ARCHITECTURAL SURVEY OF CONTEXT-AWARE SYSTEMS IN PERVASIVE COMPUTING ENVIRONMENT

Moeiz Miraoui[1], Chakib Tadj[1], Chokri ben Amar[2]
[1]LATIS Laboratory, Université du Québec, École de technologie supérieure
1100, rue Notre-Dame Ouest, Montréal, Québec H3C 1K3 Canada
{moeiz.miraoui.1@ens,ctadj@ele}.etsmtl.ca
[2]REGIM Laboratory, Université de Sfax, École Nationale d'Ingénieurs de Sfax, Route de Soukra, B.P. W, 3038 Sfax – Tunisie
Chokri.benamar@enis.rnu.tn

ABSTRACT

The main characteristic of devices in a pervasive (or ubiquitous) computing system is their context awareness which allows them to provide proactively adapted services to user and to applications according to the global context. In order to support the development and to ease the implementation of context-aware systems, many architectures were proposed with characteristics related to the application domain and techniques used. A survey of such architectures that makes comparison between them and evaluates them is strongly recommended. Proposed surveys are either restricted to a limited number of architectures or do not offer a good comparison or their evaluation is not based on appropriate criteria which keep them as simple descriptions. Our aim is to make a survey of relevant architectures which mark the evolution of context-aware systems based on criteria related to pervasive computing. This survey will serve as a guide to developers of context-aware systems and help them to make architectural choices.

Keywords: achitecture, pervasive computing, context-awareness.

1 INTRODUCTION

Pervasive computing aims to provide proactively adapted services to both user and applications according to the global context. The main characteristic of devices in such system is their context awareness. Since its apparition, pervasive computing has required tools (architectures, frameworks and middleware), methods and concepts to support the development of a context-aware system and ease their design and implementation. System architecture is created early in the development process and permits the creation of a high level design of the system which takes into account the fulfillment of requirements' implementation. The architecture design is an important step in the development of context-aware systems. Many researchers have proposed several architectures, frameworks and middleware for context-aware systems with particularities related to the application domain and techniques used. To evaluate these proposed architectures, many surveys were done but they did not cover all architectures that mark the evolution of pervasive computing. They did not offer a solid comparison or evaluation and instead are simple descriptions. Even if they exist (in limited number of surveys), these surveys were not based on criteria related to pervasive computing particularities. Our aim, therefore, is to make a survey of relevant architectures that mark the evolution of context-aware systems beginning from localization-aware systems up to present context-aware systems. This survey presents a comparison and evaluation of architectures on various criteria which are considered important for pervasive computing such as: context abstraction level, communication model, reasoning system, extensibility and reusability. Our objective is to come up with a survey that will serve as a guide to developers and architecture designers of context-aware systems in a pervasive computing environment.

The rest of this paper is organized as follows, in section 2 we review some previous surveys done until now on context-aware architectures and show their weaknesses. In section 3 we present the evaluation and comparison criteria used and argue their use in pervasive computing. In section 4 we

present detailed descriptions of some relevant architectures of context-aware systems and show their strengths and weaknesses. Before concluding this paper, we summarize the characteristics of each architecture with regards to the criteria presented in section 3.

2 RELATED WORK

In the literature there were several surveys on context-aware system architectures and for us the most significant ones are the following. Beldauf et al. [1] proposed a survey of a good number of architectures. They focused in particular on layered architectures. Their survey was based on describing layers of different architectures and the mechanisms used in each layer. In spite of the diversity of architectures cited, the survey was not done based on pervasive computing criteria and did not show clearly the strengths and weaknesses of each one. Kjaer [2] did a middleware oriented survey but it deals also with some architectural aspects. It consists of a classification according to a taxonomy judged important by the author. This classification seems to us a more detailed description of classical context-aware system architecture layers (sensor, interpretation, context management and adaptation) rather than a consistent survey. As the former one, this survey was not made in order to compare architectures according to criteria related to pervasive computing. Abshik and Conway [3] described four architectures but they did not make an evaluation of them which renders their survey as a simple description. Wingrad [4] did a specific survey on organizational models of architectures relative to context-aware human-computer interaction which is not generic enough and kept his survey specific to that domain. Finally Henrikson et al. [5] did a brief survey as a part of their work on five architectures which seems to us the most interesting one. It makes a comparison of architectures according to criteria related to pervasive computing but this survey does not cover other architectures that mark the evolution of context-aware system architecture. The criteria used for comparison are rather oriented on distributed systems even though most of them are basic for pervasive computing.

