A Preference-based Application Framework for Resource-bounded Context-Aware Agents

Ijaz Uddin[1] and Abdur Rakib[12]

[1] School of Computer Science
The University of Nottingham Malaysia Campus, Semenyih, Malaysia
{khyx4iui,Abdur.Rakib}@nottingham.edu.my
[2] Department of Computer Science and Creative Technologies
The University of the West of England, Bristol, United Kingdom
Rakib.Abdur@uwe.ac.uk

Abstract. Context-awareness is an essential component of mobile and pervasive computing. It refers to the concept that an application understands its context, reason about its current situation, and provide relevant information and/or services to the users. One of the main challenges of context-aware distributed mobile computing is the dynamic adaptation to changes in the resource-bounded operating environment with user preferences. For example, a depersonalized context-aware application may exhibit behavior that is not anticipated by its user in a given situation. In this paper, we present a personalized preference model for resource-bounded context-aware applications, which provides support for the development and execution of context-aware applications using a declarative language. We implement a simple example system that demonstrates the effectiveness of the approach in a real-world scenario.

Keywords: Context-aware agents, Rule-based reasoning, Android SDK, Smart phone, User Preference.

1 Introduction

In distributed mobile and pervasive computing research, context-awareness has emerged as an effective design and implementation approach for building adaptive smart-space applications. These applications rely on the use of current contextual information, and their dynamic adaptation to changes in the operating environment provides a high level of automation with very minimal or no user intervention. In developing smart-space context-aware applications, smart phones and wireless sensor technology play an important role. Smart phones have a variety of embedded sensors that can be used to automate data collection and provide a platform to infer rich contextual data about users, including location, time, and environmental condition. This is known as customized information according to the specific context. To be more precise, these sensors can be used to gather the contextual information of a user or to manipulate the context. Different notions of context have been studied across various fields of computer science and various physical and conceptual environmental aspects can be included in the notion of context [1]. Among others, Dey et al. [2] define a context-aware

system as a system which uses context to provide relevant information and/or services to its user based on the user's tasks.

There has been considerable work in the context-aware systems literature on context modeling and reasoning approach in general (see for example, [3–6]) and on context-aware reasoning based on user preferences in particular (see for example [7–10]). Much of this work aims at how can semantic (ontology-based) and/or other techniques be utilized for context-modeling, knowledge sharing and reasoning about context for pervasive computing systems. However, well developed theoretical foundations considering their resource-boundedness features are still lacking. The resources include the time, memory, and communication bandwidth required by the context-aware devices or agents to achieve a goal. In recent work [11–13], Rakib et al. have developed formal logical frameworks showing how context-aware systems can be modelled as resource-bounded multi-agent reasoning agents. In this paper, we extend and enhance our previous work [12, 13] by customizing user preferences to enable the personalization of resource-bounded context-aware applications.

The rest of the paper is organized as follows. In Section 2, we present our context-aware system modeling framework. In Section 3, we present the design and implementation components of a context-aware user preference framework, which extends the existing framework [13] to allow personalized services. In Section 4, we present a system specification, sensor data acquisition, and discuss the results of an experiment, and conclude in Section 5.

2 Context-aware system modelling framework

We adopt the model of multi-agent context-aware rule-based systems developed by [12]. In rule-based techniques, a context-aware system composed of a set of rule-based agents, contexts are represented using first order terms and firing of rules that infer new contexts determine context changes and represent overall behaviour of the system. In order to model contexts and rules we use ontological approach. A rule has the following format:

$m : P_1, P_2, \ldots, P_n \rightarrow P_0 \ : F : CS$ where $n \geq 0$.

where m is the rule priority. Each P_i is an atomic formula of the form $p(t_1, t_2)$, $Ask(i, j, p(t_1, t_2))$ or $Tell(i, j, p(t_1, t_2))$, where i and j $(i \neq j)$ represent agents, p is a predicate symbol and the t_k are terms. Where Ask and $Tell$ are special atoms used for communication between the agents [12]. Each term is either a constant symbol or a variable. Every variable occurring in a rule is universally quantified, and its scope is the clause in which the variable occurs. Every variable appearing in the head must also appear in the body of a rule. The "$\rightarrow$" is read as *if* and "," as *and*. The atom P_0 is called consequent (or head) of the rule and the conjunction $P_1, P_2, \ldots, P_n$ is the body of the rule. If $n = 0$, then the body is equivalent to TRUE and is called a fact otherwise it is a rule. The flag F, a placeholder, associated with every rule is used to specify the type of the rule. For instance, the character 'G' is used to represent a rule containing a Goal statement, which indicates that a certain rule execution results in goal

achievement. The character 'C' represents the communication rules, which can trigger a communication between agents (devices). The character 'D' represents the deduction rules. The indicator CS says which set the rule belongs to, and is mainly used for the preferences set generation, which is explained in more detail in the following section.

