Architectural Elements of Ubiquitous Systems: A Systematic Review

Carlos Machado
Informatics Department
UFPB
João Pessoa, Brazil
carlos@ccen.ufpb.br

Eduardo Silva, Thais Batista, Jair Leite
Computer Science Department
UFRN
Natal, Brazil
eduardoafs@ppgsc.ufrn.br,
{thais,jair}@ufrnet.br

Elisa Yumi Nakagawa
Department of Computer Systems
USP
São Carlos, Brazil
elisa@icmc.usp.br

***Abstract*—Ubiquitous systems have become an important and even essential part of our daily life. For instance, smart homes are good examples where such systems can be found. However, the design and implementation of ubiquitous systems are hard tasks, as they involve several areas of computing, as software engineering, artificial intelligence, and distributed systems. This task is even harder as there is no general reference architecture that could be used to guide the development of such systems. As a consequence, each project solves the same problem in a different way, some better than others. This paper aims at exploring, organizing, and summarizing the common, essential architectural elements of those systems. We have also investigated reference architectures for this type of systems, as these architectures are important artifacts for providing such elements. For this, we conducted a systematic review that is a technique that provides an overview of a research area to assess the amount of existing evidences on a topic of interest. As main results achieved, we have found a set of eleven elements, which appears in most of the existing systems and middlewares that can be used to define a general-use software architecture. This work could certainly contribute to a more systematized development of ubiquitous systems.**

Keywords-ubiquitous computing; systematic review; software architecture

I. INTRODUCTION

Ubiquitous computing is the term initially coined by Mark Weiser [1] when referring to computer systems available everywhere at any time. These systems are often present in our lives, in form of smart TVs, smart cars, and even whole smart homes. They are capable of automating many usual tasks and support our daily live, using concepts of artificial intelligence and distributed systems.

Lyytinen and Yoo [2] proposed a difference between ubiquitous computing and pervasive computing by defining pervasive computing as models with high coupling and low mobility, while ubiquitous computing are computing models with high coupling and high mobility. However, this distinction was not widely accepted in the literature and some works do not make distinction between these two terms. It is important to highlight these differences, since some advances in ubiquitous systems could not be applied in pervasive computing, and vice versa.

An essential part of a ubiquitous project, as in any software system, is the software architecture. This architecture encompasses a set of decisions about the software organization as its structure, interfaces, behavior, and definitions of the structural elements [3]. A software architecture is essential to guide the development of a robust system, which can evolve and change through its lifetime. To help the definition of such artifact, the concept of reference architecture was proposed. A reference architecture is a special type of software architecture that provides a common understanding of a given domain, in the case of this work, the ubiquitous systems domain [4][5].

Although a number of ubiquitous systems have been proposed and impacted several sectors of the society, there is no consensus on what are the common, essential elements of a ubiquitous systems' architecture. The understanding of what are these elements is crucial for the systematic development of new systems, as well as to the maintenance and quality of existing ones.

In this context, this paper aims to identify the main elements that constitute the architecture of ubiquitous systems and whether there is any reference architecture for this domain. To achieve this goal, we conducted a systematic review that is a technique originated from the Evidence-Based Software Engineering (EBSE) [6,7], which allows to explore, organize, synthetize, and evaluate all the contributions of a research area. A systematic review allows us to identify a variety of studies that may involve theories and concepts, technological development reports, experimental research results and many others. As main results, we have observed eleven common elements, which are present in most of existing systems and middleware, and that we identified as essential elements. These elements can be used to define a general-use reference architecture, aggregating common solutions for common problems in the ubiquitous systems development.

This paper is organized as follows: Section II presents the systematic review, from its planning to the analysis of results, focusing on the architectural elements that characterize systems for ubiquitous computing. Section III contains a discussion of the collected data. Section IV presents the threats to validity of this systematic review. Finally, Section V presents final remarks and future work.

