Capturing Context Information in a Context Aware Virtual Environment

Helio H. L. C. Monte-Alto and Elisa H. M. Huzita
State University of Maringá, Informatics Department, Maringá, Paraná, Brazil

Keywords: Virtual Environment, Context Awareness, Collaborative Work, Multi-agent Systems.

Abstract: Designing context aware applications is a great challenge given the complexity of such systems, specially concerning the mechanisms to provide sensing, or capturing, of context information. There are many works in literature toward providing context awareness in physical environments, such as pervasive systems. However, when dealing with virtual environments, such as distributed environments to support collaboration for teams distributed geographically, an increase on complexity of its design comes up. There is a need to model a software system which uses concepts present in physical environments in order to try to reduce the disadvantages of the distribution. This work explores this challenge, focusing on conceiving a solution for context awareness in distributed virtual environments. We also present a model for designing a platform to support the development of multi-agent based context aware virtual environments.

1 INTRODUCTION

Context awareness has been extensively used in distributed systems, specially pervasive systems to provide smart physical spaces. A context aware environment should be able to allow perception of it and autonomously execute actions in order to optimize its operations, its behaviours and its interactions with the user (Viterbo et al., 2009). However, there are other applications of context awareness beyond physical environments. There is a growing need for virtual environments, specially to support collaborative work for users geographically distant from each other. In such environments, context perception is not as straight forward as in physical environments, in which context is captured through physical sensors. In distributed and virtual environments, such perception must be done exclusively by software entities, requiring an appropriate software architecture to support context awareness.

In order to implement context aware systems, one of the most suitable approaches is by means of multi-agent systems, given that an agent is a computational entity which is autonomous, situated and able to sense in an environment (Wooldridge, 2002). Agents also may interact with other agents, making it appropriate for collaborative environments.

Once understood such ideas for context awareness, it has been proposed a multi-agent infrastructure to support the development of context aware systems for virtual environments called CAKMAS (Context Awareness and Knowledge-Based Multi-Agent Systems) (Monte-Alto et al., 2013). In its current architecture, it proposes solutions for persistence and dissemination, although its aim is to provide much wider support for context awareness. One of the key issues to be addressed in context aware environments is the need of capturing context information. This paper presents a solution for this problem, extending CAKMAS, and focusing on mapping the concepts of a physical environment and using them to design a virtual multi-agent environment where agents are able to sense what is happening in order to take decisions and react accordingly.

This paper is divided in six more sections. Section 2 presents some works towards context aware and multi-agent architectures and their flaws regarding context capturing in virtual environments. Section 3 presents the conceptualization to help figuring out how should a virtual environment be designed in order to properly support context awareness. Section 4 presents our proposal for a multi-agent platform to support virtual environment with context awareness capabilities. In Section 5 it is presented a case study to demonstrate the functionality and effectiveness of the proposed architecture. Finally, conclusions and future works are discussed.

H. L. C. Monte-Alto H. and H. M. Huzita E..
Capturing Context Information in a Context Aware Virtual Environment.
DOI: 10.5220/0004885406050612
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 605-612
ISBN: 978-989-758-028-4

2 RELATED WORKS

One of the most adopted approaches for context aware systems is the middleware infrastructure, which introduces a layered architecture for context aware systems with the intention of hiding low-level sensing details (Chen, 2004; Baldauf et al., 2007). In order to provide middleware solutions, it has been proposed agent-based systems as the approach for implementing these systems. Such choice gained popularity due to agent's characteristics, which brings decentralization and facilitates reasoning and sharing of context information.

A multi-agent system requires that the agents have an environment in which they live, act and interact. Many context aware multi-agent systems architectures have been proposed in the literature thus far. However, most works focus on physical environments instead of virtual environments, and how to provide context awareness in pervasive systems. There is a lack of works focused on exploiting mechanisms to allow software agents to sense what is happening in a virtual environment, as well as more intuitively designing virtual environments based on the concepts of physical environments, as presented in Section 3.

