A Service Oriented Architecture for Context-Aware Systems

Moeiz Miraoui, Chakib Tadj

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

Abstract. Devices in a pervasive system must be context-aware in order to provide services adapted to the global context. Architectures of context-aware systems should take into account many settings of a pervasive environment to be useful. Several context-aware architectures were proposed however most of them are specific to a particular application and have some limitation which reduces their usability by developers. In this paper and based on our previous work on defining context and context-awareness, we will propose a multi-agent service oriented architecture for context-aware systems. The architecture is based on the concept of service which plays an important part in the operation of a pervasive system. Our architecture is a hybrid form between client/server and peer-to-peer models. In addition of the advantages of those models, the proposed architecture enhances the extensibility, reusability, security of context-aware systems and takes into account the dynamic aspect of a pervasive environment. We will discuss in deep the contribution of the proposed architecture and its relation with other ones.

1. Introduction

In a pervasive system, the user's environment is populated with communicating smart devices. These devices provide adapted services to both the user and application. This adaptation is made according to the global context. Devices sense the global context and react proactively (without an explicit intervention of the user) to the changes of it. The goal is to help user in his everyday life's tasks. To-do so, devices in pervasive system must be context-aware thus context is a key concept in such systems. Context must be well understood to provide a better adaptation. In previous work [1] we have proposed a definition of both context and context-awareness based on the concept of service because it seems for us that this latter plays a crucial role in the adaptation task. Those definitions are more generic than others and make a good compromise between abstraction of the term and limitation of the set of required information for adaptation.

In the last few years many architectures of context-aware systems were proposed to support the development of these systems. Most of them are specific to particular applications and have limitations in many levels (management of contextual information, communication between devices, flexibility and reusability). In this paper we will present a multi-agent architecture for context-aware systems based on the concept of service. Our architecture takes into account the dynamic aspect of pervasive systems, more generic (applicable to a large variety of applications) and modular which enhance its reusability.

This paper is organized as follows: section II presents a review of previous architectures of context-aware systems, discuss them and then present our multi-agent architecture. We will argument both the use of such architecture (multi-agent) and the importance of the concept of service. In section III a discussion is presented to show the originality of our approach and our contribution. Finally, we will conclude this paper and present our further work.

2. Context-Aware Architectures

2.1. Previous architectures

Most of proposed architecture of context-aware systems makes a separation between the context sensing and the using process. This allows an abstraction of low level details of sensing and enhances the extensibility and reusability of systems. Among proposed architectures, there are basically two categories depending on whether the contextual information are centralized or distributed. The first strategy consists of using a context server where will be grouped the collected information from different sensors and provided to applications on demand. Examples of this category are, CoBrA [3], SOCAM [4], etc. In the second category, contextual information is distributed between entities of the system and uses the peer-to-peer communication model for information exchange. Examples of this category are the HYDROGEN architecture [2], the context toolkit [5] and the contexteur [6]. A survey on different architectures is done by Baldauf et al. [7]. It shows that most of them are layered architectures [8, 9, 10, 11] with basically the following layers:

- Sensing layer: it enables the physical sensing of contextual information using different kind of sensing.
- Interpretation layer: it makes transformation of gross information sensed by the previous layer to significant and useful information.
- Reasoning layer: it is not included in all architectures and permits the deduction of new information from existing ones and makes a prediction of context from previous cases which add the aspect of intelligence to the system.
- Management and storing layer: makes the management of contextual information (add, retrieve, up-date, searching, etc.) and the storing in an appropriate format.
- Adaptation layer: makes service adaptation according to context.

In the following we take a look at some architecture and analyze the approaches used in each. The context toolkit [5] was proposed to support development of context-aware systems. It is a layered architecture to separate the acquisition and the representation of context from the delivery to applications. It is based on context widgets acting similarly as GUI widgets to hide the complexity of physical sensors used by applications which gives context more abstraction and provides reusable and customizable building blocks of context sensing. The layers of the architecture are: sensors, widgets, interpreters, services and discoverers. The architecture is simple to implement, offer distributed communication between devices and reusable widgets however the discovery mechanism is centralized which makes it not a perfect peer-to-peer model, it has a limited scalability when the number of components increases, the event handling system consists of creating a thread for each event which implies an important overload of the system and does not include a context reasoning engine.

