Proposal of a Context-Aware Smart Home Ecosystem

Radosław Klimek(✉) and Grzegorz Rogus

AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
{rklimek,rogus}@agh.edu.pl

Abstract. Smart homes are more and more popular since they are regarded as synonyms of an ideal and friendly environment. Novel aspects like context awareness and pro-activity are fundamental requirements for development of such systems. Smart scenarios enable switching devices in a desired way when a certain user activity considered as a trigger involves a system activity. Some smart scenarios are proposed. Context modeling and reasoning for smart homes is widely considered. A software architecture including web services is introduced and discussed.

Keywords: Smart home · Ecosystem · Context-awareness · Smart scenario · Context modeling · Context reasoning · Architecture for smart homes · Web service · SOA · Pervasive computing

1 Introduction

A smart home is a house that uses information technology to monitor the environment, control the electric appliances and communicate with the outer world. Smart homes sometimes mean automated homes. Automation systems have been developed to automatically achieve some activities performed frequently in daily life, to obtain a more comfortable and easier living environment. There are many different systems classified as smart home systems. On one hand, a simple system using only the remote controls of electrical devices such as lighting, heating, air conditioning, audio and video entertainment systems and security is considered as a synonym of the smart home. On the other hand, contrary to the above approach, there are complex systems which are able to make their own decisions basing on the expected behavior of users. Some use cases are as follows: "Children usually go to sleep at 8.00 PM; The system automatically turns on air condition at 7.30 PM to decrease the temperature in the children's room (if the temperature is too high), when children leave the living room and enter bedrooms, then lights in bedrooms are still on, and the brightness is decreased, the TV set is muted. How much the TV volume is lowered also depends on the individual preferences of the viewer. Information about the preferences of users were collected automatically by the system basing on historical data".

Such systems can be classified as context-aware systems. One of the main goal of context-aware applications is to make inhabitants feel more comfortable and safe. To achieve this goal, ubiquitous computing is brought into our living space. By collecting sensor data and extracting specific information from datasets we can develop many kinds of context-aware applications which would improve our quality of life.

In this paper we propose some scenarios as well as a novel architecture to create an intelligent, automated home and a context-aware smart home ecosystem to control and manage all sensors and devices used in such scenarios. These aspects constitute the main findings of the paper. The proposed system covers the following features. The dynamic nature of a context-aware application requires a system allowing on-line reconfigurations. Adding or removing sensors or devices, as well defining new features, requires applications that dynamically (reactively and pro-actively) adapt their behaviors. Scenarios are considered as skeleton scenarios which are context-sensitive and filled with actions when necessary.

2 Related Works

Research in context-aware computing has recognized the need for building infrastructures to support context-awareness activities [12,26]. The authors of works [1,24,11] have conducted surveys in the context-awareness domain, whereas the work documented in [22,19] describes middleware approaches for this area of research.

Paper [13] presents a survey distinguishing different solutions for context aware engineering proposed by different researchers and developers. The approaches can be divided into the following categories: middleware solutions and dedicated service platforms; use of ontologies; rule-based reasoning; source code level programming/language extensions; model-driven approaches; essage interception. There are, however, generic cases where several paradigms or patterns are used together in the same approach. However, the above approaches are not completely disjoint and a potential developer may opt for a combination of several techniques.

Although there are a lot of frameworks and middlewares developed for context-aware systems, they are usually limited to a specific domain and designed. Examples include CoBrA [5] and SOUPA [6] for building smart meeting rooms, GAIA [9] for active spaces; ezContext [18] is a framework that provides automatic context life cycle management. The authors of [20] present a framework bridging the communication between heterogeneous devices, whereas [2] presents a system capable for integrating heterogeneous devices dynamically enabling interoperability between those.

Paper [21] presents a survey distinguishing different architectural styles used in context-aware systems. These are as follows:

1. Component-based architecture, where the entire solution is based on loosely coupled major components, interacting with each other. For example, the Context Toolkit [10] builds a framework for interfacing with devices and software entities which provide contextual information. There are three major components which perform the most critical functionalities of the system.