The ACAI Framework, presented by Khedr and Karmouch (2005), proposes agents that accomplish several requirements of context awareness, such as context composition, inferring, and inter-domain context invocation. This project focuses on issues of context representation and reasoning in agent systems for pervasive environments.

CoBrA (Chen, 2004) is an agent-based architecture to support context aware system in smart spaces - i.e. physical and pervasive environments. Its architecture focuses on a special agent called Context Broker, which maintains a shared model of the context of the environment, providing it for a community of agents, services and devices. As it focuses on capturing context from physical sensors, there is no concern about sensing context in virtual environments.

CAKMAS is an architecture whose purpose is to support knowledge-based and context-aware multi-agent systems, specially for collaborative virtual environments. It is based on the DiSEN-CSE model, which defines a whole cycle for context-awareness, including capturing, processing, disseminating and persisting context information. The current status of this project focuses on persistence and dissemination of context. This paper focuses on extending CAKMAS to support context capturing / sensing properly.

There are also many works focused on virtual environments in the area of Virtual Reality. However, they usually deal with immersive digital environments or virtual spaces, i.e., 3D environments which try to simulate physical environments (Churchill and Snowdon, 1998). Such works are concerned with issues related with sensing similar to physical environments, such as distance and collision of virtual objects. This is not the aim of our work, which proposes a distributed software system which inherits some characteristics of physical environments in order to improve awareness and collaboration for a distributed team. Further detail will be covered in Section 3.

3 CONCEPTUALIZATION

3.1 Designing a Virtual Multi-agent Environment

The term "Virtual Environment" is increasingly being part of the life of most people. Everyone is connected, and the Web has become an extension of the physical environment for many. Although the term Virtual Environment is often used for any software system in which the user interacts, there are some more specific definitions, specially in the area of Virtual Reality (VR). According to Churchill and Snowdon (1998), a Collaborative Virtual Environment (CVE) is a computer-based, distributed, virtual space or set of places in which people can meet and interact with others, with agents or with virtual objects.

The idea of virtual environments are very widespread in the area of collaborative working as a way of supporting awareness, coordination and communication in a distributed team (Benford et al., 2001; Churchill and Snowdon, 1998). Therefore, such concept is very important for this work since it defines the way the users will interact to it and also with other users through the environment.

In order to achieve a suitable virtual environment to support collaboration in a distributed team, there is nothing better than basing it on concepts involved in a physical environment. Although our aim is not to conceive a virtual environment which totally simulates a physical environment, we can take some characteristics and concepts from real physical environment in order to design a suitable virtual environment for collaboration, as it is going to be exposed as follows.

3.2 Differences Between Physical and Virtual Distributed Environments

In order to capture good ideas regarding how to design a virtual environment, it is interesting to try to highlight differences between a physical environment

and an hypothetical virtual environment. To accomplish this, it is necessary to take an overview of both environments, as it is exposed on Figures 1 and 2.

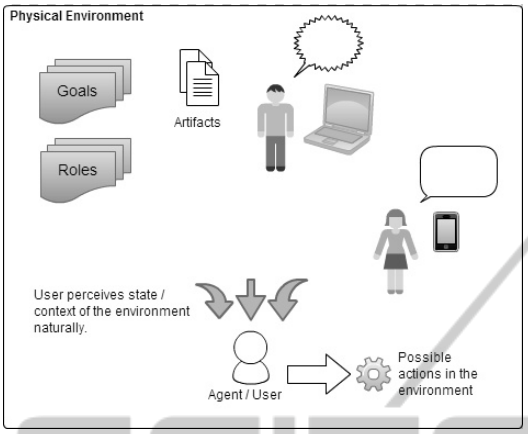

Figure 1: Physical environment.