HYDROGEN [2] is architecture and a software framework to support context awareness. It is a three-layered architecture to suit special needs of mobile devices: adaptor layer, management layer (context server) and application layer. The context server contains all information sensed by adaptor layer and provides application layer with needed context or other devices by using a peer-to-peer communication. The hydrogen approach considers context as all relevant information about an application's environment and describes it using an object oriented model. This approach is very simple to implement, the three-layered architecture is located on one device thus making it robust against network disconnections, takes into account limited resources of a mobile devices (memory, CPU, etc.) and overcome the problem of centralized architecture by using a peer-to-peer communication model between devices. However the adaptor layer makes both the sensing and the interpretation of context which does not offer the abstraction level needed for context in context-awareness and makes context very dependant on sensors which affect the reusability of the architecture's components also the architecture does not include a reasoning layer about context which makes it a simple database of contextual information rather than a smart device that adapt dynamically according to current context.

SOCAM [4] is a service oriented context-aware middleware architecture for the building and rapid prototyping of context-aware mobile services in an intelligent vehicle environment. It is a layered architecture with the following layers: context providers, context interpreter (context reasoning and context knowledge), service locating service, context-aware mobile services and a context database. It is based on the client/server model where the context interpreter collects contextual information from context providers (internal/external) and context database and delivers them to both context-aware mobile services and service location service. The main strength of the SOCAM architecture is its reasoning system which uses ontologies to describe context enabling then formal analysis of domain knowledge (context reasoning). It uses two classes of ontologies: domain specific and generalized and multiple reasoners can be incorporated into context interpreter to support various kinds of reasoning tasks. However the architecture is used for the development of a small specific application (intelligent vehicle) which limits its usability for a wide range of pervasive systems. Both reusability and extensibility of the architecture are not argumented by the authors in particular, the component of the architecture are specific to the application developed and can not easily used for other systems (more open than a vehicle environment and with different characteristics) also to add/remove a device an explicit update of components is needed this is added to the major problem of a centralized model (when the server falls down).

CoBrA [3] is broker-centric agent architecture to support context-aware computing in smart spaces. The broker is an autonomous agent that manages and controls the context model of a specific domain and runs on a resource-rich stationary computer. The broker has a layered architecture composed of context knowledge base, context reasoning engine, context acquisition module and privacy management module. The broker acquires context information from devices, agents (applications) and sensors located in its environment and fuse them into a coherent model which is then shared with devices and their agents. CoBrA uses ontologies to describe context thus enabling a good reasoning mechanism and a better share of contextual information, it uses a centralized design for storage and processing because mobile devices in a pervasive system have limited computing resources and introduces the user's privacy policy. However the centralized design is also its weakness, when the broker falls down all the system is affected and become useless. The fact of using a server for context in a pervasive system contradict the nature of context in such a system which is generally distributed in addition to overloading the system with the using of a server running on a stationary computer.

Several other architectures exist but they do not differ a lot from the ones listed in this paper which seem to us the most relevant.

The modeling of contextual information is a fundamental step for every adaptation system. The modeling task consists of providing an abstract representation of contextual information in both data structure level and semantic level which ease their manipulation. Many methods of modeling were proposed having particularities related to used techniques. A detailed discussion of these methods is out of the scope of this paper but Strang and al. [12] did a survey on them and distinguished basically the following models for context representation: a) Key-value models, b) Mark-up scheme models, c) Graphical models, d) Object oriented models, e) Logic based models, and f) Ontology based models.

According to the authors, the context representation based on ontology model is a promising method for the design of context management system adapted to a pervasive computing environment.