3 EVALUATION CRITERIA

A pervasive environment has some specific characteristics which should be taken into account when evaluating context-aware architectures. In this survey, we will make an evaluation of these architectures based on some criteria that we consider relevant for pervasive computing. These criteria are: a) level of context abstraction, b) communication model, c) reasoning system, d) extensibility and e) reusability. We have chosen these criteria due to the following reasons:

- Pervasive system uses sensors of different kinds to perceive contextual information. Software architecture must hide the complexity of the physical sensors by providing a higher level of abstraction which makes it independent of physical sensor and enhances the reusability of architecture components.
- Pervasive system is composed of proactive devices that adapt to the current context without an explicit intervention from the user. This requires that devices embed a reasoning mechanism in order to take initiatives for a correct adaptation.
- Devices must be autonomous, independent from each other and can be easily connected. The peer-to-peer communication model seems the most appropriate for a pervasive system. It offers an easy way to tie devices and the network can be set up for a very modest investment and permits an easy sharing of contextual information among devices. It does need neither a dedicated material (server) nor software (operating system, data base management system, etc.)
- A pervasive system is characterized by its rapidly changing environment due to mobility; hence devices can be added or removed dynamically without affecting the entire operation of the global system (hardware extensibility).
- Pervasive computing is a new domain of computing. Its architecture should provide reusable components in order to ease their integration and reduce development effort.

4 CONTEXT-AWARE ARCHITECTURES

4.1 Active Badge

The Active Badge project [6] developed by Olivetti Research Ltd. aims to built a system for phone calls delivery according to the called person's localization. It permits the transport of phone call to the phone closest to the called person. The system uses badges which continuously emit infra-red signals at a given frequency. These badges are carried by personnel of an enterprise and each badge contains the carrier identification. The signals emitted by these badges are perceived by some receivers distributed in the whole edifice. The perceived signals are then sent to a server. The latter presents to a receptionist the information about badges carriers and their localization. This information helps the receptionist to deliver a call to the place closest to the called person (this task can be

done automatically). The active badge is based on a distributed architecture of sensors. The layered architecture (figure 1) of an application running on the server is composed of the following four layers:

- The network controller which supervises the operation of the sensor network.
- The information presentation which is responsible for data management and control of localization information.
- The data processing which selects the interesting information at the time of localization variation
- The user interface to display the textual information about badges variation position.

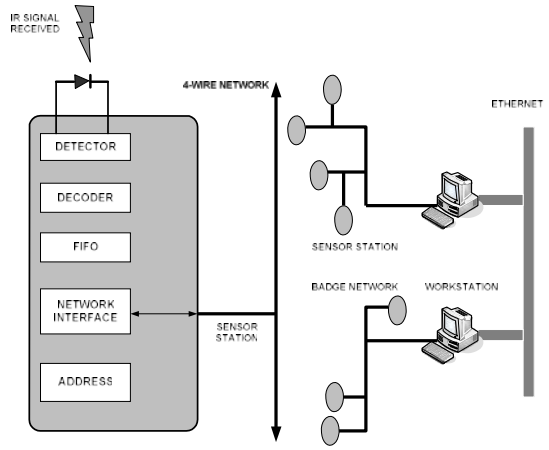

Figure 1: The Active Badge infrastructure

The active badge is a hardware architecture for localization-aware system rather than a software architecture of context-aware system with various software components. It is specific to localization systems and cannot be easily used for other kind of context-aware systems. Finally it does not make any abstraction of contextual information (localization information in this case) which makes it very dependent to the hardware infrastructure.

4.2 ParcTab

The Xerox project ParcTab [7] is a material infrastructure that enhances the development of applications aware to localization context (person location, surrounding devices, nearby people, etc.). The parcTab is a personal digital assistant (PDA) carried by the user and operates as a graphical terminal. It uses infra-red communication with a transmitter in a room of an edifice which communicates with a local area network via an RS-232 connection (figure 2). For each ParcTab there is a corresponding software agent that controls its communication with applications running on workstation in a local area network. This prevents Parctabs from making a lot of processing which consumes their limited resources. The communication system is based on the remote procedure call (RPC) between ParcTabs and applications running on a local area network workstation. This infrastructure permits the development of context-aware systems in particular those localization-aware. For example we can cite an incoming e-mail notification to a user based on his location and of nearby people by displaying the e-mail text on the parcTab displayer or by a simple beep (filters can be used to notify users for only emergency cases when the user is attending an assembly or a conference). The authors consider many others context-aware applications based on this infrastructure like remote program control, assisted collaboration, information and resources access according to the context, etc.

