A Resource-Aware Preference Model for Context-Aware System

Ijaz Uddin[1(✉)] and Abdur Rakib[2]

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

Abstract. In mobile computing, context-awareness has recently emerged as an effective approach for building adaptive pervasive computing applications. Many of these applications exploit information about the context of use as well as incorporate personalisation mechanisms to achieve intended personalised system behaviour. Context-awareness and personalisation are important in the design of decision support and personal notification systems. However, personalisation of context-aware applications in resource-bounded devices are more challenging than that of the resource-rich desktop applications. In this paper, we enhance our previously developed approach to personalisation of resource-bounded context-aware applications using a derived context-based preference model.

Keywords: Context-aware · Preferences · Personalisation
Defeasible reasoning

1 Introduction

Context-awareness is one of the core features of ubiquitous computing. While the concept of context-awareness exists since early 1990s [1], it has gained fast popularity in the recent years due to the evolution of smartphones and the growth in the usage of Internet and sensor technology. Nowadays, almost all modern smartphones are equipped with visually rich and dynamic user interfaces, as well as a range of sensors including, accelerometers, GPS, Gyro, pulse and finger print sensor. The embedded sensors in the smartphones can be used to acquire contextual data from various context sources, e.g., users, environments or other devices. The low-level sensed contextual data can be translated into machine-readable data for higher level context inference using e.g., a suitable knowledge representation and reasoning technique. In the literature, the term *context* has been defined in various ways within the context-aware computing research, however, one of the most widely accepted definitions was provided by [2] as *context* is

P. Cong Vinh et al. (Eds.): ICCASA 2017/ICTCC 2017, LNICST 217, pp. 3–13, 2018.
https://doi.org/10.1007/978-3-319-77818-1_1

any information that can be used to characterise the status of an entity. Common context types include the user-related context(e.g., profile, identity, activity, preference, location), physical or environment-related context (e.g., noise levels, temperature, wind speed, location, room number, time of day), and device-related context (e.g., resources, network connectivity, resolution). A system is said to be context-aware if it can adapt its behaviour to a given situation and provide relevant information and/or services to the users [1,2]. In the literature, various context modelling approaches and context-aware system development architectures have been proposed, however, ontology-based approach has been advocated as being the most promising one [3,4]. In our research, we model context-aware systems as ontology-driven multi-agent rule-based reasoning systems [5,6], where context is formally defined as *(subject, predicate, object)* triple that states a fact about the subject where—the subject is an entity in the environment, the object is a value or another entity, and the predicate is a relationship between the subject and object. That is, we model context as first order function free predicates, a context state corresponds to a belief state of an agent or content of its working memory, and firing of rules that infer new contexts may determine context changes and representing overall behaviour of the system [6]. In context-aware systems, user preferences play an important role in adapting their behaviour to satisfy the individual user in different contexts. The mechanism generally relies on implicit and/or explicit user, device, physical or environment-related context that manipulate working mechanism that control the way applications react to the context in use. For example, in our case only a subset of the rules of an agent's rule base could be active based on the given preferences. In this paper, we present and enhance our previously developed approach [7] to personalisation of context-aware applications using a derived context-based preference model. The main idea of our approach is that preferences are specified as derived or externally communicated/sensed context so that they can be easily controlled to personalise the system behaviour without modifying the internal settings or agent's program.

The rest of the paper is structured as follows. In Sect. 2, we briefly review closely related work. In Sect. 3, we discuss motivation for undertaking this study. In Sect. 4, we present the proposed context-aware preference model, which extends the existing framework [7] by incorporating derived-context based user preference. In Sect. 5, we discuss derived-context based user preference in more detail. In Sect. 6, we present a simple case study to illustrate the usefulness and effectiveness of the proposed approach, and conclude in Sect. 7.