In physical environment, an individual (or agent) interacts with physical objects and resources that are present in the environment, and also interacts with other individuals. All these interactions occurring in the environment are naturally and cognitively perceived by other individuals in the environment, based on their **interest** and **proximity** in the event. In summary, it is possible to highlight three main characteristics of such environment:

1. An agent **acts** in the environment by interacting with its objects and resources. It does that by means of actuators provided by the environment and linked to these objects and resources. Example: an agent is able to open a door because it has a knob which allows such action.
2. The agent is also able to **interact (or socialize)** with other agents in the environment by some means of communication, such as spoken conversation and gestures.
3. The agents can perceive what other agents are doing in the environment. This usually happens when they are interested in what the other agent is doing, or what is going on with it. The interest may be focused on the agent in particular, in the action the agent is taking, or both. In other terms, we can say that an agent intercepts the interactions of one or more different agents in order to gather useful information earlier. This also characterize the concept of **awareness**.

Summarizing, we have three main characteristics of an environment: (i) action, (ii) socialization and (iii) awareness. The great challenge is how to design and implement such concepts in a distributed and virtual environment. Figure 2 shows an overview of a distributed environment and its concerns about these issues.

One very important concept that arises from all this scenario is **context**. An agent needs to be aware of the context of the environment in order to facilitate seeking its goals. The way the context is captured in a physical environment is natural and sensed by human cognition. As well as a human agent is capable of hearing and seeing what interesting events are happening around him/her, software agents must be able to capture interesting information from the environment. They must also be capable of processing the information captured from the environment in order to turn it in reasonable information - or knowledge - because conclusions or new information can be inferred from it.

This work is specially concerned in capturing context information from the environment, providing proper awareness for the agents in a distributed and virtual environment.

4 CAKMAS PLATFORM

4.1 Overview

Based on CAKMAS architecture (Monte-Alto et al., 2013), we propose a FIPA-compliant Agent Platform (AP) (FIPA, 2000), extending CAKMAS toward a broader context aware supporting, taking into consideration all the conceptualization presented in Section 3. Such platform architecture is going to be called CAKMAS Platform. In this paper, we will focus on the ideas and mechanisms to support awareness in such platform. In other words, we present solutions for the problem of agents perceiving events in the environment according to their interest.

An overview of CAKMAS Platform can be seen at Figure 3. It illustrates the agents involved in the environment, including specialized agents of the platform, responsible for managing the platform and providing context awareness facilities. The platform is composed of the following specialized agents:

- Manager Agent (MA): a FIPA based agent that exerts supervisory control over access to the AP. It also maintains a directory of registered agents identifiers. Each agent must register with the MA.
- Directory Agent (DA): another FIPA special agent that provides yellow pages services for the agents. Environment services must be registered in this agent in order to provide description and discovery for them. Agents also may register their services with the DA or query the DA to find out what

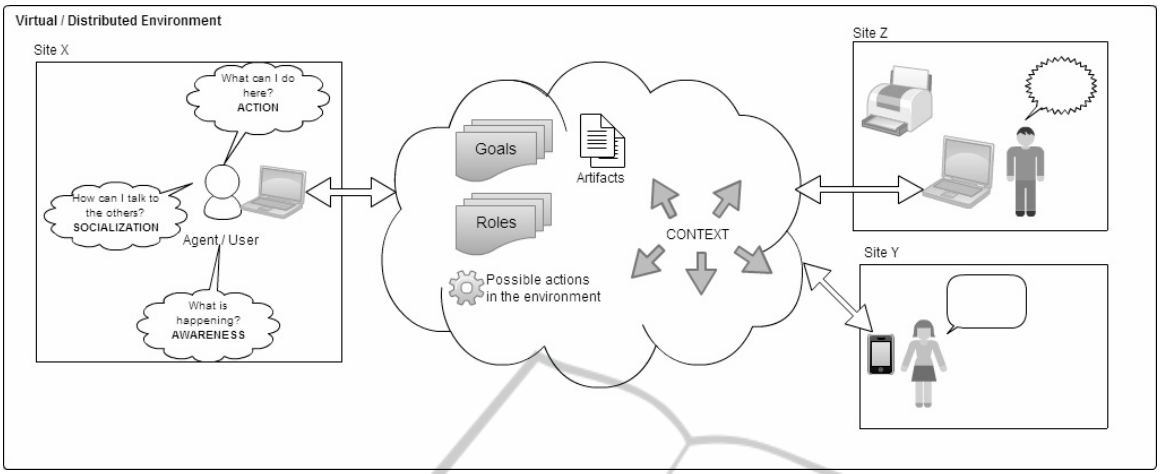

Figure 2: Virtual environment.

services are offered by other agents.

