2016 International Conference on Advances in ICT for Emerging Regions (ICTer): 216 - 221

Integrating Context-Awareness with Reminder Tools

Yohani S. Ranasinghe[1], Malaka J.Walpola[2]

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

yohani.ysr@gmail.com

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

malaka@cse.mrt.ac.lk

***Abstract*— People use reminders to recall tasks to-do. But sophisticated enough tools are unavailable. Context-Awareness in mobile computing is more engaging area in gathering information about the user's current situation and would be ideal to integrate with the reminders to generate context-aware reminder where rich context is used to identify when a reminder should be presented to its recipient. Here we have proposed a model to integrate context-awareness with reminder's 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. Also developed a context representing model and processing context data to infer valuable information in a mobile environment. Then a conceptual framework for a context-aware reminder has presented and finally "RemindMe", an android app has developed as the proof of concept.**

***Keywords*— Context-Awareness, Context-Capturing, Context-Representation, Reminder.**

I. INTRODUCTION

Still we are not fit for the rapid pace we are living. 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 become 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. 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. 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 favourite 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.

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.

II. CONCEPTUAL FRAMEWORK FOR THE CONTEXT-AWARE REMINDER

Through literature it is clear that context includes everything about the user. To understand the user it is required to capture as many information as possible. After literature study context has categorized into 5 major categories for designing purposes. 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.

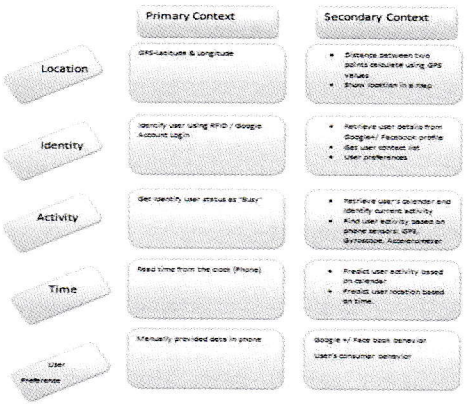

Fig. 1 Context Categorization.

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.

A. Representing Context

According to the literature there are multiple methods used for context representation, but most researches [1], [2], [3], [4] have proved that the ontology based context representation is the most successful. Context information can be stored in the local memory of the phone or in the cloud. Store in the phone only is not acceptable because 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. Mobile users 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 which need to process such as user preference to 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. Given below is the design of the context vocabulary.

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)

Only the location data is sent from the mobile app whenever the location changes, other data stored in the cloud and processed whenever location changes and or periodically. Since most of the data does not change with the mobility it 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.

B. 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 with the reminder requirements to decide whether any reminder should trigger.

1) 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 reminder generation.

Ex: Fill the fuel to a vehicle

TABLE III: 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 not giving a reminder is useless. In this scenario 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, consider only those factors when generating the matching percentage value, if it is greater than certain threshold then generate reminder.

TABLE IV: 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

does not match the reminder does not trigger. Better providing the user a chance to decide to complete that task when 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 considering other context factors matching 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.

I. Whenever user current location changes considerably. The distance considered to identify 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 changes, a notification appears and then the app can run the context matching algorithm.

II. 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 in 10 minutes interval. This is more suitable since no need to do processing to calculate the time interval.

2) Context Matching Algorithm:

Context matching algorithm consider several factors to identify best situation to trigger reminder.

- Location Manager calls LocationListener and then update the location.
- Then Set LocationChange="true"
- If LocationChange=="true" then send location (latitude , longitude) to the cloud.

Cloud contains information related to other context factors such as user identity, Activity of the user 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.

Task1 (c1, c2, c3....)

Let c1, c2, c3 be the context factors we should consider to identify the best context to trigger reminder. Weight to each context factor can be given. For example if we are near to the Motor Vehicles and Motor Traffic Registrar Office and it is during working hours and need to renew our license. In such case priority should be given as follows.

GetDrivingLicence:priority(location:0.4,time:0.4,activity:0.2)

Then need to do the calculation and obtain a value that depicts the context matching factor.

Let CMF be the Context Matching Factor, C_ip 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 V: 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

$$CMF = \sum_{i=1}^{n} C_i\, p * m \qquad (1)$$

Using (1) we calculate the context matching factor. Let "t" be the threshold value and if CMF > t then that task is consider as ok to remind task. Finally all the ID's of selected tasks send to the phone via Google Cloud Messaging. Once the app gets the Task_IDs those are 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 phone. In "Clone Cloud"[5] 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 because it requires large processing power when it comes to using rich context.