The proposed architecture is specific to a particular application domain (human-computer interaction, mobile computing, etc.). Context server based systems have the principal problem of centralized systems: if the server breaks down, the other devices of the system will be automatically affected. It needs more effort for the administration task. Also it contradicts the nature of contextual information in pervasive system which is generally distributed. The peer-to-peer architecture in particular the one proposed by Dey and al. [5] doesn't allow an efficient reasoning on context because the method used to represent context is limited (key-value model) and does not support a robust logic reasoning. The architecture proposed by Rey and al. [6] requires a strong communication between entities and does not take into account the limited resources (energy, processing, etc.) of devices in a pervasive system and the possibility of loosing connection even though it uses a distributed architecture which offers many advantages compared to the context server architecture. To summarize, there are two major aspects that need more attention and work in most proposed architectures:

- Flexibility: architectures must be more flexible and take into account the dynamic changes in a pervasive environment (add, suppression of devices).
- Reusability: architectures must offer reusable modules to ease their integration in other pervasive systems in order to decrease the effort of development.

2.2. Service based architecture

The main objective of a pervasive system is to provide proactively adapted services to both the user and the applications. Thus the concept of service plays a crucial part for the operation of such a system. As we defined in our previous work [1] context and context-awareness based on the concept of service, we will use those definitions in order to design architecture for context-aware systems based on the same concept. The architecture is peer-to-peer and multi-agent (the use of multi-agent approach will be augmented progressively).

A pervasive system is composed of a set of communicating smart devices and provides adapted services according to the global context in various forms (various qualities of services) by using different media and modalities (figure 1). In our approach, for every device, a service will be modeled with a deterministic finite state automata whose states are the forms of services (among several forms) offered by a device. Transitions between states (from a service form to another) are due to changes in values of contextual information (since we defined context [1] as any information that trigger a service or change the quality (form) of a provided service). So we can easily limit the set of contextual information for every service and enable it to adapt accordingly.

For example calls indication for a cellular phone has several forms: ring tone, vibrator, ring tone with vibrator and silent. For simplicity, we will look to the two forms of this service: ring tone and ring tone with vibrator. Initially the cellular phone indicates calls with ring tone. It senses the level of noise of its environment; if it is high (over a fixed value) it changes automatically the form of service to ring tone with vibrator to draw the attention of the user. Figure 2 shows the finite state automata for calls indication.

In the following we will give the architecture for a service inside a device then the architecture of a device in a pervasive system and finally the global architecture. Figure 3 show that a service of a device may be provided in several forms. These forms are related with deterministic finite state automata. For each form there are a set of media and modalities depending on it.

In a pervasive system, a service has the following characteristics:

- Reactive entity: a service perceives the changes of the global context using its device's sensors and reacts by changing its form or its release.
- Autonomous: It can be provided in various forms independently of other services and devices.

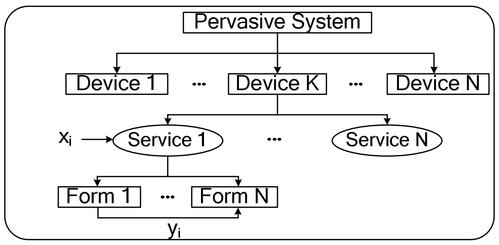

Fig. 1. A device provide several services in different forms: The change in value of x_i will trigger service1. The change in value of y_j will change the form of service.

- Proactive: It can be equipped with mechanisms enabling it to take initiatives to react (to predict the future form of a service).
- Sociability: it can exchange contextual information with other services to make aggregation (fusion) of context or to acquire useful information which is not provided by its main device.

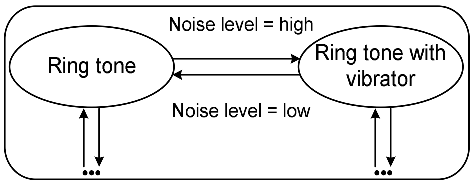

Fig. 2. A part of the finite state automata for calls indication.

