Electronic Communications of the EASST
Volume 11 (2008)

Proceedings of the First International DisCoTec Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services (CAMPUS 2008)

Survey: Agent-based Middlewares for Context Awareness

Nabil Sahli

12 pages

Guest Editors: Romain Rouvoy, Mauro Caporuscio, Michael Wagner
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Survey: Agent-based Middlewares for Context Awareness

Nabil Sahli

nabil.sahli@telin.nl, http://www.telin.nl/
Telematica Instituut, Postbus 589, 7500 AN Enschede, The Netherlands

Abstract: In the last few years, many middlewares for context awareness have claimed to be agent-based. In this paper, we make a survey on the most known frameworks. We classify them according to their level of conformity to the agent paradigm and we discuss the usefulness of agents in these frameworks. Based on this survey, we enumerate several advantages of using agents in context-aware middlewares and give illustrative examples. We also point to the weakness of existing frameworks and identify challenges to be addressed.

Keywords: Agent paradigm, middleware, context awareness

1 Introduction

Context-aware systems can be implemented in many ways. The approach depends on special requirements and conditions such as the location of sensors (local or remote), the amount of possible users, the available resources of the used devices (PCs or small mobile devices) or the facility of a further extension of the system. Chen [Che04] presents three different approaches on how to acquire contextual information: Direct sensor access, Context server, Middleware infrastructure. In this paper we focus on the third type. The middleware-based approach introduces a layered architecture to context-aware systems with the intention of hiding low-level sensing details. Compared to direct sensor access, this technique eases extensibility since the client code has not to be modified anymore.

Several architectures for these middlewares have been proposed. The most common design approach for distributed context-aware frameworks is a classical hierarchical infrastructure with one or many centralized components using a layered architecture. This approach is useful to overcome memory and processor constraints of small mobile devices but provides one single point of failure and thereby lacks robustness. Gaia [Gai] is an example of such type of architecture. The Context Toolkit [SDA99], another context-aware framework, takes a step toward a peer-to-peer architecture but it still needs a centralized discoverer where distributed sensor units (called widgets), interpreters and aggregators are registered in order to be found by client applications. Many layered context-aware systems and frameworks have thus evolved during the last years. Most of them differ in functional range, location, and naming of layers. However, other middleware solutions have proposed an agent-based system as an alternative approach, which brings more decentralization.

According to the definition proposed by Maes [Mae94] "Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed", there are important aspects of software agents and their environment which are necessary for

any (autonomous) action. An agent has then to be: (i) *autonomous*: acts on its own Compulsory, (ii) *reactive*: responds timely to changes in its environment, (iii) *proactive*: initiates actions that affect its environment, and (iv) *communicative*: exchanges information with users and other agents. The aforementioned characteristics need to be there for all agents. However, some agents may have optional properties such as *mobility* (migrating from one site to another) or *adaptability* (mainly learning) capabilities. We will see through the state of the art which properties researchers use in their frameworks. We also use the aforementioned properties to classify different middlewares according to their conformity to the agent paradigm.

The remaining paper is organized as follows: Section 2 presents several agent-based frameworks for context awareness. These frameworks are classified according to their level of conformity to the agent paradigm. The usefulness of agents in these frameworks is also discussed in this section. Section 3 enumerates the advantages of using agents in context awareness. Section 4 discusses the challenges (mainly quality of context, learning, and security aspects) the agent-based approach has to cope with in order to implement better context-aware middlewares. Finally, Section 5 concludes this paper.

2 State of the art

In order to study the different existing agent-based middlewares for context awareness, we classify frameworks into three main classes according to their level of conformity to the agent paradigm and more particularly to the agent's properties: (i) *class 1*: Agents are only communicative, (ii) *class 2*: Agents are also autonomous, reactive and proactive, and (iii) *class 3*: Agents are also mobile. A fourth class could have been considered (*class 4*: agents also learn), but none of the middlewares addressed in this report really supports *learning*.