The ParcTab is a primitive localization-aware system based on hardware infrastructure like the Active Badge. The software architecture is very dependent on hardware and does not provide a good abstraction of contextual information.

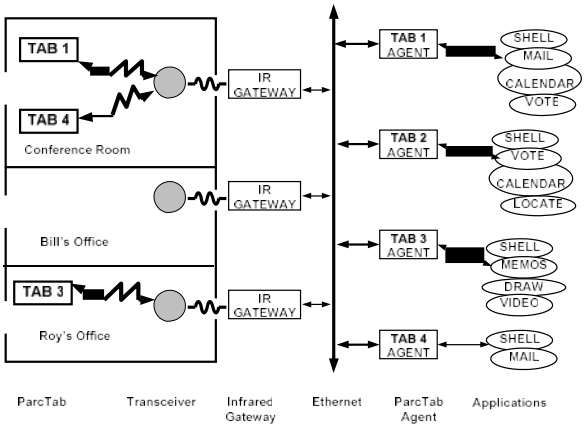

Figure 2: The ParcTab infrastructure

4.3 Stick-e-notes

The stick-e-notes project [8] is a framework to support the development of context-aware application where localization is the basic component of context. In this system the main component is a personal digital assistant (PDA) connected to a localization sensor (GPS or Active Badge). The PDA may communicate with one another depending on the application. The idea behind the stick-e-notes comes from stick notes used to remind user about something (or to briefly describe something) and stuck on a door, a device, etc. In this case, the notes are electronic and not hand written. Notes are written by the user and are attached to a specific context (example localization) and saved on his PDA. The electronic notes are automatically triggered (displayed by the PDA)

whenever the same context appears in the future. For example: the user attaches a description of a museum when he visited one, each time the user enter the same museum, the description note will be displayed on his PDA. Notes may be of different formats such as text, HTMl, sound, video, a program to execute, etc.
The authors defined four software components for the architecture:

- SEPREPARE: enables the user to prepare notes
- SEMANAGE: permits the management of notes
- SETRIGGER: enables notes triggering whenever similar context appears
- SESHOW: enables the display of triggered notes and their storage

Notes are written in SGML language for ease of information exchange.
The stick-e-notes use a limited set of contextual information (those related to localization) are hardware dependent (dedicated material) and do not provide a significant improvement of context abstraction as compared with previous systems.

4.4 Cyberguide and Guide

The cyberguide project [9] equips user with a personal electronic tourist guide aware of its context (localization, orientation, etc.). The hardware infrastructure is composed of a set of personal digital assistants (PDA) connected to some global positioning systems (GPS) to detect a tourist's position. These PDAs can communicate in infra-red among them or with a local area network. The objective is to guide a tourist in his visit by providing him with interesting sites to visit based on his actual location, paths to follow and some useful information depending on his current position. The cyberguide architecture is composed of the following elements:

- An electronic geographical card of the physical environment visited by the tourist with a special representation of remarkable objects (towers, park, museum, etc.)
- A browser that permits the detection of the tourist's current location in order to provide him with information related to the surrounding environment
- A messenger which provides a message delivery service to the tourist to send request, suggestion, communication with other tourists and to receive broadcasted messages

Another project called GUIDE [10] was proposed with the same objectives as the cyberguide. For us, it seems that the two projects are very similar with minor differences in the hardware used and web access.
These two projects are specific to localization systems, they do not interpret contextual information (to come up with a higher level of abstraction), are very dependent to the hardware used and do not offer an extensible and reusable software architecture.