A set of abstractions, namely, context widgets, context interpreters and context aggregators can be used by application developers to prototype new context-aware applications. Widgets abstract sensor-originated information and make that information available to applications.
2. Distributed architecture enables peer-to-peer interaction in a distributed fashion, such as in Solar [4].
3. Service-based architecture, where the entire solution consists of several services working together.
4. Node-based architecture allows deployment of pieces of software with similar or different capabilities; these communicate and collectively process data in sensor networks [23].
5. Centralized architecture which acts as a complete stack (e.g. middleware) and provides applications to be developed on the top of that, but provides no communication between different instances of the solution.
6. Client-server architecture separates sensing and processing, as shown in CaSP [7].
7. Agent-oriented paradigm [3] with agent-oriented computing, autonomous entities in the system which individually manage specific tasks and cooperate among themselves with standardized protocols to achieve a greater goal. An example of using that style is ACAI [14].

3 Smart Scenarios

Scenarios for smart homes are descriptions of all relevant elements including aims and context to provide context-aware automation. Scenarios are focused on functionalities and resources and how they can improve the quality of life. Smart functionalities help in easing everyday life tasks. Both normal and abnormal behaviors for smart homes are identified and considered. Smart behaviors are usually specified using use cases and their scenarios. Applying these tools is ubiquitous in software engineering, since they relatively precisely describe the desired interactions. Recognizing interactions and services enables reasoning to provide smart reactions based on context awareness.

The scenario expressed in Tab. 1 for a use case is a scenario for normal behavior. It might also refer to some social networks, for example Facebook, Instagram, or others, where some data identifying the inhabitants might be stored. That data, e.g. photos, might be used in a smart home system.

Some abnormal behaviors for smart homes are presented in work [25]. Let us consider some of them in a more general form and more formally. The scenario expressed in Tab. 3 deals with everyday activity which is taking a shower or bath. The cloud, rather than a local database, allows to analyze profile/preferences no matter where a person takes a shower.

Some other use cases and their scenarios proposed here refer to a security system enabling to detect intrusion in a smart home. The system is switched on manually using a wall-mounted panel. The night alarm is an option of the security system and enables activating the system in the entire zone, excluding

Table 1. The use case scenario for a guest recognition

UC name: “Guest recognition when the door bell rings”
Precondition: Access to the local database and social networks
Scenario: 1. A person appears in the front door and presses the entrance button; 2. The guest is scanned by the entrance camera; 3. The identification data is compared with local data base; 4. If there is a negative result, the identification data is compared with data and profiles available in social networks searching their family and friends; 5. Depending on the search result, the sound of a door bell is divided into three categories: (a) family, (b) friends, (c) others.
Postcondition: The bell at the front door rings.

Table 2. Use case scenario for the night option of the security system (turn on) (top). Use case scenario for the temporary turn off the security system (middle). Use case scenario for the night option of the security system (turn off) (bottom)

UC name: “The security system automatically turns on for a night”
Precondition: Everything works normal
Scenario: 1. If it is night time (say from 11 p.m. to 6 a.m.) then skip to the next step else finish scenario; 2. If residents are upstairs and their activity has stopped for a half hour, then turn on automatically the night option for the system.
Postcondition: The night option for the security system is on, if necessary
UC name: “The security system is (temporarily) switched off”
Precondition: The night option for the security system is on.
Scenario: 1. If it is night (say, from 11 p.m. to 6 a.m.) then skip to the next step, else finish scenario; 2. If a resident begins to go downstairs, then the security system is turned off automatically.
Postcondition: The night option for the security system is off.
UC name: “The security system is automatically switched off after the night”
Precondition: The night option for the security system is on.
Scenario: 1. If the night (say, from 11 p.m. to 6 a.m.) is over, then skip to the next step, else finish scenario; 2. If any resident begins to go downstairs, then turn off the security system.
Postcondition: The security system is off, if necessary