In our framework, we consider systems having constraint on various resources, namely time, memory, and communication. This is because many context-aware systems often run on tiny resource-bounded devices, including PDAs, smart phones, GPS system, and wireless sensor nodes. These devices usually operate under strict resource constraints, e.g., battery energy level, memory, processor, and quality of wireless connection. The logical framework developed in [12] allows us to describe a set of context-aware non-monotonic rule-based reasoning agents with bounds on computational (time and memory) and communication resources. In [13], we extended the theoretical work [12] by implementing the ontology and logic based framework using the Google Android SDK and smart phones. In this paper, we extend and enhance our previous work [12, 13] by customizing user preferences to enable the personalization of resource-bounded context-aware applications. We also discuss further experimental progress of an example system. As in our previous work [13], we lacked some sensors which were then replaced by a simulated device. In the current setting, we use actual external sensors and successfully integrated them into the framework to generate experimental results considering a real world scenario.

3 Preference in context-aware agents

In this section, we discuss the extended framework that allows defining components to provide personalized services. In order to implement user preferences, we add an extra preference manager layer and keep the original working inference engine [13] intact. The main idea of user preference is to select a subset of rules based on preferences, and the inference engine, instead of going through all the rules, will only process selected rules. The whole process is composed of different steps and modules which are explained in the following sections. The preference manager layer consists of Preference Set Generator(PSG), Context Monitor(CM), Context Set(CS), and Context of Interest (COI) provided by the user beforehand. Figure 1 shows how these components are related to each other.

3.1 Context set

The context set(CS) component is basically a column added to the rule base. A literal in this column against a rule works as indicator for that particular rule. It indicates if a rule belongs to a particular set of rules that may infer a specific contextual information, e.g., *Person(?p), OfficeRoom(?o), hasLocation(?p, ?o) → inOffice(?p, ?o)* with indicator "L" in CS can be attributed towards the contextual information about user's current location, which represents that the rule belongs to a group of rules that are part of location rules. The CS may contain multiple indicators, for example, if user location and his blood pressure

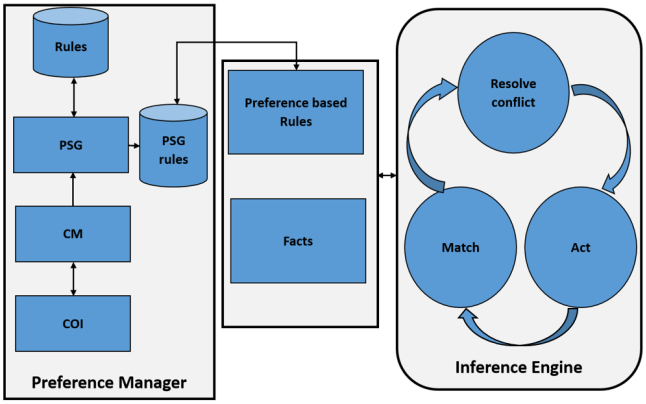


Fig. 1. Preference generation overview

mentioned in the same rule then CS can indicate both contexts defined with two different literals. The reason for such indication comes handy when the user preference is required. These all CS indicators can easily indicate the contexts included in a rule. For example, a user may want preference based on location only. So the preference set will add all the rules which CS indicates the location. It is pertinent to mention that any rule that does not have any CS indicator is a general rule, represented by "-" in the context set, and will be added to every sub set that is created for a preference set.

3.2 Context monitor

The context monitor (CM) component holds the Context of Interests (COI) of a user, i.e., it holds the values provided by the user. Context monitor after reading the values passes them to the Preference Set Generator(PSG). The PSG defines a sub set of rules based on the user preferences, called preference set. This subset is then passed to the inference engine for processing. Context monitor actively monitors the contexts of interests. Any change in the context is forwarded to the PSG to derive a new set of rules to be processed according to the changed preference(s).