4.5 CASS

The CASS tool [11] is a middleware for supporting the development of context-aware applications. It provides a good abstraction of contextual information and uses an object oriented model for context description. The architecture (figure 3) is based on a server containing a database of contextual information and a knowledge base with an inference engine to infer other contextual information using a back chaining mechanism. The mobile devices are equipped with various sensors to perceive context variation and send them to the server without local processing. Mobile devices and the server communicate via wireless mode. The server also contains a module for context interpretation that provides it with a higher level of abstraction. The architecture provides a good modularity that allows easy modification of server components in particular the inference engine. The mobile devices do not make any processing (all is done by the server) which limits the autonomy needed for pervasive systems but enhances the extensibility of the system (adding or removing devices require only the configuration of the server). CASS also provides a good abstraction of context due to its interpretation module and a reasoning mechanism which makes it more proactive however the centralized architecture is its weakness (if the server is down all the system will be affected and becomes non operational).

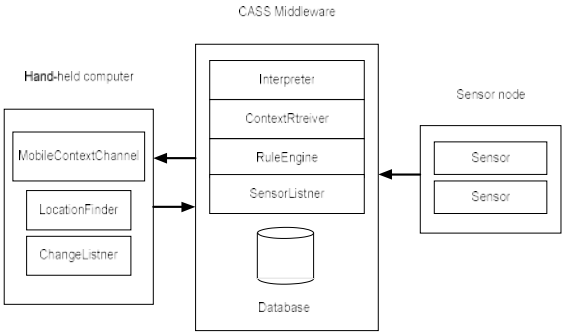

Figure 3: The CASS architecture

4.6 CORTEX

Biegel et al. [12] proposed the CORTEX framework to ease the development of context-aware mobile applications. The architecture is based on the "sentient object" which has some beneficial characteristics for pervasive computing environment

such as:

- Sensitivity: the capability of perceiving the state of the surrounding environment by using sensors
- Autonomy: the capability of operating independently of human control in a distributed manner
- Proactivity: take initiatives to achieve a goal

The sentient object contains two interfaces:

- Sensor of events perceived by sensors (sensor or consumer)
- Event emission to adapt to the current context (actuator or producer)

The core architecture (figure 4) is composed of:

- A module for fusion and interpretation of contextual information in order to increase their level of abstraction
- A module for a hierarchical representation of context in order to limit the actual situation context and then limit the set of possible actions
- An inference engine which specifies the applications behavior to a given context and uses the execution model event-condition-action

The communication between sentient objects, sensors and actuators that compose the system uses the mechanism based on events which are established dynamically during the system operation. This architecture presents many advantages as earlier stated but remains an ad hoc solution for a mobile network. The inference engine written in CLIPS language requires qualified personal to build, modify or adapt it to an other application which limit its usability. The discovery mechanism is not well detailed by authors and does not allow a measuring of the extensibility of the architecture. Also, the model of context used does not provide a complete set of contextual information needed for adaptation task.

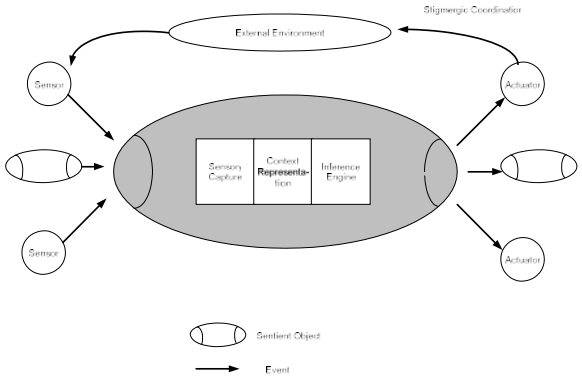

Figure 4: The CORTEX sentient object architecture

4.7 Context management framework

The CMF (context management framework) [13] allows semantic reasoning on context in real time and even in the presence of noise, incertitude and rapid variation of context. It delivers contextual information to applications by using a communication model based on events. The framework proposes a client/server (figure 5) architecture composed of the following basic components:

- Context manager: responsible for the storage of contextual information on server and the delivery of context to clients using different kinds of mechanisms (request/response, subscription/notification, etc.)
- Resource server: responsible for the acquisition of contextual information from physical sensors and their interpretation according to a specific format before sending them to the context manager
- Context recognition service: responsible for the conversion of the data stream to a presentation defined in the context ontology
- Change detection service: responsible for the detection of service change and therefore the context change
- Security: responsible for the verification and control of contextual information

The CMF uses ontology for context representation but does not offer a context reasoning module. It contains a good mechanism for context interpretation which provides a good abstraction of context and enhances the reusability in addition to a module for context security. It uses a server for context management (centralized system) which is the main problem since, when the server is down all the system will be affected and renders the devices less autonomous which is something not desirable in a pervasive computing system.