Table 3. The use case scenario for a too long showering

UC name: "Too long shower or bath"
Precondition: Access to a local/cloud database
Scenario: 1. A person/resident starts taking a shower; 2. The person is scanned and identified by the system; 3. The shower/bath timer starts; 4. The profile in a local/cloud database is searched for the longest shower/bath time of a given person, taking into account different circumstances, e.g. summer, winter, morning, evening, etc. 5. If the shower/bath time exceeds the profiled value, then other family members are notified.
Postcondition: Generating a warning to other residents, if necessary.

bedrooms upstairs. On the other hand, the night option could be activated by a system itself, i.e. automatically, in other words without direct action of inhabitants, when some general circumstances are satisfied, see Tab. 2, as well as other scenarios.

4 Understanding of Context

Context awareness is a kind of intelligent computing behavior. For computing systems, context awareness is the capability to provide relevant services and information to the users based on their situational conditions. Formally, according to [8], "A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."

In this section, we discuss the basic concepts underlying the fundamentals of context-aware systems. There are many definitions of *context*. One of the most popular is a definition presented in [8]: "Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." According to that definition, context is a result of data interpretation. The source of data in context-aware systems are sensor networks.

Context is represented in context models. Those are data structures, which are populated by abstracting and representing contextual information for further processing. Context can be represented in various formats ranging from simple key-value models to graphical representations [24]. Out of those, current research indicates that ontologies are the most expressive context representation models [11,24]. Ontologies provide a powerful paradigm for context modeling, which offers rich expressiveness and supports the dynamic aspects of context awareness. Ontologies represent concepts and relationships between them. An ontology is used to define different entities like objects, services, sensor data and relations between concept and situation. With the existing ontology from different sources, the context models can be used for modeling user behaviors and surrounding situations. These will be useful for a smart home system to make suitable choices for users. Ontology reasoning is needed to guide the smart home system to provide suitable services to the users. It uses the environment entities and information.

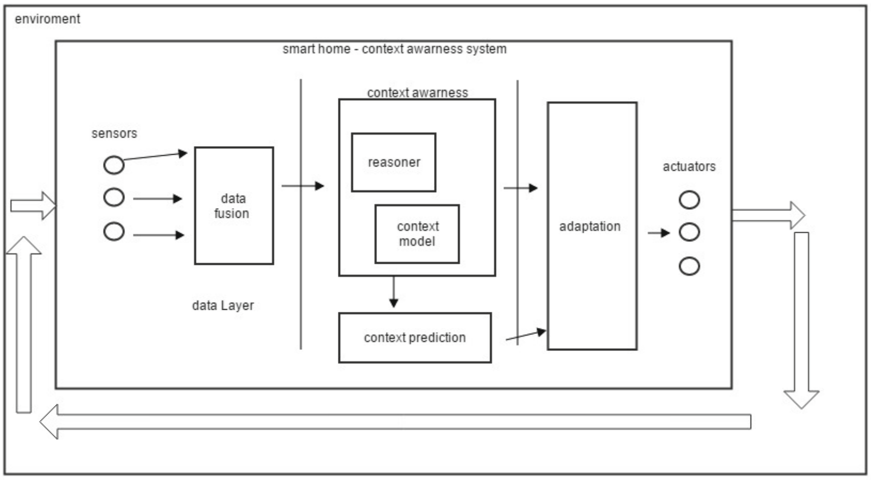

Fig. 1. Context processing in a smart home

Figure 1 provides an overview of how the context is processed and how the smart home system actions emerge from context processing efforts. On Figure 1 the context processing is viewed from the aspects of algorithms and information flows. For simplicity the aspects like hardware, physical communications, interaction protocols are intentionally left out in Figure 1.