2.1 Class 1: agents are only communicative

In this class of middlewares, "agents" (or more likely "processes") communicate between each other in order to fulfill the required tasks. The use of agents in these middlewares is merely an implementation choice that offers modularity and adaptability. For example, in BerlinTainment [WCRS04], CAMPS [QSS06] and [WW07], "agents" represent different modules that provide functionalities for mobile services but do not behave as autonomous and reactive agents. Agents in StarCCM [ZWH+06] are only used to collect data from sensors. An agent acts as the mediator between the context collector and the sensors, and it reports the context to the Context collector periodically. In [SDSP07], part of (CHIL) project which is one of the most prominent European research initiatives in the areas of pervasive computing and multi-modal interfaces, authors use an agent-based architecture to take advantage of the JADE platform.

2.2 Class 2: agents are also autonomous, reactive and proactive

In this class of middlewares, communicative agents are also autonomous, reactive, and proactive. Consequently, frameworks presented in this section could be named agent-based approaches since they fulfill the minimum set of required properties of an agent. Nevertheless, not all of these middlewares address the three properties with the same level of importance. Actually,

each proposed system focuses more or less on each of these properties depending on the middleware's objectives. In [PL07], by using the BDI (Belief-Desire-Intention) concept, the framework takes advantage of two important properties of agents: autonomy and pro-activeness. Agents, representing users are capable to perform tasks and solve conflicts while taking into account current context and goals of their users. AMUSE [TSS05] is based on the agentification (process of making a target entity workable as an agent by adding knowledge processing mechanism to the entity) of each entity of the environment. The agent-based approach takes advantage of the reasoning and cooperative interaction capabilities of agents. Agents organize themselves into groups to provide the required service. A service is then constructed by a combination of entities controlled by agents. Stu21 [SC06] and ACAI [KK05] are examples of middlewares which take advantage of agents' capabilities to design agents which are highly aware of context (able to perform advanced operations on context, see farther).

2.3 Class 3: agents are also mobile

In this third class, middlewares propose a mobile-agent-based approach. They try to take advantage of one of the optional property of agents (mobility). Even if still in its infancy, the composite research area of context-aware mobile-agent-based middlewares tends to exhibit some common guidelines for the integration of mobile agents and context-specific functions. AMASE [KRS99] is a good example which takes full advantage of agents' mobility in order to provide users with better context-aware services. It enables users of mobile handheld devices to access value-added services over wireless networks. In [HK05], the authors go one step farther by assigning a personalized profile and a specific role to mobile users at visited sites.

2.4 Discussion and Analysis

In this sub-section, we aim at refining the previous classification in order to make better comparison between the different middlewares. According to the above state of the art, we notice that software agents are used within context-aware middleware for different purposes. We enumerate these purposes in what follows:

- Implementation: agent-based approach is used as a way of implementing distributed systems (since it guarantees more modularity and adaptability).
- Context awareness: agents are aware of the context and have their own context model.
- Reasoning: agents can reason about the context, user's requests and plans, etc. in order to make good decisions.
- Negotiation: agents negotiate resources within a dynamic environment.
- Service customization and adaptability: agents provide users with more customized and adaptable services.
- Connectivity: agents cope with connectivity problems (disconnection, low-bandwidth).
- Security/Privacy: agents are used for security or privacy purposes.

We propose to classify the middlewares according to the above criteria. Table 1 shows for what reason(s) each middleware has used the agent paradigm. Now, if a certain criterion (e.g., "Context awareness") is not marked with $\surd$ (in Table 1), this does not mean that the middleware does not support this feature (here context awareness). It only means that the middleware does not use the agent paradigm to achieve that purpose. (+) and (-) correspond to our qualitative and subjective judgments on the achievement of a given property by a framework. Table 1 also indicates which properties (of agents) are used for these purposes (last column). Agents can be: Autonomous (Aut), Reactive (R), Proactive (P), Communicative (Com), Mobile (M), and Adaptive (Adap). From this benchmark, we reach the following conclusions:

	Implementation	Context awareness	Reasoning	Negotiation	Service customization & adaptability	Connectivity	Security/ Privacy	Context QoS	Agent properties
BerlinTainment (2004)	√						√ (-) (S1)		Com
Wang & Wang (2007)	√					√ (-) (C1)			Com, M
CAMPS (2006)	√								Com
StarCCM (2006)	√						√ (-) (S2)		Com
CHIL (2007)	√								Com
Plesa & Logrippo (2007)	√	√ (-) (CA1)	√ (++) (R1)	√ (+) (N1)	√ (+) (SCA1)				Aut, R, P, Com
AMUSE (2005)	√	√ (+) (CA2)	√ (-) (R2)		√ (++) (SCA2)		√ (+) (S3)	√ (+) (Q1)	Aut, R, P, Com
Stu21 (2006)	√	√ (++) (CA3)	√ (+) (R3)						Aut, R, P, Com, Adap
ACAI (2005)	√	√ (+++) (CA4)	√ (++) (R4)	√ (+) (N2)					Aut, R, P, Com
Harroud & Karmouch (2005)	√	√ (+) (CA5)			√ (+) (SCA3)	√ (+) (C2)	√ (+) (S4)		Aut, R, P, Com, M
AMASE (1999)	√			√ (+) (N3)	√ (+) (SCA4)	√ (++) (C3)	√ (++) (S5)		Aut, R, P, Com, M

Table 1: Classification according to different purposes of using agents, the 3 classes are separated by thick lines (see Figure 1 for captions)

- Many frameworks (mainly those of Class 1) use agents as a means of implementing a distributed system. Indeed, agents offer some interesting features which make the implementation easier. This is discussed in Section 3.

- The agent paradigm offers a significant support for context-aware middlewares and especially in the following areas: context awareness (modeling and processing), reasoning, negotiation, customization and adaptability for services, and network connectivity. These will be also presented as eventual advantages of using agents for context-aware middlewares (see Section 3).

- Aspects like Security/Privacy and Context Quality of Service (QoS) are less addressed by the agent-based approach. Such aspects constitute potential future challenges for agent-based context-aware middlewares (see Section 4).