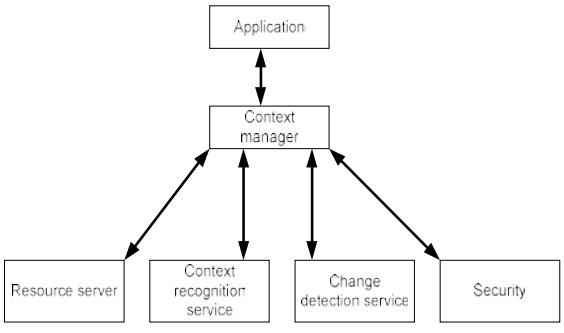

Figure 5: The CMF architecture

4.8 JCAF

Bardram [14] proposed the JCAF (java context

awareness framework) based on java programming language to support the development of context-aware applications. The JCAF architecture is composed of a set of components called “context service” communicating in a peer-to-peer mode. These components are responsible for collecting context information in a specific environment (room, hospital, laboratory, etc.). A context service contains four modules as follows (figure 6):

- Entity container: responsible for context exchange with context clients by using a communication mechanism based on events (subscription/notification). It also contains one or more entities that describe the context of an environment object (person, computer, doctor, patient, etc.)
- Transformer repository: provides basically two operations : context aggregation and translation between types of context
- Environment entity: allows communication between entities and control access to shared resources
- Access control: controls access to the entity via correct authentication of client’s query to access entity context
- Entity listener: it can be an entity of another context service and can access the entity context of a context service either by the request/response scheme or the subscription/notification scheme. It is possible to use the subscription/notification scheme according to the type of context.
- Context monitor: permits the acquisition of context via sensors and makes transformation of crude context
- Context actuator: permits commanding the actuators of the physical environment

The JCAF also controls the contextual information (trust on the information sensed by a particular sensor, error probability of information perceived by a sensor, etc.). The remote communication between the architecture components is done using java RMI (remote method invocation). The context service does not have an automatic discovery mechanism but can use a configuration file containing all others active context services.

The JCAF does not have a context reasoning mechanism and does not provide a good abstraction of context because there is no component that makes context interpretation in an explicit manner. The lack of an automatic discovery mechanism limits its extensibility but the JCAF offers reusable and portable modules because of its use of java language.

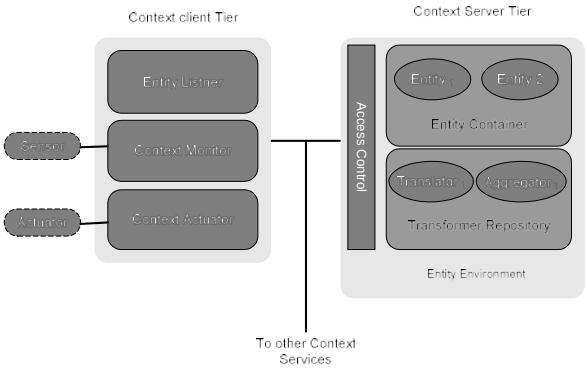

Figure 6: The JCAF architecture

4.9 Context toolkit

The context toolkit [15] was proposed as a tool to help the developers of context-aware systems. It has a layered architecture that permits the separation of context acquisition, representation and adaptation process. It is based on context widgets which operate similarly to graphical user interface widgets in order to hide the complexity of physical sensors. These widgets offer a good abstraction of context and provide reusable blocs for context sensing. The architecture (figure 7) is composed of the following components:

- Sensor: sensing of physical context
- Widgets: enable the encapsulation of contextual information and provide methods to access them in the same manner as graphical widgets
- Interpreters: make context transformation in order to provide a higher level of abstraction of context
- Aggregator: makes context grouping according to a subject or a situation
- Discoverer: maintains a register of existing capabilities in the framework (currently available components for use by applications)
- Service: executes actions for applications

This architecture is easy to implement, offers a distributed communication among system devices and reusable widgets but the discovery mechanism is centralized which does not make it a perfect peer-to-peer communication model. It has a limited extensibility when the number of devices increases. The architecture takes into account events (to notify context variation) by using a thread for each event which overloads the system and affects its performance. The architecture does not contain a layer or a module for context reasoning because the model used for context representation (key/value) does not permit a good reasoning.