3.3 Preference set generator

The preference set generator(PSG) is the main part which gives the framework an ability to provide personalized services. Since we have added CS to the rules for indication purpose, we need a layer that can work as intermediary between the user and the rule-base. This layer provides a sub set of rules which are personalized set of rules for a current context of the user. The PSG receives instructions from the CM to derive a sub set of personalized rules. The rule base of an agent consists of a variety of rules; some rules may never get a chance to execute while some others may be actively executed. In order to generate a

sub-set of the rules, the PSG has to consider the contexts that are of interest to the user.

3.4 Working mechanism

In this section, we show how these different parts work together to give preferences to the user. We have one main repository, where all the rules are stored. As the process starts, the COI is provided by the user and the values are retrieved by the CM component. The CM forwards the values to the PSG. The PSG further communicates with the rule base and picks only those rules that are of interest to the user based on the values specified in the COI. The PSG makes use of the CS to fetch the desired rules. This CS is specified by the user as COI. When the PSG rules are ready, these rules are provided to be used as the knowledge base for further processing. Comparing to our previous work we can see, that the whole system still works as described in earlier work [13]. However, the rules in the memory are replaced with only the preference based rules. Practically addition of preference layer is the major change, which reduces the overall burden from main inference engine and making it more efficient in terms of reducing rules.

4 System specification and sensor data acquisition

We have implemented the framework using both embedded sensors i.e., GPS and external sensors which are blood pressure monitor and heart rate monitor. In our experiment we have used three different agents, namely Patient care device (an Android powered smart phone), Care giver (an Android powered smart phone), and Blood pressure and heart rate monitor (BP device) (a Bluetooth-enabled device). The patient care device uses the low-level contexts from the BP device and infers high-level contexts using the set of rules in its knowledge-base. If a patient's condition is critical or an emergency scenario is detected, it interacts with the care giver agent. Care giver can be a registered doctor or nurse. The communication between the care giver and the patient takes place via SMS messages, while blood pressure and heart rate values are sent via Bluetooth to the patient care device. Figure 2 shows the patient monitoring device that we have used in our experimental setup. In our experimental model we classify different categories of blood pressure and heart rate based on the *Blood Pressure UK* and *New Health Advisor* data charts[3]. Based on the blood pressure and heart rate values, we have encoded a set of rules which are used to design and program our context-aware agents. However, due to space constraints, we have listed only few selected rules in Table 1, which are used by the patient care device.

[3] http://www.bloodpressureuk.org/BloodPressureandyou/Thebasics/Bloodpressurechart
http://www.newhealthadvisor.com/Normal-Heart-Rate-Chart.html

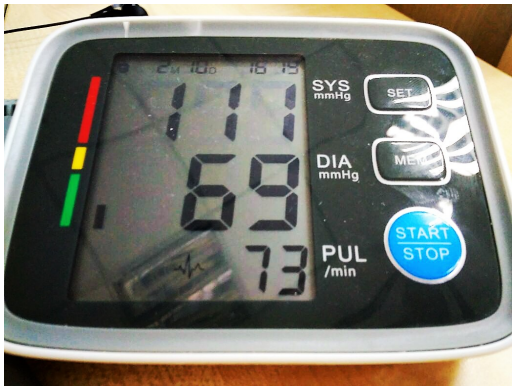


Fig. 2. Blood pressure and heart rate monitoring device

4.1 Sensor communication

In this section, we discuss two different mechanisms of the sensor data acquisition that are used in acquiring raw data from external and embedded sensors.

External sensor: The Blood pressure and heart rate monitoring device uses the Bluetooth low energy(BLE) communication settings. The BLE is relatively new technology that is very energy efficient compared to normal Bluetooth operation [14]. The communication mechanism and blood pressure measurement procedure follow certain steps, which are discussed here. As for the prototype design, a patient has to attach the strap to the upper arm and turn the device switch ON. When the device is ON, it starts sending some signals. The structure of the signal is in the following format: **[0xFD, 0xFD, 0xXX, 0xXX, 0x0D, 0x0A]** and is adopted for all the operations that the device carry throughout the measurements. The **XX** are replaced with other values such as errors or results and vary in size. When the sensor is turned ON, it sends the following signal or we can say it broadcasts the notification of its availability by sending **[0xFD, 0xFD, 0xA5, 0x0D, 0x0A]** every half a second. Any device nearby if replies back by sending the **[0xFD, 0xFD, 0xFA, 0x05, 0x0D, 0x0A]** signal, the connection will be established and the BP monitor will start the measurement. Once the measurement is taken, the BP monitor sends the result to the connected device in the following format: **[0xFD, 0xFD, 0xFC, Systolic value, Diastolic value, Heart rate,0x0D, 0x0A]**. This format indicates that the results are accurately taken and sent to the connected smartphone device. In case of an error, which may arise due to very low heart rate or inflation taking too much time or the low battery message, the BP monitor sends its corresponding signals to the connected smart phone device. These values are written to a file and saved in the smartphone's memory. Once written to the memory, the application program can access and read the contents of the file for further processing. In our case, Patient care device receives three values from the blood pressure device and one from its embedded GPS as location.