- Service Broker Agent (SBA): Inspired by a similar solution proposed by Tapia et al. (2008), this agent is responsible for managing incoming requests from the user to be processed by services. It basically invokes the service and returns the response back to the users, ensuring that every action from the user in the environment passes through the AP, which allows context capturing by the agents.
- Ontology Agent (OA): the agent responsible for persisting and disseminating context information among the agents in the environment (Monte-Alto et al., 2013).

The remaining agents are application specific agents. Every agent in the platform is called a CAKMAS Agent, and inherits some special features that must be available in order to support context awareness, as explained below.

4.2 Agent Internal Architecture

The internal architecture of a CAKMAS Agent is shown on Figure 4. Such architecture is characterized by the BDI (Belief-Desires-Intentions) model and four stages which define its behaviour and life cycle. This four-stages behaviour is compliant to the architecture of SemantiCore, an AP for Semantic Web applications (Blois et al., 2007). It also matches the essence of agents, which must sense the environment by sensors, make some decisions based on its knowledge and act accordingly through actuators (Russell and Norvig, 2003).

The **belief** of an agent in CAKMAS is represented as knowledge about the environment - including the context of the environment - and related application domain, as well as the rules involved. Such knowledge is described and implemented by means of ontologies and the rules are implemented as inference rules, using specially Web Semantic standards and technologies such as OWL, RDF, SPARQL and SWRL (W3C, 2001). It is also by means of ontologies that context information (or context knowledge, given that the context is represented semantically) is represented in our architecture, as explained in (Monte-Alto et al., 2013). This provides a uniformity in the way context knowledge is shared and used for reasoning.

The **desires** of an agent are the goals it wants to achieve. A goal can be defined as a state, or a set of facts that the agent wants to be true in the environment. An agent knows when a goal is achieved when its knowledge about the environment matches the description of the goal. When the agent decides, based on its goals, to commit with an action plan, an **intention** is created. In other terms, the intentions of an agent are comprised of action plans that it decided to take in order to reach its goals.

Goals also define the **interests** of an agent. An agent may be interested only on information that can help it to reach its goals, and such information is captured by the agent by means of its sensors. This way, the desire of an agent defines which kind of context information it wants to be aware of.

The four-stages model of the agent behaviour is also totally focused on the BDI model. A **sensor** captures messages from the environment and filters them according to the interest, defined based on the desires, of the agent.

The **decisory** component is the responsible for reasoning about the facts coming from the environment over its own knowledge (belief), aiming to achieve its goals. In general, such decisions are made by means of running an inference engine with infer-