2 Related Work

The use of preferences in context-aware systems for decision making and personalization has been a highly researched topic. For instance, incorporating preferences in context-aware applications, mainly in manipulating the context, storing, management and its use in the future has been a subject of interest to many researchers (see, e.g., [8–10]). Even the research in database technology has seen

the effect of personalised queries where the result of a query depends on the current context available [9]. However, these methods are used for developing resource-rich systems with large scale databases. Some more recent preference oriented works consider different approaches, e.g., in [11] authors use user profiling technique for storing contexts of different users. It matches all the rule instances with the facts stored in the working memory and the profile is loaded based on the current context. This approach perhaps requires extensive memory to run the system.

Similarly, context-aware recommendation applications are also part of user preferences, where an application is recommended to the user based on his past patterns. In [12], the authors have proposed a model for personalising recommendations and improving user experience by analysing the context in use. They have used ranking algorithms for context based items. The system integrates the social media to explore the user preferences and based on those preferences it personalises the user experiences.

As digital healthcare often designed to exploit recent advances in computing technology, traditional healthcare information systems make use of context-aware technologies to improve the quality of healthcare services. In [13], the authors proposed a context-aware system framework for automated assistance and independent living of senior citizens. It mainly focuses on the personalisation and adaption of preferences. Besides other tasks, a local context manager is used in order to process the data from low-level to high-level. The decision making module is the IDSS or intelligent decision support system, which is a cloud based service. This IDSS has in itself large number of reasoners such as *Lifestyle Reasoners and Management*, which works on different data types. The reasoner can store long-term data that have certain patterns or routines, which defines the lifestyle of some users. Thus it can detect changes and indicate changed behaviour of users in terms of their health status. In [14], the authors propose using defeasible logic rules to describe system behaviour and for modelling context-dependant preferences. Their work is closely related to our work presented in this paper. However, in our work we use defeasible reasoning to model and describe behaviour of the context-aware agents.

3 Motivation

The motivation for undertaking this study is that, the usage of social networks and cloud computing has dominated the context-aware platform by providing more resource-rich techniques on server/cloud. It is practically possible to scale a high end system with the use of resource-rich cloud computing. However, there is certainly attention required when systems are developed considering tiny resource-bounded devices. To add more, if a system is intended for elder care or patient care then the chances are that a patient might not have his social networking account or may not be using it actively. Development of a system which is independent of other services can be beneficial for rapid implementation of elder care or remote system where resources are limited. Further to this, our

previously developed externally received context-based preference mechanism [7] works on different indicators provided by the user to generate a preference set. However, there are some contexts which can not be obtained from external or embedded sensors, and a user might be interested in those contexts in order to generate the preference sets. For example, a context *Patient(Alan)*, the status of a person of being a patient can not be obtained from a sensor, instead it has to be derived using some rules. Based on the status of a person being a patient, the system can generate a preference set accordingly. Similarly, derived context based approach could be useful for generating a preference set when the context that was actually expected from an external source cannot be obtained perhaps due to a sensor malfunction. For example, if the contextual information of user's presence in his office cannot be received from the GPS, an agent may derive it using a set of rules and information obtained from a occupancy sensor. One such example can be found in the work by [15], which mainly deals with the survivor tracking at the current stage but can be evolved further to be used in elder care or patient care system. In light of the above literature, we propose a preference model suitable for implementing context-aware systems that run on resource-bounded devices. Furthermore, the preferences in our model are filtered through two different layers, one is generalised preference that deals with a particular context, e.g., preference required at office or home [7], second is when a conflict occurs between the rules of the preference set [14]. By incorporating these two different preference layers, we propose an approach aimed at providing preferences to the users with minimal usage of system resources and independent of any other services.