II. Systematic Review

This systematic review was conducted in the context of software architectures for ubiquitous computing, aiming at evaluating relevant studies until March 2013. To conduct this systematic review, the process was divided into three steps, as illustrated in Figure 1: Planning, Execution, and Evaluation. In the first step, we defined the search criteria and the inclusion and exclusion criteria that were used to collect related works for ubiquitous or pervasive computing. This step was also responsible for defining what we expect to extract from the found studies. The second step consisted in the execution of the systematic review, in which was performed the search for the primary studies (i.e., conference publications, periodicals, thesis, etc.), using the planning from the first step. The second step also applied the inclusion and exclusion criteria, in order to filter the results that were relevant to this review. Finally, in the third step, the results were evaluated to extract data to formulate the answer for the research questions.

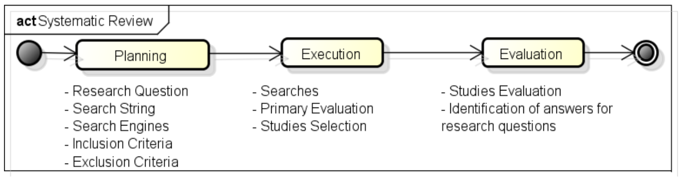


Figure 1: Systematic Review Steps

A. *Planning*

This step of the systematic review defines: (i) research questions, (ii) search strategies and (iii) inclusion and exclusion criteria.

1) *Research Questions*

In order to identify the primary studies that present common, essential architectural elements for ubiquitous systems, the following Research Questions (RQ) were defined:

- RQ1: Which are the reference architectures for ubiquitous systems? Note: This question was formulated in order to find reference architectures for ubiquitous systems. These architectures could provide common, essential elements of ubiquitous systems.
- RQ2: What are the common architectural elements for ubiquitous systems? Note: This question was defined as a complement for RQ1, and also intends to identify the common elements for ubiquitous systems.

2) *Search Strategy*

To establish the search strategy for the primary studies, from RQ1 and RQ2, the following keywords were chosen: “Reference Architecture” and “Ubiquitous Computing”. We also identified synonyms for these keywords, or similar contexts: “Reference Architecture” may be referred as “Reference Model” and it is directly related to “Software Architecture” or “Architectural Model”. In addition, “Ubiquitous Computing” is related to “Pervasive Computing”, as we explained in Section 1. Middleware for ubiquitous computing were also considered, through the keywords “ubiquitous middleware architecture” and “pervasive middleware architecture”. This inclusion had two goals: (i) to obtain an overview of existing systems, since middleware are designed to meet a wide variety of ubiquitous/pervasive applications, and (ii) the identification of the elements of these middlewares that consist in important components for ubiquitous systems. Thus, it was established the following search string: (("Reference Architecture" OR "Reference Model" OR "Software Architecture" OR "Architecture Model") AND ("Ubiquitous Computing" OR "Pervasive Computing" OR "ubiquitous middleware" OR “pervasive middleware”)). This string was used in the following publications databases: *IEEEXplorer*, *ACM Digital Library*, *Web of Knowledge* and *ScienceDirect*. The search string was adapted for each database in order to perform a directed search on title, abstract, and keywords. Only publications in English were considered.

The review process was designed as follows: The search must be performed in digital libraries, which include the main vehicles where the literature can be published. After that, the reviewers may read the title, abstract, and keywords of the found studies, in order to define which studies are worth reading the full text. After reading them, the answers of the research questions might be formulated.

3) *Inclusion and Exclusion Criteria*

To evaluate and select relevant studies, we defined a set of inclusion and exclusion criteria. These criteria were applied after each search, to define the relevance of a given study. The Inclusion Criteria (CI) was used to include relevant studies in this systematic review, namely:

- IC1: The study proposes, uses or evaluates a reference architecture for ubiquitous systems; and
- IC2: The study presents a middleware for ubiquitous computing, explicitly exhibiting its architecture.