Embedded sensor: We have also simulated the emergency case, where the location is acquired using Google Play services API. The API is recommended by Google for accurate and faster location retrieval and also consumes less energy while acquiring the location. A fine grained location can also be determined using GPS, WiFi, and Cellular network. It can also update the location on a preset interval along with the distance. For example, if a location is acquired at point A, it will recalculate the location after the preset interval or if a user moves by a preset distance e.g., say 10 meters. In that case we always get an accurate location for a user. Once the location is acquired we further make use of the reverse geo-coding technique to retrieve a user readable format from the longitude and latitude that we capture from the sensor. The end result is an accurate human readable address. These sensed values or low-level contexts make no sense at all unless they are translated into meaningful high-level contexts. For that reason, we have also followed standard blood pressure and heart rate measurement guidelines, and encoded the expert knowledge into a set of Horn-clause rules.

4.2 Experimental results

In our previous work [13], implementation of the agents' inference engine and experimental results of a depersonalized context-aware application scenario have been provided. However, presenting detailed experimental results are out of scope of this paper. Nevertheless, we explain how the external sensor sends the raw data and how they are processed, and shows only the main rules that are most likely to be fired in the case scenario shown in Fig. 2. In Fig. 2, the blood pressure and heart rate monitoring device shows three values. The first value on top is the systolic value, the middle one is diastolic value, and the last one is the heart beats per minute. These three values appear on the screen and are forwarded to the patient care agent. Patient care agent has a variety of rules besides those presented in Table 1. For the experimental purpose, we assumed the data to be tested for the generic values of a healthy adult male. In this scenario the systolic value is between 90 and 120 and the diastolic value is between 60 and 80, which ultimately will trigger the rule resulting in the normal blood pressure category, i.e., the following rule:

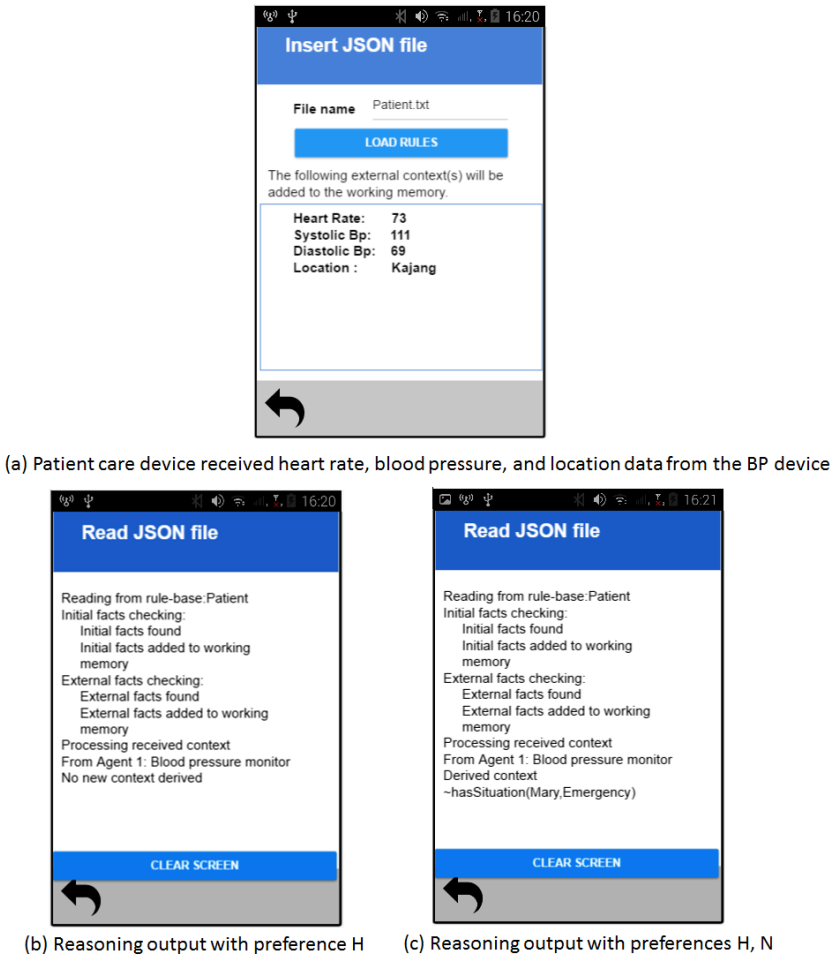
Person(?p),hasSystolicBloodPressure(?p,?sbp), hasDiastolicBloodPressure(?p, ?dbp),greaterThan(?sbp,90), greaterthan(?dbp,60), lessThan(?sbp,120), lessThan(?dbp,80)→hasBPCategory(?p,Normal)