4 Resource-Aware Preference Model

The logical framework and its extension to accommodate preferences presented in [6,7] serve as the basis of the whole framework. In this paper, we extend our previous work [7] to incorporate preferences using a derived context-based preference model, while maintaining the resource utilisation factor intact [16]. Note that our approach to preferences is based on two levels. First level works on the basis of communicated/sensed or derived context, while second level assigns priorities to different rules to give preference to one rule over another to resolve conflicts. In [7], the preferences were based on the user provided or externally communicated/sensed contexts. However, the implicitly derived contexts were not considered to make changes to the preference sets. Here, we consider the derived contexts to be dealt as input in case if they are indicated to be the contexts of interest by the user. The structure of inference engine and internal set-up remain the same. However, some changes are made within the preference manger layer of the system architecture and to the point when new contexts are derived.

4.1 Context-Aware System Architecture

As mentioned before, we design context-aware systems as multi-agent rule-based reasoning agents. In general, there are several different ways agents in a multi-agent system can be programmed. In our case, programming agent behaviour

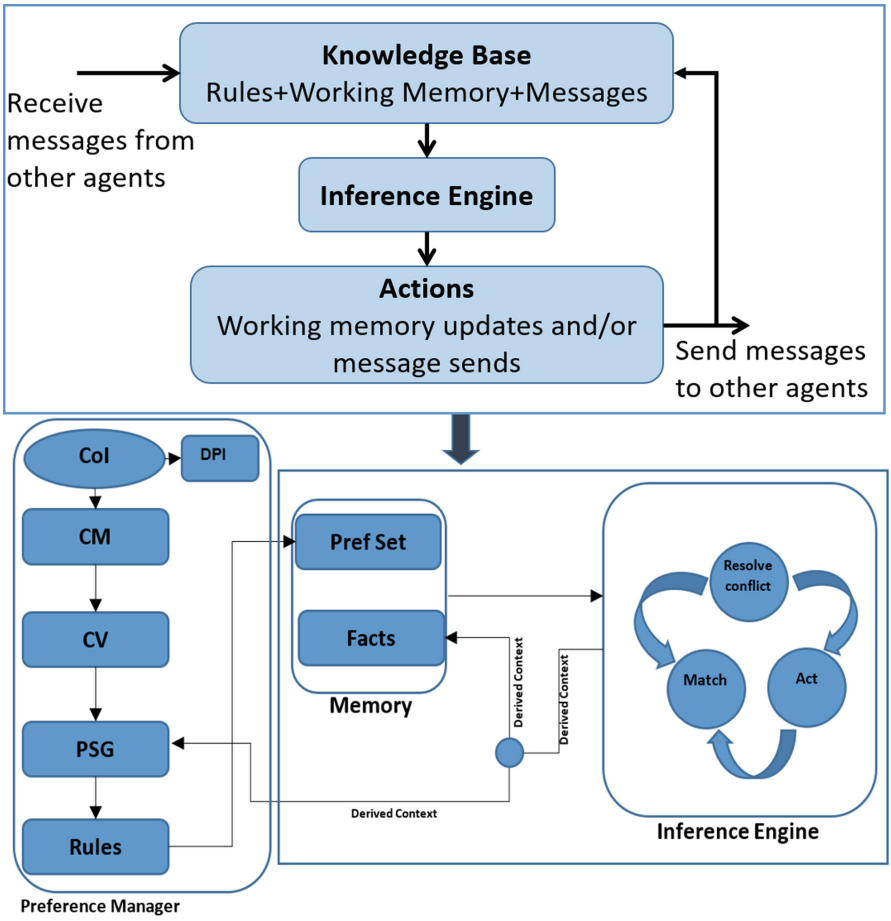

Fig. 1. System architecture and preference generation overview

using a declarative rule language consists in building a layered architecture using the Horn clause rules at the upper layer and Android Java is used in the lower layer to handle agent communication. The knowledge base is the upper layer of the architecture, which contains annotated ontology-driven rules (translated from OWL2 RL ontology augmented with SWRL rules). The upper part of the Fig. 1 represents the layered architecture of our system. A formal specification of the rule syntax is given in the following section.