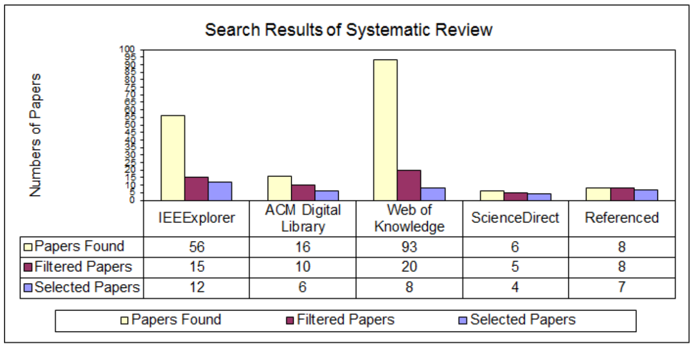
The Exclusion criteria (EC) were defined to exclude studies with no relevance for this review, i.e., studies that do not contribute to answer RQ1 or RC2. The ECs are:

- EC1: The study is not related to ubiquitous or pervasive systems;
- EC2: The study is not in English;
- EC3: The study does not have abstract or the full text is not available;
- EC4: The study consists of a compilation of studies from conferences or workshops, for example; and
- EC5: The study defines a low-level architecture, describing hardware or operational elements.

It is worth saying that a relevant study to this systematic review is defined as a study that does not satisfy any of the exclusion criteria, satisfying at least one of the inclusion criteria.

B. *Execution Results*

Upon concluding the searches, we obtained the results summarized in Figure 2. This figure shows the number of papers found by the searching process and the selected papers. In the figure, the found papers represent the number of papers returned by the automatic searching process and evaluated, i.e., we read their titles, keywords, and abstract. The selected papers represent papers whose abstracts and keywords evidenced that they are interesting for our systematic review and they were selected to be fully read.



	IEEExplorer	ACM Digital Library	Web of Knowledge	ScienceDirect	Referenced
□ Papers Found	56	16	93	6	8
■ Filtered Papers	15	10	20	5	8
■ Selected Papers	12	6	8	4	7

Figure 2: Search Results

As illustrated in Figure 2: (i) from 56 results found by the *IEEExplorer* search engine, 15 were filtered and 12 were selected for the second stage; (ii) from 16 results found by the *ACM Digital Library* engine, 10 were filtered and six were selected for the second stage; (iii) from 93 results found by the *Web of Knowledge* search engine, 20 were filtered and eight were selected for the second stage; (iv) from six results found by the *ScienceDirect* search engine, five were filtered and four were selected. Additionally, eight new studies were found from the evaluation of references of the selected articles in the first instance, and seven of them were selected. The total number of selected papers is 37. After a full analysis of each work and the application of the inclusion and exclusion criteria, 13 studies were considered relevant for our study, as listed in Table I.

Among these studies, we highlight the E6, E8, and E11 studies that present surveys on middleware for ubiquitous computing and cite, among others, precursor architectures, such as Gaia [17] and Homeros [13]. However, because these surveys have different goals we used them only as a source for searching new middlewares. Besides that, E10 presents a systematic review about ubiquitous computing, but it focuses on the characterization of ubiquitous computing projects. Note that this study is also interesting for our systematic review; however, it differs from ours, because we aim to identify the architectural elements commonly found in ubiquitous systems, as well as existing reference architectures.

TABLE I: SELECTED PAPERS LIST

Study	Author	Year
E1	*Jiehan Zhou et al* [9]	2009
E2	*Yi Liu, Frenq Li* [10]	2006
E3	*Tao Xu, Bertrand David, René Chalon, Yun Zhou* [11]	2011
E4	*Shriram. R , Vijayan Sugumaran* [12]	2007
E5	*Seung Wok Han, Yeo Bong Yoon and Hee Yong Youn* [13]	2004
E6	*Saeed, A. and Waheed, T.* [14]	2010
E7	*Chang-Woo Song et al* [15]	2013
E8	*Eugster, Patrick Th.; Garbinato, Benoît; Holzer, Adrian* [16]	2009
E9	*Román, M. et al* [17]	2002
E10	*Spínola, R. and Travassos, G* [8]	2012
E11	*Raychoudhury, V., Cão, J., Kumar, M., Zhaung, D.* [18]	2013
E12	*DA, K., Dalmau, M., Roose, P.* [19]	2012
E13	*Fernandez-Montes, A., Ortega, J. A., Alvarez, J.A* [20]	2009