Similarly the heart rate, as observed, falls within normal range and will trigger the following rule: *Person(?p),hasHeartRate(?p, ?hrt),greaterThan(?hrt,70), lessThan(?hrt,75)→ hasHRCategory(?p,Average)*

Hence both the blood pressure and heart rate categories fall in the normal range, which are the deciding factor in this case and the patient care agent will not interact with the care giver agent. Moreover, trying with different set of situations can produce different results, including false alarm. For example, blood pressure and heart rate readings could be high if they are measured while a person climbs stairs, and as a result the patient care agent may interact with a

Table 1. Blood pressure and heart rate rules

Blood pressure category rules				
Category	m	Corresponding rule	F	CS
Low BP	1	Person(?p), hasSystolicBloodPressure(?p,?sbp), hasDiastolicBloodPressure(?p, ?dbp), lessThan(?sbp, '90), lessThan(?dbp,60) ⟶ hasBPCategory(?p,LowBP)	D	-
Normal	1	Person(?p),hasSystolicBloodPressure(?p,?sbp), hasDiastolicBloodPressure(?p, ?dbp), greaterThan(?sbp,90), greaterthan(?dbp,60), lessThan(?sbp,120), lessThan(?dbp,80) ⟶ hasBPCategory(?p,Normal)	D	-
Pre high	1	Person(?p), hasSystolicBloodPressure(?p,?sbp), hasDiastolicBloodPressure(?p, ?dbp),greaterThan(?sbp,120), greaterThan(?dbp,80),lessThan(?sbp,140), lessThan(?dbp,90)⟶ hasBPCategory(?p,PreHigh)	D	-
High	1	Person(?p), hasSystolicBloodPressure(?p,?sbp), hasDiastolicBloodPressure(?p, ?dbp), greaterThan(?sbp,140), greaterThan(?dbp,90)⟶ hasBPCategory(?p, HighBP)	D	-
Heart rate category rules				
Category	m	Corresponding rule	F	CS
Athlete	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,48), lessThan(?hrt,55) ⟶ hasHRCategory(?p, Athlete)	D	-
Excellent	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,54), lessThan(?hrt,62) ⟶ hasHRCategory(?p,Excellent)	D	-
Good	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,61), lessThan(?hrt,66) ⟶ hasHRCategory(?p,Good)	D	-
Above Average	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,65), lessThan(?hrt,71) ⟶ hasHRCategory(?p,AboveAverage)	D	-
Average	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,70), lessThan(?hrt,75) ⟶ hasHRCategory(?p,Average)	D	-
Below Average	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,74),lessThan(?hrt,82) ⟶ hasHRCategory(?p,BelowAverage)	D	-
Poor	1	Person(?p), hasHeartRate(?p,?hrt), greaterThan(?hrt,81) ⟶ hasHRCategory(?p,Poor)	D	-
Some example rules to derive different situations				
Category	m	Corresponding rule	F	CS
Emergency	2	Patient(?p), hasBPCategory(?p,HighBP), hasHRCategory(?p,Poor) → hasSituation (?p,Emergency)	D	H
Emergency	2	Patient(?p), hasBPCategory(?p,PreHigh), hasHRCategory(?p,Poor) → hasSituation (?p,Emergency)	D	H
Emergency	2	Patient(?p),hasBPCategory(?p,Normal), hasHRCategory(?p,Poor) → hasSituation (?p,Emergency)	D	N
Emergency	2	Patient(?p),hasBPCategory(?p,LowBp), hasHRCategory(?p,Poor) →hasSituation (?p,Emergency)	D	L
Non Emergency	1	Patient(?p),hasBPCategory(?p,Normal), hasHRCategory(?p,Average) → ∼hasSituation (?p,Emergency)	D	N
Non Emergency	1	Patient(?p),hasBPCategory(?p,Normal), hasHRCategory(?p,AboveAverage) → ∼hasSituation (?p,Emergency)	D	N
Non Emergency	1	Patient(?p),hasBPCategory(?p,Normal), hasHRCategory(?p,Good) → ∼hasSituation (?p,Emergency)	D	N