4.2 Rule Structure

A typical rule format of our framework can be found in [7], while some changes are made when derived-context based preference is intended. The typical structure of a rule looks like: $m : P_1, P_2, \ldots, P_n \rightarrow P_0 \ : F : CS$ where $n \geq 0$.

The $CS(= \{-||P||\_P||tag\})$ is mainly used for the preference set generation. The different CS indicators are used by the framework to determine the nature of preferences required by the user. In case when we do not wish to attach a rule to any of the preference set then we can simply use it as a general rule that can be indicated by the "−" sign. That is, any rule with a "−" sign will be considered as a common rule and will be added to any preference set. The predicate P can be a context/fact, e.g., *hasLocation(Alan, UNMC)*. The predicate $\_P$ indicates that the rule attached to this format is only selected when P is derived by the inference mechanism. Thus, it is a potential context to be used as a preference only if an agent derives it by the inference mechanism. For example, *_hasLocation(Alan, UNMC)* is a potential context to be used as a preference, however, the preference set will be generated based on this preference if the context *hasLocation(Alan, UNMC)* is inferred by the agent confirming that the user is indeed located at *UNMC* (The University of Nottingham Malaysia Campus), and hence he expects preferred services available at *UNMC*. The tag indicator is used for general preferences and can be used to gather different rules into one group identified by the literal or tag given. For example, a rule with a tag of "L" may refer to the context related to location, hence all the rules with tag "L" are considered to be the members of the corresponding tag.

4.3 Preference Manager Layer

To incorporate the preferences, preference manager layer plays its role in managing the modules it carries, and to give a user the feel of personalization and also allows the inference engine to work with minimum overload. The general idea of the preferences provided is to extract a subset of rules from the whole rule base based on the user preferences. The preference manager layer is composed of Preference Set Generator (PSG), Context Monitor (CM), Context Set (CS), Context of Interest (COI), Context verifier (CV), and Derived Preference Indicator (DPI). The lower part of the Fig. 1 depicts the preference manager module and relationship between these components. The detailed description of the CS, CM and PSG can be found in [7]. Due to space limitations, we only briefly describe the newly added components.

- **Context Verifier (CV)** component is responsible for validating the contexts received from the sensors/agents and matches them with the user provided COI. If the COI matches with the sensed/received contexts then it can allow the PSG to generate the preference set. A straight forward example is location. If a user has COI *hasLocation(Alan, UNMC)* and the GPS sends the location as *hasLocation(Alan, Home)*, then it will drop the COI, as the location does not match with the COI. Hence the preference can not be added.
- **Derived Preference Indicator (DPI)** (or _COI) is responsible for generating a list of potential preferences from the COI. It matches a potential context with derived context in case a preference is enabled. If it finds a derived context that is being considered as a potential preferred context then DPI will send that context to the PSG. Unlike sensed/communicated context, derive context does not require validation and DPI directly sends it to the PSG.

To further elaborate the concept, let us suppose that an agent has a set of rules to model the behaviour of a person. Now a person can become patient if he is sick, which is a possibility. So, a system designer may add $\_Patient(Alan)$ as a derived preference. Which means that those rules related to the $\_Patient(Alan)$ will be added to the preference set once Alan gets sick.

5 Derived-Context Based User Preference

Since we have different indicators for the rules, it is necessary to determine the level of preferences required by the user. This mechanism is handled by the preference level monitor (PLM).

5.1 Preference Level Monitor (PLM)

Preference levels give user a choice of where the preferences are desired and up to which level the preferences are desired. The PLM can accommodate both the simple preference along with the facts/context value based preferences. As discussed in Sect. 4.2, the user can opt for any of the four different preference indicators. The Algorithm 1 goes through different checks to perform the better preferences and make the appropriate list of preferences. The algorithm presented in [17] has been revised to accommodate the derived context preference mechanism, changes are reflected in lines 16–22. One thing is to mention here is that the PLM Algorithm will make a separate list of derivable preference indicators, which will not be used by the CV, instead it will be passed once the contexts are derived. This is because, in advance, the CV will match the COI with the externally received contexts.