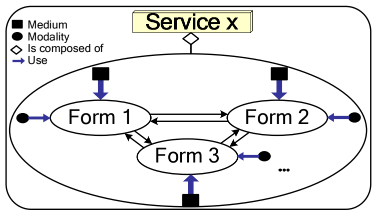

Fig. 3. Example of a service having three forms.

These are the principal characteristics of a software agent. For this reason, each service of a device will be modeled by an agent with internal states. These agents will be controlled and managed by a central agent (device agent) who plays the part of a context server for other agents (service agents). Each agent must be registered for a set of contextual information that concerns it. The service agent will be notified by the central agent each time the value of one of these information changes (to limit information flow). Also the communication with other service agents of the same device will be made only via the central agent (figure 4). In the case of a simple sensor, it will be modeled by a central agent without service agents since it provides one type of services (collect information).

The central agent of a device perceives the changes of the values of contextual information via the device's sensors or by communication with the other central agents of the other devices composing the pervasive system. If a change of contextual information concerns one of its service agents, then it communicates to it the change of value. With this intention, each central agent holds a list of contextual information which concerns each one of its service agents. With the reception of this information, the concerned service agent reacts either by a transition in its finite state automata to change the form of service or by triggering the service it self (initial form).

The central agent has layered architecture containing in addition of basic layers (sensors, interpretation, reasoning, management/storing and adaptation) two modules of communication: internal module for exchange of information with the device service agents and an external module for the exchange of information with the other central agents of the pervasive system. If a central agent perceives or senses the change of value of a contextual information (through the sensors of the device) and which concerns or not one of its service agents it broadcasts the new value to all central agents to enable other agents to react and will be notified by concerned central agents. This will allow the broadcaster central agent to: a) remember next time the central agents concerned with that information and b) up-date the list of active central agents in the pervasive system. For that reason we envisage to add for each central agent a cache memory for this type of transactions with an emptying system FIFO (first in first out) when the memory is full. Also to use an internal cache memory for internal transactions with the service agents (figure 5). The change of value of the same contextual information can be broadcasted by several central agents. This enables it to apply a fusion process for better control of errors.

Inside a device the central agent is a server to the service agents of the device (client/server model) and in the whole pervasive system, it can be either a server or a client for the other central agents (peer-to-peer model). The agent service has just two tasks: a) trigger a service and b) change the form of a service (make a transition in his automata) all the others tasks are made by the central agent (server).

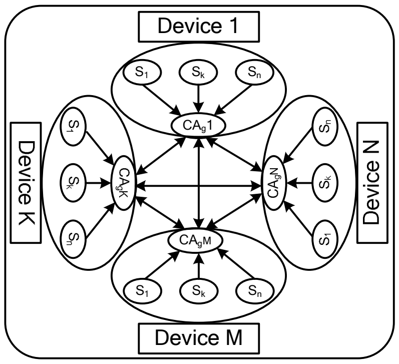

Fig. 4. The global multi-agents architecture. $\mathbf{S_i}$: Service agent i, $\mathbf{CA_g}$ **i**: Central Agent i, $\leftrightarrow$: Communication.

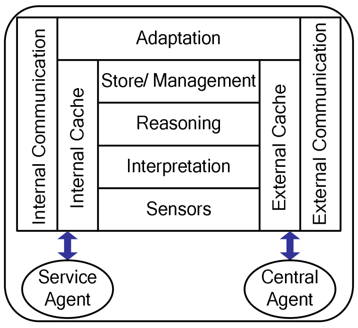

Fig. 5. Central agent architecture.

3. Discussion

The architecture that we proposed is hybrid architecture: client/server in the level of the multi-agent system (central agent is a server and service agents are clients) of each device and a peer-to-peer in the level of multi-agent system of the set of devices that compose the pervasive system (central agents). This makes possible to benefit from the advantages of the two architectures. Peer-to-peer architecture allows distributing contextual information and the processing relative to this information between the entities (devices) of the system. This reduces the load of processing and decrease the information flow between them. Client/server architecture allows an effective control of concurrent and multiple accesses to contextual information (central agent) and discharges the clients (service agents) from the operations of storage and information management.