C. Conceptual Framework Design

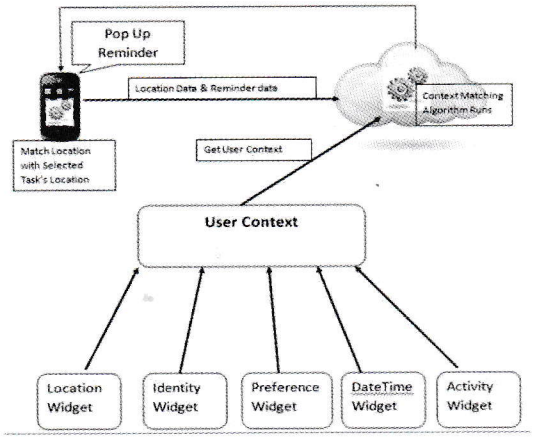

Fig. 2 High Level Conceptual Design of the Reminder.

III. 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 (Fig. 3).

Add to-do task → user can add to-do tasks.

View to-do task →User can view to-do tasks.

About Me → this contains 6 options Location, DateTime, Activity, Identity, Preference and whether (Fig. 4). Users can click each option and view context capture under each category.

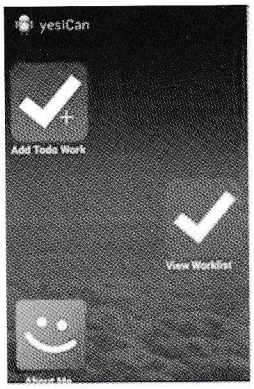

Fig. 3 "RemindMe" App Main Menu.

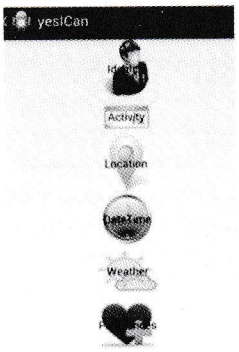

Fig. 4 "AboutMe" Main Menu.

A) 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 (Fig. 5). 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 is sent to Google Cloud.

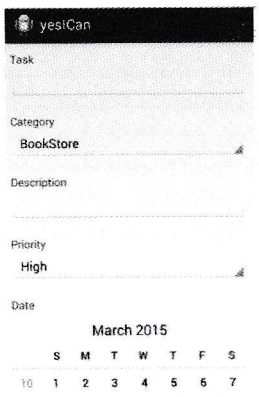

Fig. 5 Add a Task To-Do

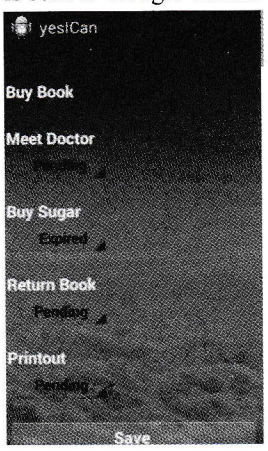

Fig. 6 View To-Do List

B) 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 (Fig. 6). When first create the task the default status is "Pending" 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.

TABLE VI: Sample Test Case

To-Do Task	Location	Activity	Date	Time	Priority	Reminder	Conclusion
Task1	Current Location	Set Calendar Event: Busy	Today	Now	Medium	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.
Task 8	Kandy	Available: Idle	Future Date	Set	Low	No	User can select any preferred task To-Do
Task 9	Current Location	Set Calendar Event : Busy	Today	Now	High	Yes	High Priority Task
Task 10	Colombo	Available	Tomorrow	Now	Low	No	Date is not matching

C) 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 captured 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 manually set user preferences. Weather: Is captured using Google Weather API. DateTime: captures through phone clock and calendar.

IV.TEST CASES

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 Table VI.

V. CONCLUSION

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, it is possible to build context aware reminders and this would be an ideal framework that helps to reduce the burden from human head in advanced and make life much smoother in future.

VI. 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, such as whether using alarm, message, SMS, or vibrations ,is given by the users, but since this is a context aware system it is better if the type of reminder 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.

ACKNOWLEDGEMENT

I would like to offer my sincere thanks to Dr.Shantha Fernando for his enduring guidance and encouragement and support.

REFERENCES

[1] 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.

[2] 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.

[3] A. Padovitz, A. Zaslavsky, and S. W. Loke, "A unifying model for representing and reasoning about context under uncertainty," 2006.

[4] 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.

[5] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in Proceedings of the sixth conference on Computer systems, 2011, pp. 301–314.