Context-aware applications differ from traditional applications since they use sensed information to adapt the service provisioning to the current context of the user. In order to achieve that, context-aware applications, in general, should be capable of:

- sensing context from the environment;
- observing, collecting and composing context information from various sensors;
- autonomously detecting relevant changes in the context;
- reacting to these changes, by either adapting their behaviour or by invoking appropriate services;
- interoperating with output service providers.

The data gathered by sensors needs to be evaluated and processed in order to provide useful information. Raw context data can be collected either from a single sensor node, or by aggregating context data available from multiple sensor nodes. The collected context data need to be stored for further processing. The raw context tends to be noisy and inconsistent, which calls for proper context pre-processing, inconsistency detection and resolution mechanisms. After processing, context is represented using particular pattern or design descriptions, called context models. In addition, if the data is organized as low level information toward concepts, then it may not be usable when domains change. Thus, organizing the data into a higher level concept it is necessary to give users understandable definitions, as well as building up knowledge concerning general rules that will be used in providing services. Such higher level concept will also be useful in transferring experiences to different domains. That is why data collected directly from sensor have to computed. That secondary context can be computed by using sensor data fusion or data retrieval operations such as web service calls. Figure 2 presents approach using data layer to represent raw sensor

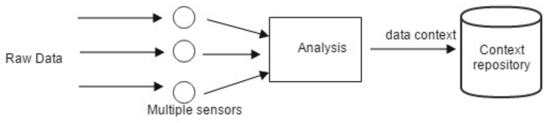

Fig. 2. From raw data to context information – data layer

readings in a format suitable for data classification.

Sensors collect data, and many sets of data are used to construct the abstract concept to describe the environment, as well as the attribute of the users. For situation, the semantics of high-level behaviours and services are processed together to provide extra information that will benefit the system to make decisions. During such processes, the output of sensors is categorized and matched with the ontology built for application domain.

External devices generate very large amounts of data. Often, these are data that can not be directly used in the model context. An example of such data are data generated by the camera in in scenario expressed in Tab. 1. The data streaming transmitted by a camera from a point of view of the context model does not seem interesting. Interesting is the information about the user id identified by image analysis. We need data that uniquely identifies a person calling to the door. The streaming data from camera are compared to digital images from a local database or in a case then we obtain the negative comparison result, the service is called to link and search images from Facebook profile. That external service identify the person by searching a list of friends' profiles. The result of the activities are context data that classify person to one of the group: family, friend, alien.

5 Proposal of a System Architecture

In this section we present our concept of the overall architecture for context-aware applications in smart homes. The proposed system is based on a service-oriented architecture with active use of certain architectural patterns. Based on [18], we the adopted the following design principles related to context-aware management frameworks:

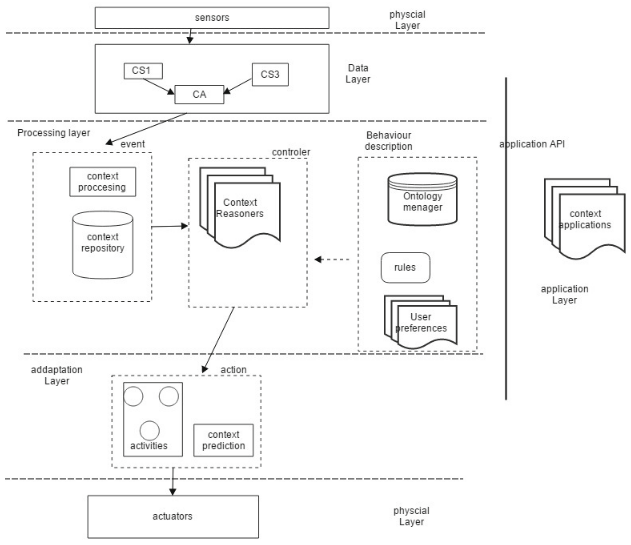

Fig. 3. Overview architecture for smart homes

– the functionalities need to be divided into layers and components in a meaningful manner,
– multi-model reasoning,
– monitoring and event detection: Events play a significant role in smart home systems, which is complemented by monitoring. Detecting an event triggers an action autonomously.