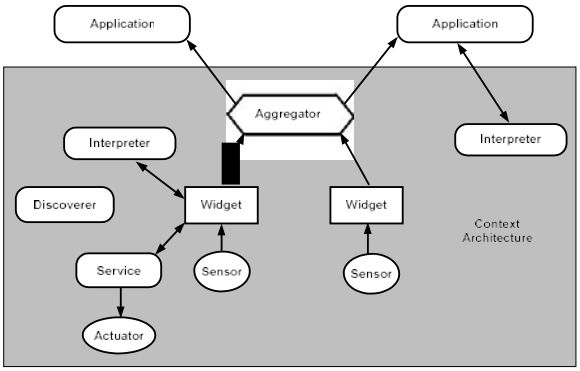

Figure 7.:The context toolkit architecture

4.10 Hydrogen

Hydrogen [16] is an architecture and a framework for context-aware systems. It is a three layered architecture that responds to particular requirements of mobile devices. The architecture (figure 8) has the following layers: adaptation, management and application. The context server (management layer) contains all the sensed information perceived by the sensors of the adaptor layer and provides context to the application layer of the attached device or other devices using a peer-to-peer communication model. The Hydrogen approach considers context as any pertinent information on an application environment and describes it using an object oriented model.

The architecture can be implemented easily, is simple and takes into account the limited resources of mobile devices (battery, memory, processing, etc.) and uses a peer-to-peer communication model (distributed). The adaptor layer does both the sensing and the interpretation task of context which does not offer a good abstraction of context and limits the reusability of such component. Also, it makes it very dependent to sensors. The architecture does not contain a reasoning module on context to ease the adaptation task.

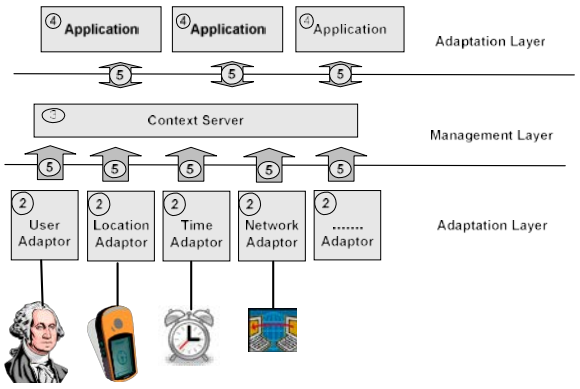

Figure 8: The Hydrogen architecture

4.11 SOCAM

SOCAM [17] is an architecture of a service oriented context-aware middleware for building and rapid prototyping of context-aware mobile services in an intelligent car. The architecture (figure 9) is composed of the following components: context provider, context interpreter, (context knowledge and context reasoner), service locating service, context-aware mobile service and context database. The architecture uses the client/server model where the context interpreter collects contextual information from context providers (internal or external) and context database and provides them to the context-aware mobile services and the service locating service. The main strength of the SOCAM architecture is its context reasoner which uses ontology for context description and allows a robust reasoning on context. It uses two classes of ontologies: domain specific and generalized ontologies. Several reasoning systems can be incorporated in the context interpreter to support a variety of reasoning tasks.

The architecture was proposed to support the development of a small non distributed application (intelligent car) which limits its use in a wide range of pervasive computing applications. The context interpreter is overloaded with an important quality of information (ontologies of different domains) which affects the global performance of the system but enhances its reusability, in addition to the major problem of a centralized architecture that contradicts the nature of a pervasive system which is a distributed one with autonomous devices.