(**CA1**) The choice of plans must take into account the current context of the user and decisions are made based on the most recent information about the environment. The plans that drive the system's behavior are expressed in terms of user context, which makes new plans more adaptive and flexible.
(**CA2**) AMUSE proposes context management ability and cooperation ability to resolve context conflict to the agents. Moreover, it embeds long-term-context maintenance ability to the agents to accumulate cooperation history and experiences.
(**CA3**) Stu21 proposes *context aware* agents which offer several features concerning context awareness: keeps a model of context, listens to context changes, uses RDF/XML to interact with other agents about context, tec.
(**CA4**) ACAI proposes agents which fulfill several requirements of context awareness (e.g. context composition, inferring, inter-domain context invocation, etc.).
(**CA5**) The framework uses context policies in order to ensure that the context is relevant at each visited site (for mobile agents).
(**R1**) Agents in this middleware are based on the Belief-Desire-Intension (BDI) concept.
(**R2**) Agents in AMUSE employ an inference mechanism to perform interaction with other agents and control the entities they represent.
(**R3**) The *context aware agent* in Stu21 has reasoning capabilities (based in rules inference) to reason about the context (e.g., reasons over its own context to deduce beliefs and goals).
(**R4**) A *Reasoner agent* RA contains a new hybrid inference system that integrates logical reasoning, fuzzy reasoning and semantic rule representation into one system. The RA uses data captured from sensors and from users' and services' profiles to deduce new context information and to trigger actions. The RA uses logic-reasoning mechanisms to ensure that instances of captured context are consistent both with each other and with arguments defined in the ontology.
(**N1**) Communicating agents negotiate preferences for users in order to avoid possible conflicts.
(**N2**) Autonomous agents representing entities in the environment negotiate context specifications.
(**N3**) Agents negotiate resources during their migration.
(**SCA1**) The planning mechanism offers a certain level of customization and adaptability (agents choose appropriate plans and also retract properly and select an alternative plan in case of plan failure).
(**SCA2**) The organization and re-organization mechanism processed by different agents guarantees customized and adaptable services.
(**SCA3**) This is handled by *context policies*
(**SCA4**) Mobile agents in AMASE are pro-active and run autonomously in the network, they can collect information and deliver it to their users as soon as the requested information becomes available.
(**C1**) Even if they mentioned that some agents could be mobile, the authors did not use this feature in the proposed middleware.
(**C2**) Agents are mobile, they move from one site to another. A site assistant at the visited site establishes a negotiation process with the site assistant at the home site to determine the visiting user profile and the authorized services.
(**C3**) Agents are mobile and their migration depends on the current network QoS. Mobile agents can also load remote codes. Through this dynamic code loading mechanism at creation time, the used mobile agents are automatically updated to newer versions.
(**S1**) The FIPA-compliant MAS-architecture *Java Intelligent Agent Componentware* (JIAC) used by the middleware, offers components realizing security functionality (preventing unauthorized service usage and prohibiting agents from attacking other agents or platforms).
(**S2**) This is provided by CORBA (the model component used by StarCCM).
(**S3**) AMUSE has an agent Security Manager (part of the agent platform) which allows the enforcement of access policies for agent migration and resources.
(**S4**) Both security and privacy are handled by the policy mechanism.
(**S5**) AMASE contains security features for user authentication, agent signing and access control. When the user sends an agent, constant parts of the agent are signed using the private key of the user. The agent system of course needs the public key of the user. This key is stored as a X509 certificate either in the agent or it can be retrieved from an AMASE directory. AMASE supports access control lists. A service agent can create access control lists and define which rights a user will be granted.
(**Q1**) To realize QoS-aware ubiquitous service construction, while considering multiple contexts, AMUSE proposes contract-based service construction scheme of agents. QoS awareness of the system is also measured. AMUSE investigates how much the QoS awareness is improved. To measure the QoS awareness, a User Request Achievement level is deployed. Using this metrics, the system can measure how much the user requirement is fulfilled with provided quality of service.

Figure 1: Captions

- An important property of agents (Adaptability, which includes learning) is hardly used. Integrating learning mechanism could however be a good value-added for the next generation of agent-based context-aware middlewares (see Sections 3 and 4).

3 Advantages

Based on the state-of-the-art and on the conclusions drawn from the comparative table (Table 1), we identify five main advantages of using an agent-based approach for context-aware middlewares. These advantages are related to the five following aspects: implementation, reasoning, negotiation, scalability and adaptability, context awareness, and mobility.

3.1 Implementation

One of the main advantages of using agents in implementing distributed systems is modularity. In fact, agent-based applications are mainly configured by selecting and defining the participating agents. Therefore different modules made up by groups of agents may be changed easily. These systems can also take advantages of the communication facilities offered by agents (e.g. standardized Foundation for Intelligent Physical Agents FIPA). This explains why several context-aware middlewares use agents in the service layer. JADE is probably one of the most used platforms to build an agent-based system.

3.2 Reasoning

Thanks to their autonomy and pro-activity, agents are well suited for reasoning. In a context-aware middleware, this capability is useful to achieve several goals, such as to:

- Plan: agents are able to make a choice between different plans (in case there are a number of alternative plans for achieving the same goal) and apply appropriate decision procedures. If a plan fails, agents are able to retract properly and select an alternative plan. Finally, when the user has a number of goals that cannot be achieved simultaneously, agents are able to make a decision about which goals to try to achieve. All these features are useful in context awareness systems as suggested by [PL07].