Since a system designer is aware of the different rules used to design the system and their possible outcome, it is fairly easy for him to use the preferences accordingly. In basic terminologies suppose we have a health care domain, where the system allows a user to monitor his blood pressure. The blood pressure can be categorised as High, Low and Normal levels besides declaring the user as a *Patient*. So, while keeping in mind that the possibility of a user to become a *Patient*, the *Patient* can be made as a derivable preference. Unless the user is derived as a *Patient*, the rules belong to the patient category will not be added to the corresponding preference set. In the next section, we explain the overall idea considering a simple case study.

6 A Simple Case Study

We consider a system consisting of a number of agents, including a person agent (Agent 1 represented by a smartphone) who is a user and may change his location detected by the GPS embedded into his smartphone. The user is also known to have his Blood pressure issues which is monitored by the BP device (Agent 2) and has heart rate monitor enabled (Agent 3). The user casually visits hospital

Input: **COI**: Current Context of Interest,_**COI**:Derivable COI,**R:** Rules, $\mathbf{F}_e$**:** Facts from external agents or sensors,$\mathbf{F}_d$**:** Facts derived, **CS:** Context Set, **Regex:** regular expression
Output: Preference Set based on COI

START
if *Regex(COI)==[a-zA-Z]* **then**
  Fetching Simple preference
  for *r→[R]* **do**
    if $\exists x \in$ ***COI*** *such that* $x \in$ ***CS[r]*** **then**
      Add r to Preference Set
    end
  end
end
else if *Regex(COI)==[a-zA-Z]+([a-zA-Z0-9]+)* ***OR*** *[a-zA-Z]+([a-zA-Z0-9]+,[a-zA-Z0-9])* **then**
  Fact-based preference of the form A(b) or B(b,c)
  for *r→[R]* **do**
    if $\exists x \in$ ***COI*** *such that* $x \in$ ***CS[r]*** *AND* $x \in \boldsymbol{F}_e$ **then**
      Add r to Preference Set
    end
  end
end
else if *Regex(_COI)==[a-zA-Z]+([a-zA-Z0-9]+)* ***OR*** *[a-zA-Z]+([a-zA-Z0-9]+,[a-zA-Z0-9])* **then**
  Derived-Context based preference of the form A(b) or B(b,c)
  for *r→[R]* **do**
    if $\exists x \in$ ***_COI*** *such that* $x \in$ ***CS[r]*** *AND* $x \in \boldsymbol{F}_d$ **then**
      Add r to Preference Set
    end
  end
end
else if ***CS[r]== "-"*** **then**
  Add r to general rule
end
END

Algorithm 1. PLM working algorithm

for the check up, and person agent can interact with Out Patient handling agent (Agent 4, located at Hospital). The user also has some preferences for his office which is located as UNMC. The office has an occupancy sensor (Agent 5), which can detect if the user is in the office or not.

6.1 Context-Based Preferences

As mentioned above, the user is not static and he may change his location time to time. When he arrives at hospital, his location is detected and processed to derive a new context being a patient. We will use this derived context to make