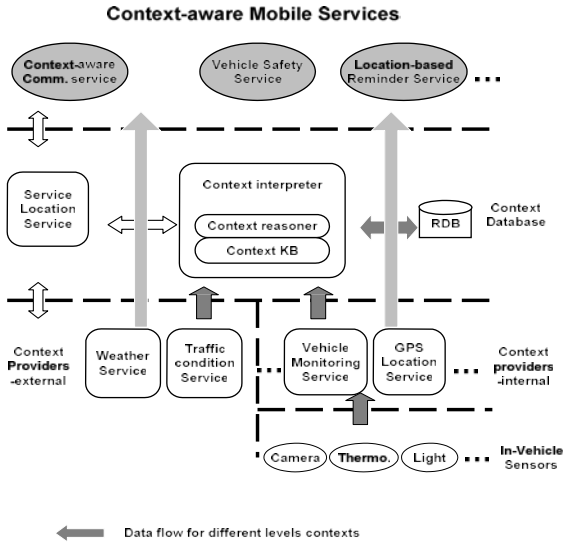

Figure 9: The SOCAM architecture

4.12 CoBrA

CoBrA [18] is an architecture based on broker

agent to support the development of context-aware applications in an intelligent space. The broker is an autonomous agent that manages and controls the context model of a specific domain. It runs on a dedicated computer (server) with powerful resources. The broker agent has a layered architecture (figure 10) containing the following components: context knowledge, context reasoner engine, context acquisition module and privacy management module. The broker agent collects context from devices, other agents and sensors of its surrounding environment and makes their fusion in a coherent model which will be shared among devices and their corresponding agents. CoBrA uses ontology for context description which allows a good reasoning and a better sharing of contextual information. It uses a centralized model for the storage and the processing of context in order to save the limited resources of mobile devices and uses a confidentiality policy for the user. The architecture requires a dedicated server for the broker which increases its cost and limits its usability in addition to the problem of a centralized architecture.

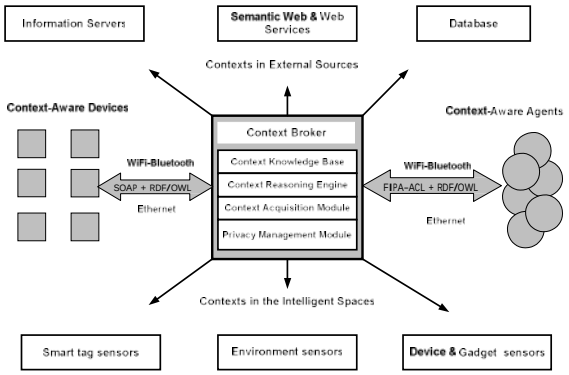

Figure 10: The CoBrA architecture

5 DISCUSSION

Except for primitive localization-aware system architecture, most of the proposed architectures make distinction between context sensing processes from its use. This permits an abstraction of low level sensing details and increases the extensibility and reusability of architecture components. Among proposed architecture, there are two approaches depending on whether contextual information are centralized or distributed. Most of these architectures are layered and composed of the following components:

- Sensor: physical sensing of contextual information.
- Interpretation: transformation of crude information into a more significant and useful information.
- Reasoning: (not present in all architecture) deduces and predicts new contextual information.
- Storage and management: basic operation in managing contextual information (add, remove, research, update, etc.)
- Adaptation: adaptation of provided services according to the current context

The proposed architectures are mostly specific to an application domain (localization systems, human-computer interaction, etc.) and require additional effort for their adaptation to other domains. Architectures based on a server suffer from the problem attributed to a centralized system: when the server breaks down, all other system components will be affected also it requires a dedicated hardware and software which increases its implementation cost. A centralized architecture contradicts the nature of contextual information in a pervasive computing system which is in general distributed and the mobility characteristic of devices in such environment. Rare are the architectures which contain all the layers mentioned above and most of them do not use a sound and reliable context model which permits efficient reasoning and eases the adaptation task. Context modeling is out of the scope of this paper but it is key concept for architecture design (context management layer and reasoning layer), A survey made by Strang et al. [19] containing an interesting comparative study of different modeling methods concludes that ontology makes the best description of context compared to other methods. A pervasive system is characterized by its rapidly changing environment due to mobility; hence devices can be added or removed dynamically without affecting the entire operation of the global system which requires a dynamic and automatic devices and resources discovery mechanism. This aspect was not deeply discussed in most architectures and needs more attention in future systems. Architectural design of context-aware systems needs more efforts in order to provide an appropriate architecture that suits pervasive system requirements. The table below summarizes characteristics of surveyed architectures (table1).

6 CONCLUSION

Context awareness is an important feature of applications in pervasive computing. In this survey, we presented relevant context-aware architectures that were proposed to support and ease the development of such system. For each architecture, we discuss its strength and weakness based on criteria that are related to pervasive computing. This survey shows that most of the proposed architectures are layered which allows the separation of context acquisition and context use in order to increase the level of context abstraction and hide the physical

sensing complexity. This enhances both reusability and extensibility of the system. In order to offer proactive systems, architectures embed a reasoning system to ease adaptation task which is not present in all architectures but it becomes a vital requirement for future systems. This survey aim is to serve as a guide to offer a useful recommendation to developers and designers of context-aware systems and help them decide on available architectural choice.