The system architecture, see Figure 3, is divided into the physical layer, the data layer, the processing layer, the adaptation layer and the application layer. They are responsible for sensor operation, data processing, behavior recognition, evolution of the evaluation process and interaction with the user's work. The application API provides access to the context-awareness system for user applications (for example mobile). It represents the architectural cut between context management and context utilization. The application API has direct access to the context repository, ontology model, user preferences and set of defined rules. It reads the current context and commits user updates into the context. The inference engine may also notify user applications through the application API, if that is required to infer new context. Processing of context information is a challenging task. Deducing rich information from basic sensor samples may require complex computation. The system should provide mechanisms to distribute context processing activities among multiple components. That way we propose to use the context sources and the managers hierarchy architectural pattern aimed at providing a structural schema to enable the distribution and composition of context information processing components.

This approach, see Figure 4, enables encapsulation, is more efficient and enables flexible and decoupled distribution of context processing activities (sensing,

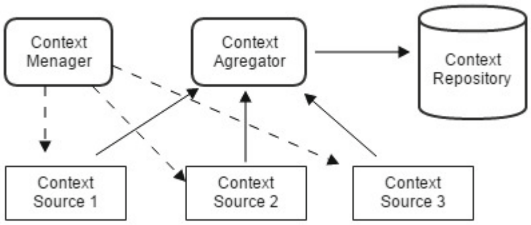

Fig. 4. Decomposition of the Data Layer

aggregating, inferring and predicting). We define three types of context processor components, namely context source, context manager and context aggregation. Context source components encapsulate single domain sensors, such as movement or time measuring devices. Aggregation manager components cover multiple domain context sources, such as the integration of context data from context sources. Context manager is used to manage a set of context sources dynamically by creating or destroying some of them.

The second architecture pattern used in our approach is the schema of Event-Condition-Action (ECA). Figure 5 provides an overview of how the context is processed and how the pervasive computing system actions emerge from context processing efforts. The entire system consists of three components.

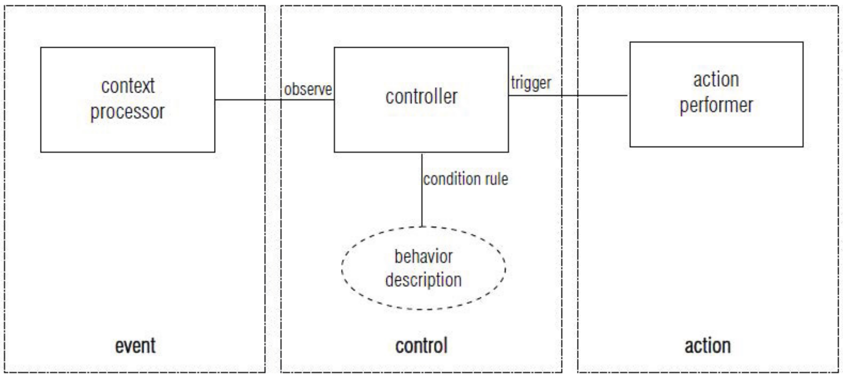

Fig. 5. Processing and matching schema: Event – Control – Action

1. Context processor gathers information about all events that occur in the smart home.
2. Controller analyses incoming events regarding some patterns of behavior descriptions, i.e. it matches incoming events to patterns stored as a behavior description.
3. Action performer produces smart actions switching devices in a desired way.