C. Evaluation Results

We found four studies (E1, E2, E11 and E13) that present reference architectures for ubiquitous or pervasive systems: [9], [10] [18], and [20]. The architecture proposed by Zhou [9] is focused on service composition in pervasive systems, while the architecture presented by Liu [10] was defined in a more generic way. Although the authors state that the work is about pervasive computing, the architecture of Liu [10] introduces an element of mobility, which is a typical feature of ubiquitous systems. The architecture proposed by Raychoudhury [18] was defined to support comparisons between existing pervasive systems. Thus, it does not support mobility, and it describes a multi-level structure, which blends elements of high level of abstraction, as reasoners, with elements of low abstraction, such as network protocols. Finally, the architecture proposed by Fernandez-Montes [20] is focused on building applications for smart environments, focusing on requirements for architectural elements.

These works contributed to answer RQ1 about reference architectures for ubiquitous systems. Using these four architectures and other studies on middleware for ubiquitous computing (i.e., studies E3, E4, E5, E7, E8, E9, and E12), it is possible to identify common elements that are essential for ubiquitous systems architectures, in order to find answers to RQ2. Table II describes the elements identified in the evaluated architectures.

TABLE II: COMMON ELEMENTS OF UBIQUITOUS SYSTEMS

Element	Description	Studies
Sensor	Hardware element responsible for providing context information.	E1, E3, E7, E8, E9, E11, E12
Actuator	Hardware element responsible for changing the environment, giving feedback to the user.	E3, E8
Context Service	Service used to recover context information from sensors. It may aggregate many sensors.	E1, E3, E4, E7, E8, E9, E11, E12, E13
Actuation Service	Service used to give feedback to the user. It may aggregate many actuators	E3, E8, E13
Context Repository	Data repository for context information and quality parameters	E1, E2, E3, E4, E5, E7, E9, E11, E12, E13
Event Module	Module to support asynchronous monitoring	E1, E5, E7, E9, E11, E13
Reasoning Module	Module that allow the production of new context information from existing data	E1, E2, E3, E7, E8, E9, E11, E12, E13
Adaptation Module	Module responsible for changing the system behavior according to a preset of rules.	E1, E5, E9, E11, E12, E13
Coupling and Mobility Mechanism	Mechanism that abstracts the notion of environment, making the system functional in various different environments. It uses tracking mechanisms, service search and mobile communications	E2, E4
Aggregation or Composition Module	Module for composing/aggregating context information from lower level information.	E2, E3, E7, E8, E9, E11, E12, E13
Security Module	Module responsible for implementing protection rules, such as authentication mechanisms, access restrictions and service validation.	E2, E5, E9, E11, E13

In Table II, the first column names the element, the second column contains a brief description of the element, and the third column lists the primary studies that present a concept similar or equal to the element in question. Therefore, it can be stated that for the development of ubiquitous systems, this set of eleven elements may be included, since they are commonly found in those systems. Moreover, we can conclude that they are essential elements in ubiquitous systems architectures.

III. DISCUSSION

In the context of ubiquitous systems, a related work presented a systematic review that characterized software projects for ubiquitous systems and intended to understand how this type of systems affects the life cycle of software development [8]. This study also identified a list of 10 main characteristics of ubiquitous systems, as presented in Table III. In this table, we also observe that the set of the common architectural elements found by our systematic review is able to meet the main characteristics mentioned by this previous systematic review. This table also lists the studies that present some element that aggregates a given characteristic.