Table 1. Some example rules of Agent 1

Id	m	Rule	Identifier
R1	3	Patient(?p), hasBloodPressure(?p, Low) ⟶ hasSituation (?p, Emergency)	_Patient(Alan)
R2	3	Patient(?p), hasBloodPressure(?p, High) ⟶ hasSituation (?p, Emergency)	_Patient(Alan)
R3	2	Tell(2, 1, hasBloodPressure(?p, High)) ⟶ hasBloodPressure(?p, High)	_Patient(Alan)
R4	2	Tell(2, 1, hasBloodPressure(?p, Low)) ⟶ hasBloodPressure(?p, Low)	_Patient(Alan)
R5	1	Patient(?p), hasHeartRate(?p, Normal) ⟶ ∼ hasSituation(?p, Emergency)	_Patient(Alan)
R6	2	Tell(3, 1, hasHeartRate(?p, Normal)) ⟶ hasHeartRate(?p, Normal)	_Patient(Alan)
R7	1	Person(?p), GPS(?loc) ⟶ hasLocation(?p, ?loc)	-
R8	2	hasLocation(?p, Hospital), PatientID(101), hasPID(?p,101) ⟶ Patient(?p)	-
R9	2	Patient(?p), hasReason(?p, ?r), MedicalReason(?r) ⟶ isOutPatient(?p,?r)	_Patient(Alan)
R10	2	isOutPatient(?p, ?r) ⟶ Tell(1, 4, isOutPatient(?p, ?r))	_Patient(Alan)
R11	2	Tell(5, 1, hasOccupancy(?p, Yes)) ⟶ hasOccupancy(?p, Yes)	GPS(UNMC)
R12	2	hasOccupancy(?p,Yes) ⟶ Tell(1, 6, hasAircon(?p, On))	GPS(UNMC)

Table 2. Preference set transition

System status	COI	_COI	Facts in WM
Initial information	GPS(UNMC)	Patient(Alan)	PatientID(101), hasPatientID(Alan, 101)
Iterations of the system case scenario, where a user moves to different locations at different times with preferences enabled are GPS(UNMC) and Patient(Alan)			
User location	Derived facts	Preference indicator found in WM	Corresponding subset of rules
GPS(Home)	–	No	R7, R8
GPS(UNMC)	–	GPS(UNMC)	R7, R8, R11, R12
GPS(Hospital)	hasLocation(Alan, Hospital) Patient(Alan)	Patient(Alan)	R1, R2, R3, R4, R5, R6, R7, R8, R9, R10

a preference set for him at the hospital, which will illustrate how the sensed/externally received context-based preference as well as derived-context based preference work together to minimise the load on the agent's inference engine by reducing the number of rules while achieving the desired results. The rules in Table 1 are some example rules that are used to design Agent 1. The initial facts provided to the system are *PatiendID(101)* and *hasPatientID(Alan,101)*. The location is detected by the GPS sensor and also added to the agent's working memory as a fact. Once the COI is defined, the system checks and separates the COI from _COI. The _COI is put aside for the later use once the system starts working. As a result, the Table 2 shows us set of rules that are in the preference set for a given set of user provided preferences. In Table 2, we show the transition of facts, Context of Interest (COI) and how the rules are grouped. We assume that the initial location of the user is his Home. Later on, the user visits the

smart hospital and accordingly his location is detected which in turns deduce that the user is a Patient. Accordingly, the derived-context is used as a preferred context that helps generating a new set of rules by replacing the existing rules to be used in the agent's inference engine.

6.2 Rule-Based Preferences

It is always possible that a conflict occurs between the rules, and to resolve it we assign priorities (column m in Table 1) to the rules. The rule priorities give one rule preference over another rule. In this case study, we deliberately made a scenario where according to the facts we can have two different rules generating contradictory outcome as *hasSituation(Alan, Emergency)* and $\sim$*hasSituation(Alan, Emergency)*. Which if not handled can derive unwanted conclusion. Therefore, we assigned the priorities to rules, as a part of defeasible reasoning, and in the scenario described below, the rules R1 and R2 are assigned priority 3, while R5 has priority 1. Since R1 and R2 having higher priority than that of R5, the preference will be given to R1 and R2 over R5. Thus, avoiding any unwanted outcome. A more detailed discussion on defeasible reasoning can be found in [6].

7 Conclusion and Future Work

In this paper, we present derived-context based user preference as a personalisation mechanism into context-aware applications. The proposed approach supports preferences that could be easily controlled to personalise the system behaviour without modifying the internal settings or agent's program. We also present a revised algorithm to identify relevant user preferences. The research on context-aware user preferences, specifically on decision support system still in its early stages, many challenges remain in this area. In the future, we would like to explore the integration of social network based preferences into the system and analyse its effectiveness from different aspects, especially from the resource usage point of view.