- Derive behaviours from context: reasoning takes place when an agent needs to make sense out of the contextual information that is captured from the sensing infrastructure or other agents, and when an agent needs to determine its subsequent behaviour. When a piece of contextual information is captured, agents are able to provide their own interpretation of the contextual information and to determine their subsequent behaviours. Chen et al., developed a proof-of-concept agent system called CoolAgent [CTS+01] that demonstrates the significance of logical reasoning and ontology sharing in a context-aware distributed computing environment.

- Reason about uncertain context: agents can have fuzzy-logic based inference capability that allows them to reason about uncertain context, and to provide actions that are not explicitly defined [KK05].

3.3 Negotiation

In context-aware systems, negotiation can be necessary to solve conflicts. Agents are largely used for negotiation within the Artificial Intelligence community. Similarly, agents can also be used to support context-aware middlewares with negotiation process. In what follows three examples of agents negotiating in a context-aware system (drawn out of the state of the art of this paper):

- Contexts collected from different sources can be contradictory or conflicting. Agents (holding conflicting contexts) are able to negotiate (here, cooperatively) to solve context conflicts [PL07].

- When agents are mobile, negotiation between the resource manager and the agent that represents a service must take place to make sure that the necessary resources are available at the time the service is requested by the mobile user [HK05].

- Resource availability tends to be poor and unstable in ubiquitous environment. Consequently, not only user context, but also resource context of hardware, network and software should be considered. In addition, multiple users would be given the ubiquitous services simultaneously, thus problems of effective resource sharing and assignment should be addressed. In [PL07], agents negotiate in order to cope with these problems.

3.4 Scalability and Adaptability

Agent-based approach offers two other advantages, namely adaptability and scalability.

- Scalability: context-aware applications must be scaled to different users and devices. Scalability should also be given according to different kinds of networks showing different QoS characteristics. This can be achieved by agents either through re-configuration, dynamic adaptation of the runtime behaviour, duplication (duplicating the agents responsible for critical tasks, thus distributing the load between multiple identical agents and removing bottlenecks), or mobility [HK05, KRS99] (mobile agents can negotiate their resource requirements with the host systems to ensure a certain QoS). In [HK05], the authors demonstrate how, by using policies, they can enable dynamic changes in the behaviour of the system in a flexible and scalable manner.

- Adaptability: By using mobile agents, applications can provide value-added features which are based upon the adaptation of services. Since agents are pro-active and run autonomously in the network, they can collect information and deliver it to their users as soon as the requested information becomes available. Besides, agent environment may be reconfigured at runtime, i.e. agents may be added or removed to adapt the functionality provided (when offering a new service for example) [KRS99]. Finally, through the notion of *Inter-agent Relationship* IAR (which consists on grouping agents in cooperative clusters) [TSS05], the authors give an interesting example of how autonomous agent configuration can bring more adaptability to context-aware services.

3.5 Context awareness

By using these capabilities, mentioned above, agents can achieve better context awareness. Stu21 [SC06] shows a good example of how an agent can be aware of the context. The Stu21's *context agent* is able to keep a model of context, reason over its own context to deduce beliefs and goals, communicate changes in its own context to other agents, listen to context changes in pees as input to its own decision-making, and initiate, execute, and terminate behaviours in response to context changes. Another example is given in [KK05] where agents are able to achieve context representation unification, context composition, inference and dissemination, context-sensitive communication protocol, and context management.

3.6 Mobility

Usefulness of mobile agents has been debated since early 90s. After a fall of interest in the last decade, the interest to mobile agents is again on the rise [Zas04]. This is motivated by advances in enabling technologies, including wireless networks, diverse devices, portable platforms, sensor networks.