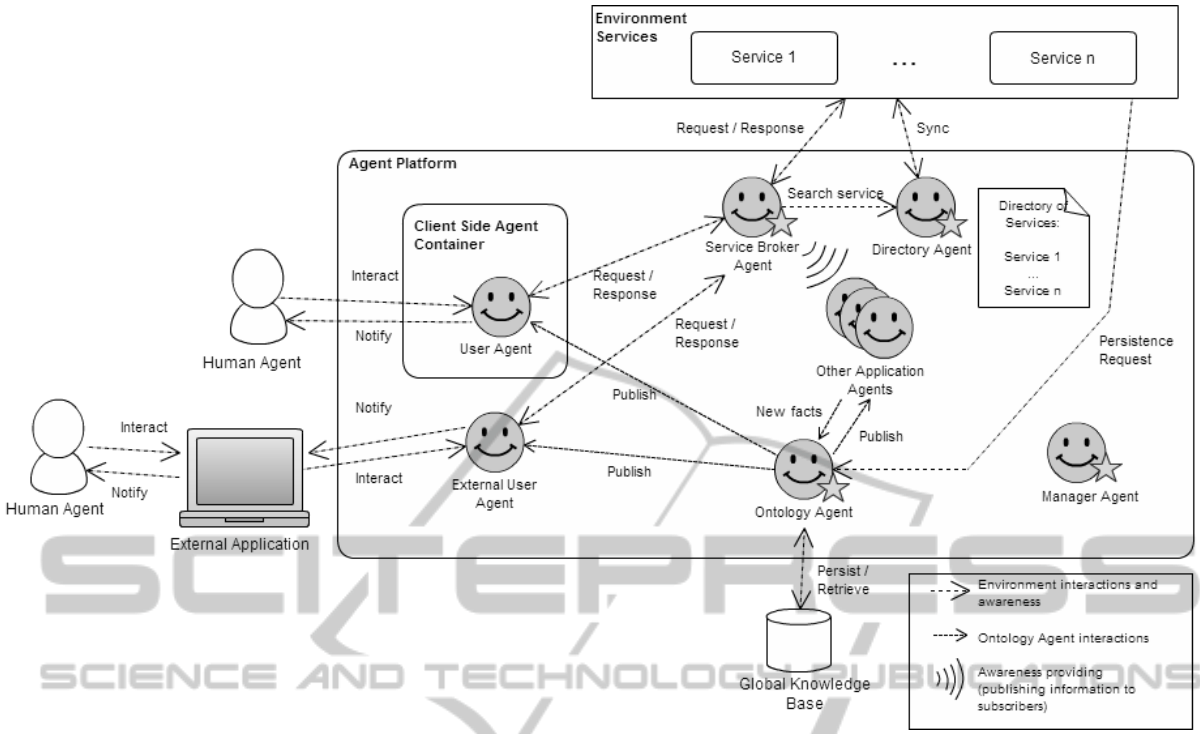

Figure 3: CAKMAS Platform.

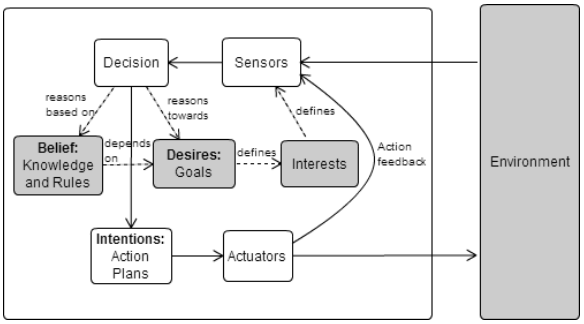

Figure 4: Internal Architecture of a CAKMAS Agent.

ences rules over the current knowledge the agent has about the context of the environment. These decisions should result in the execution of actions to react accordingly to these events and also pro-actively try to achieve its goals by acting in the environment and interacting with other agents collaboratively.

The **execution** component use the concept of *workflow* to execute actions in specific orders based on the decisions taken by the decisory component. The actions can also use specific **actuators** to act in the environment - e.g. the service to send an e-mail requires a specific actuator to invoke this service. Given that many agents may have to execute similar actions and use the same actuators, such architecture allows reusing of actions and actuators.

Regarding context capturing support, CAKMAS Agents use the pattern *Observer* to get informed about events which they may be interested. Each agent has a list of subscribers, which are the agents that are interested in the actions of this agent. Whenever the agent acts in the environment, usually by means of message exchanges, each one of its subscribers also receives the message. In other words, the subscribers intercept the message of the publisher, becoming aware about the occurrence.