References

1. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: Proceedings of the First Workshop on Mobile Computing Systems and Applications, pp. 85–90. IEEE Computer Society, Washington (1994)
2. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**(1), 4–7 (2001)
3. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. Int. J. Ad Hoc Ubiquit. Comput. Arch. **2**(4), 263–277 (2007)
4. Perera, C., Zaslavsky, A.B., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutorials **16**(1), 414–454 (2014)

5. Rakib, A., Haque, H.M.U., Faruqui, R.U.: A temporal description logic for resource-bounded rule-based context-aware agents. In: Vinh, P.C., Alagar, V., Vassev, E., Khare, A. (eds.) ICCASA 2013. LNICST, vol. 128, pp. 3–14. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05939-6_1
6. Rakib, A., Haque, H.M.U.: A logic for context-aware non-monotonic reasoning agents. In: Gelbukh, A., Espinoza, F.C., Galicia-Haro, S.N. (eds.) MICAI 2014. LNCS (LNAI), vol. 8856, pp. 453–471. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13647-9_41
7. Uddin, I., Rakib, A.: A preference-based application framework for resource-bounded context-aware agents. In: Kim, K.J., Joukov, N. (eds.) ICMWT 2017. LNEE, vol. 425, pp. 187–196. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-5281-1_20
8. Lai, J., et al.: Bluespace: personalizing workspace through awareness and adaptability. Int. J. Hum. Comput. Stud. **57**(5), 415–428 (2002)
9. Stefanidis, K., Pitoura, E., Vassiliadis, P.: Modeling and storing context-aware preferences. In: Manolopoulos, Y., Pokorný, J., Sellis, T.K. (eds.) ADBIS 2006. LNCS, vol. 4152, pp. 124–140. Springer, Heidelberg (2006). https://doi.org/10.1007/11827252_12
10. Hong, J., Suh, E.H., Kim, J., Kim, S.: Context-aware system for proactive personalized service based on context history. Exp. Syst. Appl. **36**(4), 7448–7457 (2009)
11. Hoque, M.R., Kabir, M.H., Seo, H., Yang, S.-H.: PARE: profile-applied reasoning engine for context-aware system. Int. J. Distrib. Sens. Netw. **12**(7) (2016)
12. Alhamid, M.F., Rawashdeh, M., Dong, H., Hossain, M.A., Saddik, A.E.: Exploring latent preferences for context-aware personalized recommendation systems. IEEE Trans. Hum. Mach. Syst. **46**(4), 615–623 (2016)
13. Kyriazakos, S., et al.: eWALL: an intelligent caring home environment offering personalized context-aware applications based on advanced sensing. Wirel. Pers. Commun. **87**(3), 1093–1111 (2016)
14. Fong, J., Lam, H.-P., Robinson, R., Indulska, J.: Defeasible preferences for intelligible pervasive applications to enhance eldercare. In: 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 572–577. IEEE (2012)
15. Thanakodi, S., Nazar, N.S.M., Tzen, B.S.P., Roslan, M.M.M.: Survivor tracking system based on heart beats. In: Kim, K.J., Joukov, N. (eds.) ICMWT 2017. LNEE, vol. 425, pp. 550–557. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-5281-1_61
16. Uddin, I., Rakib, A., Haque, H.M.U.: A framework for implementing formally verified resource-bounded smart space systems. Mobile Networks and Applications **22**(2), 289–304 (2017)
17. Uddin, I., Rakib, A., Haque, H.M.U., Vinh, P.C.: Modeling and reasoning about preference-based context-aware agents over heterogeneous knowledge sources. Mob. Netw. Appl. (2017). https://doi.org/10.1007/s11036-017-0899-5