Code mobility can improve the traditional client/server paradigm along two different and orthogonal lines [BBC+04]: (a) allowing the dynamic decision of where the service should be located; (b) allowing the dynamic decision of which host provides the needed resources. In addition, mobile agents benefit from the additional flexibility of moving code together with the state modified during the already performed computation. Mobile agents offer several advantages to context-aware middlewares. Indeed, they:

- reduce communication: as a result of today's wireless telecommunication networks charging for the connection time, mobile users try to minimize the connection time to the network. To enable a mobile user to work with a device while not being connected to the network, one approach consists on decoupling the mobile computer and its applications from the network [KRS99]. Here, agent systems are very well suited. The mobile agent paradigm does not need continuous network connectivity because connections are required only for the time needed to inject agents from mobile terminals to the fixed network infrastructure. Since these agents are also autonomous, they can carry on services even when their users are disconnected and deliver service results back at their reconnection.

- are easily updated: a code can be located (and updated) in any remote server. Mobile agents can then load this code and get automatically updated to newer versions. AMASE uses these concepts [KRS99].

- take into account the network's QoS: depending on the available bandwidth, agents may decide to migrate to another host or to stay at their host site. For example, the migration of an agent across a low-bandwidth GSM connection requires considerable time and therefore cost. As a result, an agent can delay its migration until a faster connection becomes available.

- reduce complexity: A virtual pervasive world is characterized by large number of devices with limited resources and software services. A user's request may need to navi-

gate through complex heterogeneous networks, often crossing boundaries of multiple distributed systems. Mobile agents can make use of dynamically available resources and make decisions according to changing situations, which reduces the complexity of supporting context-aware services. Mobile agents use standards (e.g., RDF/OWL Ontology, FIPA message protocol) in order to cope with the heterogeneity of distributed systems.

- guarantee more adaptability: being both intelligent (autonomous, reactive, and pro-active) and mobile, allows agents to perform adaptation in pervasive systems. Their dynamic behaviour (based on up-to-date information that could not have been otherwise acquired in a pre-planned or more static manner) offers a better context-aware service [Zas04].

4 Challenges

The agent-based technology has also some drawbacks, often identified in the associated security issues (especially for mobile-agent-based systems). These potential weaknesses have challenged researchers to investigate and provide rich mechanisms, tools and strategies for security. However, according to our survey, most of existing context-aware middlewares seem not to care a lot about this aspect. Besides, none of the studied middlewares has addressed the Quality of Context (QoC). This section formulates these two drawbacks as future challenges for agent-based context-aware middlewares. In addition and as previously mentioned, our survey shows that most existing middlewares do not take advantage of the learning capabilities of agents. We thus suggest the integration of this propriety in order to build more efficient context-aware agents.

4.1 Security and privacy

Security is an important issue in agent-based applications since code and data can move between different systems. Besides, the agent-based computing itself must be secured. This involves protecting agents against code modifications and malicious agents. Moreover, privacy of data must be guaranteed. Agents carrying this data have to be protected against unauthorized reading.

Current agent-based applications already support some security and privacy features. For instance, some middlewares presented in our state-of-the-art address these requirements:

- in security: few architectures include security features preventing unauthorized service usage and prohibiting agents from attacking other agents or platforms. For example, AMASE [KRS99] contains security features for user authentication, agent signing, and access control. When the user sends an agent, constant parts of the agent are signed using the private key of the user. The agent system of course needs the public key of the user.

- in privacy: in [WCRS04], privacy aspects regarding personal information are addressed by encapsulating sensitive information within dedicated personal user agents. In AMASE [KRS99], agents support access control lists. A service agent can create access control lists and define which rights a user will be granted.

Nevertheless, most of these security and privacy features (except in AMASE) are offered by the tools of implementation (e.g., JADE), which support generic security features. Security and

privacy should rather be more considered. Agent-based middlewares should propose security and privacy features which are more suitable for context awareness, instead of only relying on security facilities offered by agent-based environments.

4.2 Quality of Context

Over the past decade, many context-aware applications have been built. However, few of them have been deployed in real life. According to [BGT+06], the quality of context is one main reason. QoC is either ignored or not well addressed in the existing context-aware systems. Agent-based middlewares do not make exception. The authors also claim that systems are often unable to guarantee an acceptable QoC because of two main reasons:

- Different sensors may produce different sensed data values which will lead to the inconsistency of sensor-based applications.
- Most sensors usually send sensed data periodically so that it is very difficult for computers to know what happens in the time interval between two sensor signals.