It is worth highlighting that the establishment of the relationship between the characteristics and architectural elements was based on a careful analysis of this domain literature, focusing on the characteristics and roles of each element identified by our systematic review. In the next paragraph, we discuss how each characteristic is associated to the elements, as shown in Table III.

TABLE III: CHARACTERISTICS OF UBIQUITOUS PROJECTS ASSOCIATED WITH THE COMMON ARCHITECTURAL ELEMENTS OF UBIQUITOUS SYSTEMS

Characteristic	Element	Studies
Service Omnipresence	Coupling and Mobility Mechanism	E2, E4
Invisibility	Sensor	E1, E2, E7, E11, E12
	Actuator	E3, E8
	Context Service	E1, E2, E7, E11, E12
	Actuation Service	E3, E8
Context Sensitivity	Sensor	E1-E3, E7-E9, E11, E12
	Context Service	E1, E2, E7, E11, E12
	Context Repository	E1-E3, E7-E9, E11, E13
	Reasoning Module	E2, E3, E8, E9, E11-E13
	Coupling and Mobility Mechanism	E8, E9, E11, E13
Adaptable Behavior	Context Service	E1, E2, E7, E11, E12
	Event Module	E5, E7, E9, E11
	Adaptation Module	E1, E5, E9, E13
Experience Capture	Reasoning Module	E4, E11, E12
Service Discovery	Event Module	E1, E9
Function Composition	Reasoning Module	E2, E3, E8, E9, E11, E12
	Coupling and Mobility Mechanism	E8, E9, E11, E13
Spontaneous Interoperability	Coupling and Mobility Mechanism	E2, E4
Heterogeneity of Devices	Sensor	E8, E9
	Event Module	E5, E11
Fault Tolerance	Coupling and Mobility Mechanism	E4
	Event Module	E5, E9
	Adaptation Module	E1, E5, E9
	Reasoning Module	E12
	Context Service	E12
	Security Module	E11

The **Service Omnipresence** characteristic can be supported by the Coupling Mechanism and Mobility mechanism, since it uses mobile communication protocols that allow access to services anywhere, anytime.

The **Invisibility** characteristic is related to: (i) the Sensor element, which captures context information from the environment, without any explicit order of the user; (ii) the Actuator element, which forwards the system's actions to the environment; (iii) the Context Service and Actuation Service, which are the architectural elements that enable access to sensors and actuators.

Context Sensitivity is a key feature of any ubiquitous system. Sensors and Context Services are directly related to this characteristic, allowing the identification of the context and the execution of operations according to the current context. The Context Repository is responsible for storing context information. The Reasoning Module performs inferences about contextual information and can produce new information. The Aggregation or Composition module performs the context information composition.

Adaptable Behavior defines that the system must adapt to the environment, offering services according to the current context. The Context Service is essential for the identification of the context, while the Event Module is responsible for triggering an event for context changing. After that, the adaptation can be performed. This characteristic may also be attributed to the Context Repository, as in E5. Finally, the Adaptation Mechanism performs the required adaptation for the new context. The

Experience Capture characteristic consists of capturing and storing information for future use. It is typically related to the Reasoning Module, which uses machine learning and other artificial intelligence techniques. This module has a role similar to the Aggregation or Composition Module found in some studies, such as E8. The existing difference between these modules lies in the fact that the Reasoning Module is able of generating new context information, while the Aggregation or Composition Module only groups or composes the context information. In most studies; however, these modules are integrated.

Service Discovery is supported, in most studies, by the Event Module, which is proactive in relation of services, monitoring and discovering available services, making them available through a publish-subscribe mechanism. However, this behavior may be aggregated to the Context Repository, as in E5.

Service Composition determines the system ability of providing new services to the final user, based on existing services. The Reasoning Module is related to this characteristic, since this module must be able of identifying the basic services (E2, E3, E8, E9, and E12) and compose them according to some business rule. The Aggregation or Composition Module, in some studies (E8, E9, E11), is used to perform the composition. In addition, the Reasoning Module can infer new contextual information to provide it as a new service. However, the new services that may be offered vary between applications.

