- User can explicitly set one or more factors to consider when generating reminders. Then when ever those factors will match the reminder will trigger regardless of the exact context match.
- Whenever the context has not matched until the date and time expires, then before reach the time an alarm occurs to inform the user that the particular reminder item will expire in future.

Sample test cases are given in the table below.

TABLE VII: Sample Test Cases

To-Do Task	Location	Activity	Date	Time	Priority	Reminder	Conclusion
Task1	Current Location	Set Calendar Event: Busy	Today	Now	High	No	User is busy, therefore no reminder occurrence
Task 2	Current Location	Available	Today	Now	High	Yes	User available and context match
Task 3	New York	Available	Today	Now	High	No	Location does not

							match
Task 4	Dematagoda	Available	Tomorrow	Set	High	Yes	When near to Dematagoda
Task 5	Dematagoda	Busy	Tomorrow	Set	High	No	User is busy
Task 6	Ibbagamuwa	Available	Set Future Date	Set	High	No	Context does not match
Task 7	Ibbagamuwa	Available	Today	1hr in future	High	Yes	Since no context match happen and the task is going to expire. Therefore reminder occurs to warn.

5. Conclusion

The major focus of this research is to design a conceptual framework for a context-aware reminder. Although context awareness is an upcoming trend there is less work done under the area of context aware reminders due to limitations in context capturing methods, and also representing large amount of data about a user or an entity and processing them to get rich user context is difficult. First, we tried to identify the context information that can be captured about a user and, divided that into five major categories; Location, User preference, User current Activity, Date and Time, Identity of the user. Literature based evaluation has done on capturing user context. Lack of real world implementation in IoT and sensors has made capturing user's exact context bit difficult. But higher priority has given here on capture context because once context is captured accurately that can be used in any application. So in here more work has done to identify context capturing method and did a literature based evaluation on those techniques. Then context widget is used for each context category that simplifies the process. Because inside the context widget any method or technique can be used to get the context information what matters is the final information it provides. Therefore we can use one or more techniques to capture context. After capturing context, studies had done to identify best representation model and processing mechanisms. Then developed the framework for context aware reminder which uses rich user context. But in the real world those mechanisms cannot be used to capture context due to lack of sensor establishment and sensing techniques. Need advanced sensing techniques to get precise user context. Although development became difficult due to lack of resources if we can get rich user context, this would be an ideal framework that helps to reduce the burden from human head in advanced and make life much smoother in future.

6. Future Work

Once rich user context is captured the key issue arouse is privacy of context information. With the development in this paradigm the amount knows about the user will get higher. When someone knows more about you that automatically violates privacy. Privacy guarantees are hard and also expensive to implement. But, user should be able to have control over their contextual information and over who may gain access to it. We propose under future work more attention should pay to ensure privacy of user

information. In this piece of work type of reminder is given by the users, but since this is a context aware system it is better if the type of reminder such as whether using alarm, message, SMS, or vibrations can be identified by the system itself. Also aggregating rich context is essential, mentioned here is a conceptual framework for a context aware reminder, future researchers can extend the context capturing methods and get rich user context.

7. References

[1] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the internet of things," Internet Comput. IEEE, vol. 14, no. 1, pp. 44–51, 2010.

[2] J.-P. Vasseur and A. Dunkels, Interconnecting smart objects with ip: The next internet. Morgan Kaufmann, 2010.

[3] M. Abo-Zahhad, S. M. Ahmed, and M. Mourad, "New Technique for Mobile User's Location Detection, Future Prediction and their Applications."

[4] O. Bucur, P. Beaune, O. Boissier, and others, "Representing context in an agent architecture for context-based decision making," in Proceedings of the Workshop on Context Representation and Reasoning (CRR'05), Paris, France, 2005, vol. 5.

[5] A. K. Dey, G. D. Abowd, and others, "The context toolkit: Aiding the development of context-aware applications," in Workshop on Software Engineering for wearable and pervasive computing, 2000, pp. 431–441.

[6] M. Martin, O. Brdiczka, D. Snowdon, J.-L. Meunier, and others, "Learning to detect user activity and availability from a variety of sensor data," in 2013 IEEE International Conference on Pervasive Computing and Communications (PerCom), 2004, pp. 13–13.

[7] A. K. Dey and G. D. Abowd, "CybreMinder: A context-aware system for supporting reminders," in Handheld and Ubiquitous Computing, 2000, pp. 172–186.

[8] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," Commun. Mag. IEEE, vol. 48, no. 9, pp. 140–150, 2010.

[9] H. Junker, O. Amft, P. Lukowicz, and G. Tröster, "Gesture spotting with body-worn inertial sensors to detect user activities," Pattern Recognit., vol. 41, no. 6, pp. 2010– 2024, 2008.

[10] P. Lukowicz, A. Ferscha, and others, "From context awareness to socially aware computing," IEEE Pervasive Comput., no. 1, pp. 32–41, 2011.

[11] M. Demirbas, M. A. Bayir, C. G. Akcora, Y. S. Yilmaz, and H. Ferhatosmanoglu, "Crowd-sourced sensing and collaboration using twitter," in World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a, 2010, pp. 1–9.

[12] H.-T. Cheng, S. Buthpitiya, F.-T. Sun, and M. L. Griss, "OmniSense: A Collaborative Sensing Framework for User Context Recognition Using Mobile Phones," HotMobile10 Annap. MD, 2010.

[13] X. Bao and R. R. Choudhury, "VUPoints: collaborative sensing and video recording through mobile phones," ACM SIGCOMM Comput. Commun. Rev., vol. 40, no. 1, pp. 100–105, 2010.

[14] K. Henricksen and J. Indulska, "Developing context-aware pervasive computing applications: Models and approach," Pervasive Mob. Comput., vol. 2, no. 1, pp. 37–64, 2006.

[15] G. Kortuem, “Proem: a middleware platform for mobile peer-to-peer computing,” ACM SIGMOBILE Mob. Comput. Commun. Rev., vol. 6, no. 4, pp. 62–64, 2002.
[16] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang, “An ontology-based context model in intelligent environments,” in Proceedings of communication networks and distributed systems modeling and simulation conference, 2004, vol. 2004, pp. 270–275.
[17] M. Mikalsen and A. Kofod-Petersen, “Representing and reasoning about context in a mobile environment,” in Proceedings of the First International Workshop on Modeling and Retrieval of Context. CEUR Workshop Proceedings, 2004, vol. 114, pp. 25–35.
[18] A. Padovitz^1, A. Zaslavsky^1, and S. W. Loke, “A unifying model for representing and reasoning about context under uncertainty,” 2006.
[19] S. W. Loke, “Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective,” Knowl. Eng. Rev., vol. 19, no. 03, pp. 213–233, 2004.