Nevertheless, multi-agent system paradigm seems to be well suited to solve, at least, the first problem. Indeed, agents can take advantage of their capabilities of negotiation, communication, cooperation, learning, and reasoning in order to solve inconsistence conflicts. As for the second problem, agents can also be useful. For example, they may learn which sensors are most likely to sense a change between two successive signals and then adapt their behaviour toward these sensors (for example, by imposing a more frequent monitoring). They may also reason about uncertain contexts and use an advanced learning mechanism in order to predict context changes. Since such processes may require more intelligence (which suggests more complexity), a distributed solution (multi-agent system) is inevitable. Finally, and given a set of QoC measurements, agents can learn (from these measurements) in a distributed way how the context quality is evolving and then make appropriate decisions in order to enhance the QoC (since they are autonomous and able to reason).

4.3 Learning

Learning is one important property of agents. However, according to our benchmark (no "Adaptive" instance mentioned in the last column of Table 1), most agent-based middlewares do not integrate learning capabilities to their agents (except Stu21 [SC06] which implements a simple learning mechanism dedicated to a very specific task), and yet it could be very useful. Indeed, learning can be used by agents to: (i) improve the QoC (see the previous section), (ii) make applications more adaptable to users' needs, (iii) make predictions about context and anticipate potential problems, and (iv) address context conflicts in a more intelligent way.

5 Conclusion and Future Work

In this paper we made a survey on existing agent-based context-aware middlewares. We classified them according to their conformity to the agent paradigm. We thus noticed that in most proposed architectures, the use of agent-based approach is merely an implementation choice. Other

sorts of distributed systems also could have been used as well. Nevertheless, others middlewares take more advantage of the agent paradigm. Then we refined the classification according to 8 different purposes of using agents (implementation, reasoning, context awareness, QoS, Security/privacy, connectivity, service customization and adaptability, and negotiation). We compared the targeted middlewares according to these criteria. Based on this caparison, three more conclusions were reached: (a) the agent paradigm offers a significant support for context-aware middlewares in different areas (context awareness, reasoning, negotiation, customization and adaptability for services, and network connectivity); (b) aspects like Security/Privacy and Context QoS are less considered; (c) one capability of agents (adaptability, which includes learning) is hardly used. Given these observations, we enumerated several advantages of using agents in context-aware middlewares and select illustrative examples from the studied systems. We also pointed to the weakness of existing systems and identify three challenges to be addressed (security/privacy, QoC, and learning) by such systems.

In conclusion, agents seem to be very suitable to fulfill the main requirements [WCRS04] (modularity, scalability, adaptability, and distributedness) of middlewares for context awareness. However, the achievement of all these requirements will depend on the use of the agent paradigm as a central paradigm in the considered systems.

This paper was motivated by our research group's current work on a middleware for context awareness called Context Management Framework (CMF) [KBIP06], part of the Dutch project program Freeband. Even if CMF (which is already implemented and successfully tested) is highly distributed, it does not rely on an agent-based architecture. We thus made this survey to see if agents could be a value-added for CMF. Indeed, our study shows that an agent-based approach, apart of being an implementation choice, can bring other benefits if the agent paradigm is fully used. We expect that the "agentification" of CMF, which has a strong focus on the QoC, would enhance the adaptability of context-aware services.

Bibliography

[BBC+04] P. Bellavista, D. Bottazzi, A. Corradi, R. Montanari, S. Vecchi. *Mobile Agent Middlewares for Context-aware Applications*. CRC Press, 2004.

[BGT+06] Y. Bu, T. Gu, X. Tao, J. Li, S. Chen, J. Lu. Managing Quality of Context in Pervasive Computing. In *Proc. of the 6th Int. Conf. on Quality Software*. Pp. 193–200. IEEE, 2006.