Spontaneous Interoperability is the system ability of using many elements without the need of external intervention. This characteristic is supported by the Coupling and Mobility Mechanism, since this element is responsible for mobile computing protocols and for handling, in a high abstraction level, environment changes (E2 and E4).

The **Heterogeneity of Devices** characteristic defines that the distinct elements must be uniformly accessed. The E8 and E9 studies discuss the role of sensors in providing information from heterogeneous sources, as well as the role of the Event Module to monitor different services in a transparent way to users.

Regarding the **Fault Tolerance** characteristic, the Coupling and Mobility Mechanism is directly related to the mobile devices used by the users to access the system. Therefore, this mechanism must be able of handling the most common problems related to mobile computing, as connection instability and fluctuations in the data flow (as shown in E4). The Event Module may trigger many events, including faults or errors in any of the available services. The faults can the handled by the Adaptation Mechanism. In E12, the responsibility of fault tolerance is diffuse, whereas several elements detect and treat its own inappropriate behavior. The Security Module also supports this characteristic, by providing authentication and access control mechanisms.

In short, it is observed that the common architectural elements identified by this study adequately meet all the characteristics of ubiquitous systems stated by Spinola and Travassos [8].

Note that although only two studies (studies E2 and E4) explicitly presented the Coupling and Mobility Mechanism, it was identified that this element type is essential for ubiquitous systems, since these systems have essentially a mobility element, to allow the system be accessible anywhere. The E3 and E7 studies presented a query mechanism to recover context information from the Context Repository. However, we chose not to explicitly insert this element, since it was observed that this element is commonly implemented as part of the Context Repository, because it is highly dependent on the format of the stored context information. Many low-level or very specialized elements were not considered common architectural elements. For example, the Operating System and Network Protocol were not considered, since they were cited only by studies about low level architecture.

IV. THREATS TO VALIDITY

A major threat to validity of this systematic review refers to the completeness of this study, i.e., if in fact all the related papers were included. This problem may have occurred because relevant studies were not found by the search mechanisms, for instance, by the technical limitations of the search mechanisms. Another threat refers to the results and conclusions presented in the evaluation step. We tried to minimize those problems by adopting a dual revision approach for each paper, performed by the different reviewers of this work. This strategy contributes to reduce possible bias or misinterpretation. The findings were also validated by more than one reviewer. These strategies ensured that the set of the found architectural elements cover the essential requirements of an architecture for ubiquitous systems.

V. Conclusion and Future Work

The ubiquitous computing enables the use of contextual information from any environment at any time. Ubiquitous computing exploits technological advances in pervasive computing and mobile computing, integrating mobility, engagement, and distribution. Considering its relevance, attention to the development of ubiquitous systems is essential.

This work presented a literature review with the aim of summarizing the knowledge about reference architectures and common architectural elements for ubiquitous systems. As main result, the common, essential elements of ubiquitous systems were identified, analyzed, and summarized. This paper also mapped these elements in the main characteristics of ubiquitous systems. This mapping is important to verify that the identified elements meet the essential characteristics of ubiquitous systems. Furthermore, this set of elements can be considered as basis of any ubiquitous systems. Therefore, the identification of this set can be considered an important contribution to systematize the development of such systems. Moreover, we have observed that the four reference architectures found in our systematic review do not comprise all architectural elements identified in this work. In this scenario, as a future work, we intend to define a more complete, well-structured reference architecture. Thus, it is intended that this architecture can effectively contribute to the development of ubiquitous systems that have become increasingly important to our daily lives.

VI. ACKNOWLEDGEMENTS

This work is supported by Brazilian funding agencies FAPESP, Capes, and CNPq.

VII. References

[1] M. Weiser. “The Computer for Twenty-Frist Century”. Scientific American, September 1991.