By using pattern decomposition and ECA, we model advanced context processing as it is seen in Figure 7. For example, in scenario "Security System",

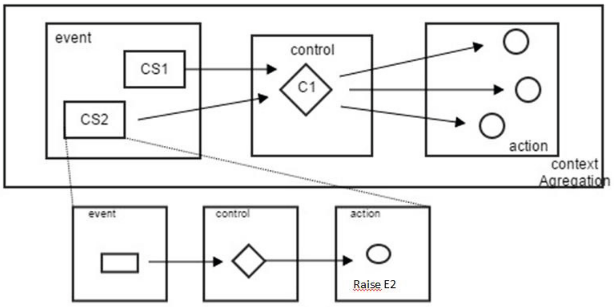

Fig. 6. Sample processing of events for the ECA schema

see Tab. 2, the event (*IsNight* and *ResidentgoDown*) is a compound event observed on the following components: (i) a context source component (CS1 in Figure 6) that detects an *Night* event (E1) and (ii) a context source component (CS2 in Figure 6) that detects when a resident go down on stairs (E2). Within the security system detector context Agregator (CA), the following condition rule2 is described in controller C1, characterizing the recursive nature of the event-control-action pattern. As we said before, one of the important foundations for modern context awareness is parallelization of reasoning. There are many independent events which smart home systems have to analyze. We propose some extend pattern ECA by using the publish/subscribe pattern to define the connection between controller and set of event, see Figure 7. The historical behaviors might be encoded into logical specifications, and can be later analyzed for satisfiability, c.f. works [16,15,17], supporting the current reasoning process and behavior recognition.

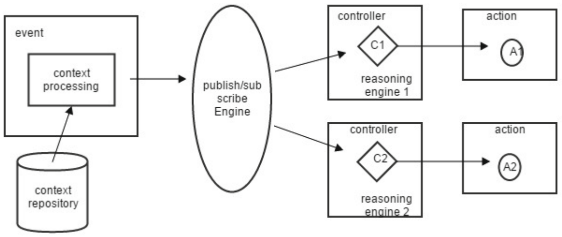

Fig. 7. Distributed Context Reasoning

6 Conclusions

In this paper we propose some solutions that refer a novel software architecture smart homes. The findings include a new approach for context modeling and reasoning for smart homes. This work opens some research areas which are of crucial importance for the idea of smart homes.

Future works may include the context-based reasoning methods and tools. It should result in a CASE software involved in creating industrial-proof tools.

References

1. Anagnostopoulos, C.B., Tsounis, A., Hadjiefthymiades, S.: Context awareness in mobile computing environments. Wireless Personal Communications 42(3), 445–464 (2007), http://dx.doi.org/10.1007/s11277-006-9187-6
2. Bartelt, C., Fischer, T., Niebuhr, D., Rausch, A., Seidl, F., Trapp, M.: Dynamic integration of heterogeneous mobile devices. ACM SIGSOFT Software Engineering Notes 30(4), 1–7 (2005), http://dblp.uni-trier.de/db/journals/sigsoft/sigsoft30.html#BarteltFNRST05
3. Bradshaw, J.M.: Software Agents. MIT Press, Cambridge (1997)
4. Chen, G., Li, M., Kotz, D.: Data-centric middleware for context-aware pervasive computing. Pervasive Mob. Comput. 4(2), 216–253 (2008)
5. Chen, H., Finin, T., Joshi, A.: Semantic web in the context broker architecture. In: Proceedings of Percom 2004, pp. 277–286 (2004)
6. Chen, H., Perich, F., Finin, T.W., Joshi, A.: Soupa: Standard ontology for ubiquitous and pervasive applications. In: MobiQuitous, pp. 258–267. IEEE Computer Society (2004)
7. Devaraju, A., Hoh, S., Hartley, M.: A context gathering framework for context-aware mobile solutions. In: Chong, P.H.J., Cheok, A.D. (eds.) Mobility Conference, pp. 39–46 (2007)
8. Dey, A.K., Abowd, G.D.: Towards a better understanding of context and context-awareness. In: Workshop on The What, Who, Where, When, and How of Context-Awareness (CHI 2000) (April 2000), http://www.cc.gatech.edu/fce/contexttoolkit/
9. Dey, A.K., Abowd, G.D., Salber, D.: A context-based infrastructure for smart environments, tech. report of georgia institute of technology (1999), http://www.cc.gatech.edu/fce/contexttoolkit/pubs/MANSE99.pdf
10. Dey, A.K.: Providing Architectural Support for Building Context-aware Applications. Ph.D. thesis, Atlanta, GA, USA (2000)
11. Gellersen, H.W., Schmidt, A., Beigl, M.: Multi-sensor context-awareness in mobile devices and smart artifacts. Mob. Netw. Appl. 7(5), 341–351 (2002)
12. Hong, J.I., Landay, J.A.: An infrastructure approach to context-aware computing. Human-Computer Interaction 16(2-4), 287–303 (2001), http://dblp.uni-trier.de/db/journals/hhci/hhci16.html#HongL01
13. Kapitsaki, G.M., Prezerakos, G.N., Tselikas, N.D., Venieris, I.S.: Context-aware service engineering: A survey. J. Syst. Softw. 82(8), 1285–1297 (2009), http://dx.doi.org/10.1016/j.jss.2009.02.026
14. Khedr, M., Karmouch, A.: Acai: agent-based context-aware infrastructure for spontaneous applications. J. Network and Computer Applications 28(1), 19–44 (2005)
15. Klimek, R.: Preference models and their elicitation and analysis for context-aware applications. In: Gruca, A., Czachórski, T., Kozielski, S. (eds.) Man-Machine Interactions 3. AISC, vol. 242, pp. 353–360. Springer, Heidelberg (2014)
16. Klimek, R.: A system for deduction-based formal verification of workflow-oriented software models. International Journal of Applied Mathematics and Computer Science 24(4), 941–956 (2014), http://www.amcs.uz.zgora.pl/?action=paper\&paper=802