Table 1: Characteristics of surveyed architectures

	Context	Communication model	Basic software components	Context reasoning	Extensibility	Reusability
Active badge	--	C/S	-	--	-	-
ParcTab	--	C/S	-	--	-	-
Stick-e-note	-	P2P	-	--	-	-
Cyberguide	-	Hybrid	-	--	-	-
Context toolkit	++	P2P	Widget	-	-	++
CASS	++	C/S	Object	++	+	++
SOCAM	+	C/S	-	++	-	+
CORTEX	++	P2P	Sentient object	+	+	+
CoBrA	+	C/S	Agent	++	+	+
Hydrogen	-	P2P	Object	--	+	+
JCAF	++	P2P	context service	--	+	++
CMF	++	C/S	-	-	++	+

C/S: Client/Server P2P: Peer-to-peer

7 REFERENCES

[1] M. Baldauf, S. Dustdar, F. Rosenberg, "A Survey On Context-Aware Systems", International Journal of Ad Hoc and Ubiquitous Computing, 2(4), 63-277, Inderscience Publishers, 2007.

[2] K.E. Kjær, "A Survey of Context-Aware Middleware", Proceedings of the IASTED software engineering conference, 2007.

[3] A. Singh, M. Conway, "Survey of Context aware Frameworks – Analysis and Criticism", UNC-Chapel Hill ITS Version: 1, 2006.

[4] T. Winograd, "Architectures for Context", in Human-Computer Interaction Journal, 16:2-3, Special Issue on Context Aware Computing, 2001.

[5]K. Henricksen, J. Indulska, T. McFadden, S. Balasubramaniam, "Middleware for Distributed Context-Aware Systems", OTM Conferences, pp. 846-863, 2005

[6] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The Active Badge Location System", ACM Transcation onInformation Systems 10 (1), pp. 42-47, 1992.

[7] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis and M. Weiser, "An overview of the ParcTab ubiquitous computing experiment", IEEE Personal Communications, vol. 2, n. 6, pp. 28–43, December 1995.

[8] J. Pascoe, "The Stick-e Note Architecture: Extending the Interface Beyond the User", In Proceedings of International Conference on Intelligent User Interfaces. pp. 261-264, 1997.

[9] G. Abowd, C. Atkeson, .J Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: A mobile context-aware tour guide" CHI'96 Short paper, 1997.

[10] K. Cheverest, D. Nigel, M. Keith, F. Adrian, E.Christos, "Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences", CHI Letters, Volume 2, Issue No. 1, 2000.

[11] P. Fahy, S. Clarke, "CASS – a middleware for mobile context-aware applications", Workshop on Context-Awareness, MobiSys 2004.

[12] G. Biegel, V. Cahill, "A framework for developing mobile, context-aware applications", Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communication, pp.361–365, 2004.

[13] P. Korpipää, J. Mantyjarvi, J. Kela, H. Keranen, E-J. Malm, "Managing context information in mobile devices", IEEE Pervasive Computing, Vol. 2, No. 3, July–September, pp.42–51, 2003.

[14] J. E. Bardram, "The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications", In Hans Gellersen, Roy Want, and Albrecht Schmidt, editors, Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005), volume 3468 of Lecture Notes in Computer Science, pages 98–115, Munich, Germany, May 2005. Springer Verlag.

[15] A. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications", Human- Computer Interaction, 16:97–166, 2001.

[16] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, "Context-awareness on mobile devices – the hydrogen approach", Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2002 pp.292–302.

[17] T. Gu, X H. Wang, H. K. Pung, D. Q. Zhang. "A Middleware for Context-Aware Mobile Services", IEEE Vehicular Technology Conference. Milan, Italy, 2004.

[18] H. Chen, "An Intelligent Broker Architecture for Pervasive Context-Aware systems", PhD Thesis, University of Maryland, Baltimore County, 2004.

[19] T. Strang, C. Linnhoff-Popien, "A Context Modeling survey", In the first International Workshop on Advanced context modeling, Reasoning and management, UbiComp 2004.