[2] K. Lyytinen and Y. Yoo, “Issues and Challenges in Ubiquitous Computing”. Communications of the ACM. n. 12, v. 45, 2002. pp. 63-65.

[3] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice. Addison-Wesley, Boston, 1998.

[4] R. Cloutier et al, “The Concept of Reference Architectures”. Systems Engineering, 13. V. 13, n.1, UK, 2010, pp. 14-27.

[5] E. Y. Nakagawa, P. O. Antonino, and M. Becker, “Reference Architecture and Product Line Architecture: A Subtle but Critical Difference”. Proc. 5th European Conference on Software Architecture (ECSA'2011). Essen, Germany, 2011. pp. 207-211.

[6] T. Dyba, B. Kitxenham, and M. Jorgensem, “Evidence-Based software engineering for practitioners”. IEEE Software, v. 22, n. 1, 2005. pp. 58-65.

[7] B. Kitchenham, T. Dyba, and M. Jorgensen, “Evidence-Based Software Engineering”. Proc. 26th International Conference on Software Engineering (ICSE). IEEE Computer Society. Washington, DC, USA, 2004. pp. 273-28.

[8] R. Spínola and G. Travassos, “Towards a framework to characterize ubiquitous software projects”. Information and Software Technology. v. 54, 2012. pp. 759-785.

[9] J. Zhou et al., “PSC-RM: Reference Model for Pervasive Service Composition”. Proc. Fourth International Conference on Frontier of Computer Science and Technology. 2009. pp. 705-709

[10] Y. Liu and F. Li, “PCA: A Reference Architecture for Pervasive Computing”. Proc. 1st International Symposium on Pervasive Computing and Applications, 2006. pp. 99-103.

[11] T. Xu, B. David, R. Chalon, and Y. Zhou, “A context-aware middleware for ambient intelligence”. Proc. Workshop on Posters and Demos Track. ACM, NY, USA. 2011. pp. 10:1-10:2

[12] R. Shriram and V. Sugumaran, “Adaptive middleware architecture for information sharing on mobile phones”. Proc. 2007 ACM Symposium on Applied computing. ACM, New York, NY, USA. 2007. pp. 800-804

[13] S. W. Han, Y. B. Yoon, H. Y. Youn, and W. Cho, “A new middleware architecture for ubiquitous computing environment”. Proc. 2nd IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems. 2004. pp. 117-121

[14] A. Saeed and T. Waheed, “An extensive survey of context-aware middleware architectures”. Proc. IEEE International Conference on Electro/Information Technology (EIT), 2010. pp. 1-6

[15] C. Song, D. A. Lee, K. Chung, K. Rim, and J. Lee, “Interactive middleware architecture for lifelog based context awareness”. Multimedia Tools and Applications. Springer, US, 2013. pp. 1-14

[16] P. Eugster, B. Garbinato, and A. Holzer, “Middleware Support for Context-Aware Applications”. Proc. Middleware for Network Eccentric and Mobile Applications, Springer, 2009. pp. 305-322.

[17] M. Román, C.K. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt, “Gaia: A Middleware Infrastructure to Enable Active Spaces”. Proc. IEEE Pervasive Computing, 2002, v. 1, pp. 74-83

[18] V. Raychoudhury, J. Cão, M. Kumar, and D. Zhaung, “Middleware for pervasive computing: A survey”. Pervasive and Mobile Computing, April 2013, pp. 177–200,

[19] K. Da, M. Dalmau and P. Roose, “WaterCOM: An Architecture Model of Context-Oriented Middleware”, Proc. Workshops (WAINA), 2012 26th International Conference on Advanced Information Networking and Applications, 26-29 March 2012. pp. 53-60.

[20] A. Fernandez-Montes, J. A. Ortega, J. A. Alvarez, and L. Gonzalez-Abril, “Smart Environment Software Reference Architecture”. Proc. NCM '09. Fifth International Joint Conference on INC, IMS and IDC, 25-27 Aug. 2009, pp. 397-403