[Che04] H. Chen. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD dissertation, University of Maryland, 2004.

[CTS+01] H. Chen, S. Tolia, C. Sayers, T. Finin, A. Joshi. Creating Context-Aware Software Agents. *First GSFC/JPL Workshop on Radical Agent Concepts*, September 2001.

[Gai] Gaia Project http://gaia.cs.uiuc.edu/ last visited Sept. 2007.

[HK05] H. Harroud, A. Karmouch. A Policy Based Context-Aware Agent Framework to Support Users Mobility. In *Proc. of AICT-SAPIR-ELETE*. Pp. 177–182. IEEE, 2005.

[KBIP06] H. van Kranenburg, M. Bargh, S. Iacob, A. Peddemors. A Context Management Framework for Supporting Context-Aware Distributed Applications. *IEEE Communications Magazine*, pp. 767–74, August 2006.

[KK05] M. Khedr, A. Karmouch. ACAI: agent-based context-aware infrastructure for spontaneous applications. *J. Netw. Comput. Appl.* 28(1):19–44, 2005.

[KRS99] E. Kovacs, K. Röhrle, B. Schiemann. Adaptive Mobile Access to Context-Aware Services. In *Proc. of the 1st Int. Symposium ASAMA*. P. 190. IEEE, 1999.

[Mae94] P. Maes. Agents that reduce work and information overload. *Commun. ACM* 37(7):30–40, 1994.

[PL07] R. Plesa, L. Logrippo. An Agent-Based Architecture for Context-Aware Communication. In *Proc. of the 21st Int. Conf. on Advanced Information Networking and Applications Workshops*. Pp. 133–138. IEEE, 2007.

[QSS06] W. Qin, Y. Suo, Y. Shi. CAMPS: A Middleware for Providing Context-Aware Services for Smart Space. LNCS 3947, pp. 644–653. Springer, 2006.

[SC06] A. Singh, M. Conway. Survey of Context aware frameworks - Analysis and Criticism. Technical report, University of North Carolina, 2006.

[SDA99] D. Salber, A. K. Dey, G. D. Abowd. The context toolkit: aiding the development of context-enabled applications. In *Proc. of the SIGCHI conf. on Human factors in computing systems*. Pp. 434–441. ACM, 1999.

[SDSP07] J. Soldatos, N. Dimakis, K. Stamatis, L. Polymenakos. A breadboard architecture for pervasive context-aware services in smart spaces: middleware components and prototype applications. *Personal Ubiquitous Comput.* 11(3):193–212, 2007.

[TSS05] H. Takahashi, T. Suganuma, N. Shiratori. AMUSE: An Agent-based Middleware for Context-aware Ubiquitous Services. In *Proc. of the 11th Int. Conf. on Parallel and Distributed Systems*. Pp. 743–749. IEEE, 2005.

[WCRS04] J. Wohltorf, R. Cissee, A. Rieger, H. Scheunemann. BerlinTainment: An Agent-Based Serviceware Framework for Context-Aware Services. In *1st Int. Symposium on Wireless Communication Systems*. 2004.

[WW07] C. Wang, X. Wang. Multi-agent Based Architecture of Context-aware Systems. In *Int. Conf. on Multimedia and Ubiquitous Engineering*. Pp. 615–619. 2007.

[Zas04] A. Zaslavsky. Mobile Agents: Can They Assist with Context Awareness? In *Proc. of IEEE Int. Conf. on Mobile Data Management*. Pp. 304–305. IEEE, 2004.

[ZWH+06] D. Zheng, J. Wang, W. Han, Y. Jia, P. Zou. Towards A Context-Aware Middleware for Deploying Component-Based Applications in Pervasive Computing. In *Proc. of the 5th Int. Conf. on Grid and Cooperative Computing*. Pp. 454–457. IEEE, 2006.