In the multi-agent peer-to-peer architecture for the central agents, if a device of the pervasive system breaks down, its central agent will be inaccessible (inactive), but it does not affect the operation of the system. Only the services provided by that device become unavailable. A replacement by services provided by others devices is possible.

In the same context, it is possible to add new devices to the system without disturbing it since it will detect the new device automatically if it exchanges information with one of the components of the system (extensible dynamic architecture).

This architecture is reusable for the following reasons: the multi-agent system of a device (central agent and service agents) can be used in another pervasive system with minor modifications because the services offered by a device are the same ones (example: a GPS device always provides the geographical coordinates whatever its use).

The encapsulation of service details of a device as well as the contextual information management model within the device reinforce the protection of the device against undesirable accesses which can damage its quality of services (the access is done via an interface: central agent).

In this architecture, we did not consider the user's context because it's implicitly included in the device's context (example: the localization of the user is detected by a GPS that is with the procession of the user).

4. Conclusion And Further Work

The main characteristic of devices in a pervasive system is their context-awareness in order to provide adapted services. The design of architecture for such systems is a basic task for their development and implementation. Most of proposed architectures do not provide a lot of help to developers and have some limitations. In the light of existing architectures of context-aware systems, and based on the concept of service, we have proposed a multi-agent architecture. We discussed the characteristics and strong points of such architecture that can be summarized in the following points: hybrid architecture between client/server and peer-to-peer which takes advantages from both architectures. Extensible with few modifications thus dynamic (add and removal of devices). Reusable since the services of each device remain almost unchanged whatever the enclosed system thus it is possible to use it with its multi-agent system in another application. Easy to implement by using ready components of multi-agent domain in particular the mechanism of communication.

Our outlined architecture requires completion on the level of communications inter-devices and intra-devices by using the existing means of communication in multi-agent systems as well as the context representation model. We plan to use ontology which seems to us suitable for such architecture. Finally we will work on the reasoning layer of the central agent to make it more intelligent.

References

[1] M. Miraoui, C. Tadj, "A service oriented definition of context for pervasive computing," in Proceedings of the 16th International Conference on Computing, IEEE computer society press, Mexico city, Mexico, November 2007.

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

[3] H. Chen, T. Finin, A. Joshi, "An ontology for context-aware pervasive computing environments, " Knowledge Engineering Review, vol. 18, pp. 197-207, 2003

[4] T. Gu, H. K. Pung, D. Q. Zhang, " A middleware for building context-aware mobile services", in the proceedings of IEEE Vehicular Technology Conference (VTC 2004). Milan, Italy

[5] A. K. Dey, G. D. Abowd, " A Conceptual framework and a toolkit for supporting rapid prototyping of context-aware applications", anchor article of a special issue on Context-Aware ComputingHuman-Computer Interaction (HCI) Journal, Vol. 16 (2-4), 2001, pp. 97-166.

[6] G. Rey, J. Coutaz, " Le contexteur : capture et distribution dynamique d'information contextuelle ", ACM transaction pp. 131-138 Mobilité & ubiquité, 2004

[7] M. baldauf, S. Dustdar, F. Rosenberg, "A survey on context-aware systems" International Journal of Ad Hoc and Ubiquitous Computing forthcoming, 2004

[8] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Keränen, E. Malm. "Managing Context Information in Mobile Devices". IEEE Pervasive Computing. 2003

[9] P. Fahy, S. Clarke. "CASS – Middleware for Mobile Context-Aware Applications". MobiSys 2004

[10] G. Biegel, V. Cahill. "A Framework for Developing Mobile, Context-aware Applications". In Proceedings of 2nd IEEE conference on Pervasive computing and Communications, Percom 2004

[11] M. Román, C. Hess, R. Cerqueira, A. Ranganat, R. H. Campbell, K. Nahrstedt. "Gaia: A Middleware Infrastructure to Enable Active Spaces". In IEEE Pervasive Computing, Oct-Dec 2002.

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