(a) Patient care device received heart rate, blood pressure, and location data from the BP device

(b) Reasoning output with preference H

(c) Reasoning output with preferences H, N

Fig. 3. Sensed data and rule execution results using preference

caregiver, which can be considered as false alarm. However, more sophisticated rules can be added to deduce about the condition of the user based on the variable such as if the person is running, climbing etc. Furthermore, preferences can be used to personalize, where the results matter more important to the user. The preference set generated is based on the COI values that reside in the CS. For example, when we provide COI as rules which deal with high blood pressure only, indicated with the symbol H in the CS, the system will not produce any new context based on the values given above, shown in Fig. 3 (b). This is because when preference is applied only rules which are of type H will be added to the preference set along with other general rules if any, and will ignore other rules having preference types L and N. However, when the COI is changed from H to H,N which includes those rules that are dealing with both high and normal blood pressure, we get different results. In this case, when we run the application

again for the same input, we see that the system triggers the rule which infers *∼hasSituation(Mary, Emergency)* as shown in Fig. 3 (c).

5 Conclusion and future work

In this paper, we discussed and presented a preference model to personalization of resource-bounded context-aware applications. We also discussed the updated progress of the system and integration of the external sensors to the framework developed in our previous work in [13]. In future work, we would like to further narrow down the concept of preference so that it can be applied to the values of the contexts. In that case, a user will have control over the preference selection within the context rather than on rules.

References

1. Lieberman, H., Selker, T.: Out of context: Computer systems that adapt to, and learn from, context. IBM Systems Journal. 39(3-4), 617–632 (2000)
2. Dey, A.K.: Understanding and using context. Personal Ubiquitous Comput. 5(1), 4–7 (Jan 2001), http://dx.doi.org/10.1007/s007790170019
3. Ranganathan, A., Campbell, R.H.: An infrastructure for context-awareness based on first order logic. Personal Ubiquitous Comput. 7(6) (2003) 353364.
4. Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H., Malm, E.J.: Managing Context Information in Mobile Devices. IEEE Pervasive Computing 02(3) (2003) 4251.
5. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology Based Context Modeling and Reasoning using OWL. In: PERCOMW 04, Washington, DC, USA, IEEE Computer Society (2004) 18–22.
6. Kofod-Petersen, A., Mikalsen, M.: Representing and Reasoning about Context in a Mobile Environment. Revue dIntelligence Artificielle 19(3) (2005) 479498.
7. Barkhuus, L., Dey, A.: Is Context-Aware Computing Taking Control away from the User? Three Levels of Interactivity Examined. UbiComp 2003: Ubiquitous Computing, Volume 2864 of the series Lecture Notes in Computer Science (2003) pp 149-156
8. Stefanidis, K., Pitoura, E., Vassiliadis, P..: Modeling and Storing Context-aware Preferences. In Proceedings of the 10th East European Conference on Advances in Databases and Information Systems. Volume 4152 of the series Lecture Notes in Computer Science pp (2006) 124-140.
9. O. Coutand. A Framework for Contextual Personalised Applications. Dissertation thesis. The University of Kassel, 2008. ISBN: 9783899587746.
10. Hong, J., Suh, E., Kim, J., Kim, S.: Context-aware System for Proactive Personalized Service Based on Context History. Expert Syst. Appl. 36(4) (2009) 7448–7457.
11. Rakib, A., Haque, H.M.U., Faruqui, R.: A temporal description logic for resource-bounded rule-based context-aware agents. In: Context-Aware Systems and Applications. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 128, pp. 3–14. Springer (2014)
12. Rakib, A., Haque, H.M.U.: A logic for context-aware non-monotonic reasoning agents. In: Human-Inspired Computing and Its Applications. Lecture Notes in Computer Science, vol. 8856, pp. 453–471. Springer (2014)
13. Uddin, I., Rakib, A., Haque, H.M.U.: A Framework for Implementing Formally Verified Resource-Bounded Smart Space Systems. Mobile Networks and Applications (2017) 1–16.
14. Aguilar, S., Vidal, R., Gomez, C.: Opportunistic sensor data collection with bluetooth low energy. *Sensors*, 17(1):159, 2017.