17. Klimek, R., Faber, Ł., Kisiel-Dorohinicki, M.: Verifying data integration agents with deduction-based models. In: Proceedings of Federated Conference on Computer Science and Information Systems (FedCSIS 2013), Kraków, Poland, September 8-11, pp. 1049–1055. IEEE Xplore Digital Library (2013)
18. Martin, D., Lamsfus, C., Alzua, A.: Automatic context data life cycle management framework. In: 2010 5th International Conference on Pervasive Computing and Applications (ICPCA), pp. 330–335 (2010)
19. Modahl, M., Bagrak, I., Wolenetz, M., Hutto, P., Ramachandran, U.: Mediabroker: An architecture for pervasive computing. In: Proc. of the 2nd IEEE International Conference on Pervasive Computing and Communications, pp. 253–262 (2004)
20. Nakazawa, J., Tokuda, H., Edwards, W.K., Ramachandran, U.: A bridging framework for universal interoperability in pervasive systems. In: ICDCS. IEEE Computer Society (2006)
21. Perera, C., Zaslavsky, A.B., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: A survey. IEEE Communications Surveys and Tutorials, 414–454 (2014)
22. da Rocha, R.C.A., Endler, M.: Evolutionary and efficient context management in heterogeneous environments. In: Proceedings of the 3rd International Workshop on Middleware for Pervasive and Ad-hoc Computing, MPAC 2005, pp. 1–7. ACM (2005)
23. Sanchez, L., Lanza, J., Olsen, R., Bauer, M., Girod-Genet, M.: A generic context management framework for personal networking environments. In: 2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking Services, pp. 1–8 (July 2006)
24. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management, pp. 31–41 (2004)
25. Tran, A.C., Marsland, S., Dietrich, J., Guesgen, H.W., Lyons, P.: Use cases for abnormal behaviour detection in smart homes. In: Lee, Y., et al. (eds.) ICOST 2010. LNCS, vol. 6159, pp. 144–151. Springer, Heidelberg (2010)
26. Yau, S.S., Karim, F., Wang, Y., Wang, B., Gupta, S.K.S.: Reconfigurable context-sensitive middleware for pervasive computing. IEEE Pervasive Computing 1(3), 33–40 (2002), http://dx.doi.org/10.1109/MPRV.2002.1037720