4.3 Service Broker Agent

The Service Broker Agent (SBA) is responsible for providing an interface between the agents of the platform - usually User Agents - and the services provided in the environment by managing service requests and responses. It was based on the FUSION@ architecture proposed by Tapia et al. (2008), and its functionality is detailed in Figure 5 and described as follows:

1. An agent sends to SBA a request for a service available in the environment, probably discovered via DA;
2. SBA receives the request via its sensor and pass the message to its decisory component;
3. The decisory component makes some security checks by asking DA whether the service really

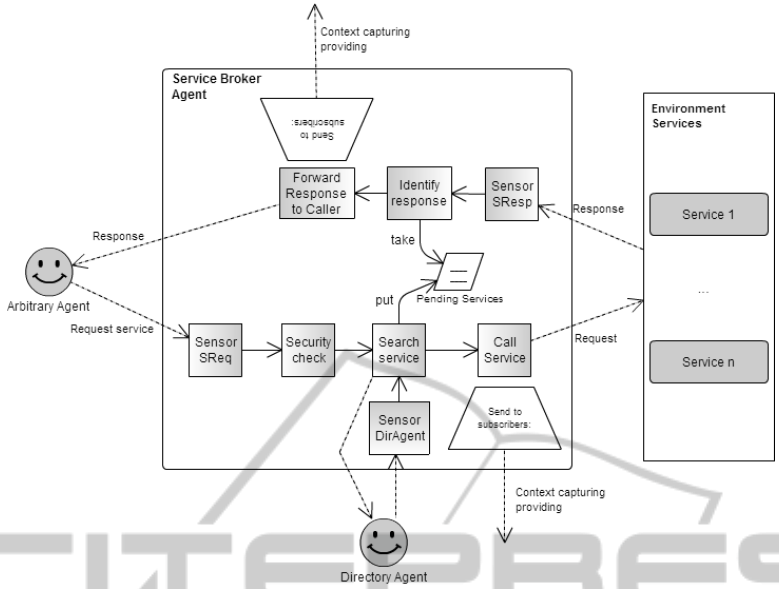

Figure 5: Service Broker Agent behaviour.

exists and if its request message is semantically and syntactically correct.

4. Following the decision to continue the process, some actuators are called. One of them invokes the service properly. The other sends the information about the service request to every subscriber of SBA, which is the key to provide awareness in the environment.
5. SBA receives a response from the service invoked via an appropriate sensor, and send it to the decisory component;
6. The service response is identified and packed in order to send the response to the requester. It also deregisters the service as being done.
7. The agent sends the response for the requester agent and also for every subscriber.

Given this process, we have a combination of the *Observer* approach and the *broker* architectural model to provide awareness for the agents in the virtual environment.

5 EMPIRICAL STUDY

In complex and distributed systems it is often not straight forward to figure out every detail of its infrastructure. CAKMAS Platform provides such kind of infrastructure, thus it is expected that uncertainty during the modelling of such architecture comes up. It is necessary some empirical study to ensure that every detail has been covered and make sure that it is effective. To accomplish this, a case study was carried on, aiming to give strength to our proposal and also expanding the sight to a particular application. A prototype of the scenario was also implemented in order to allow an experimental study and to catch a glimpse of the improvements introduced by CAKMAS Platform.

The scenario used for this study is the same application which is the focus of research of our research group: a Global Software Engineering environment, called DiSEN. Part of its architecture aims to provide context awareness to such environment by means of some components such as the Middleware, the Agent Manager and the Object / Service Manager.

The idea of this case study is to map DiSEN's components to CAKMAS Platform's elements, as if it was the base for DiSEN's context awareness features' implementation. The scenario, already presenting such mapping, is represented in Figure 6, and described as follows.

5.1 Preconditions

Developer Bob is member of a team which is working on project DiZEN (a fictional and contradictory project) and whose project manager is Alice. Bob has been allocated to work on a task T with a certain priority P1. Therefore, his personal agent BobAgent is interested in being aware when there is a change in T's priority. Alice also has a personal agent called AliceAgent, which may act in the environment on her behalf.

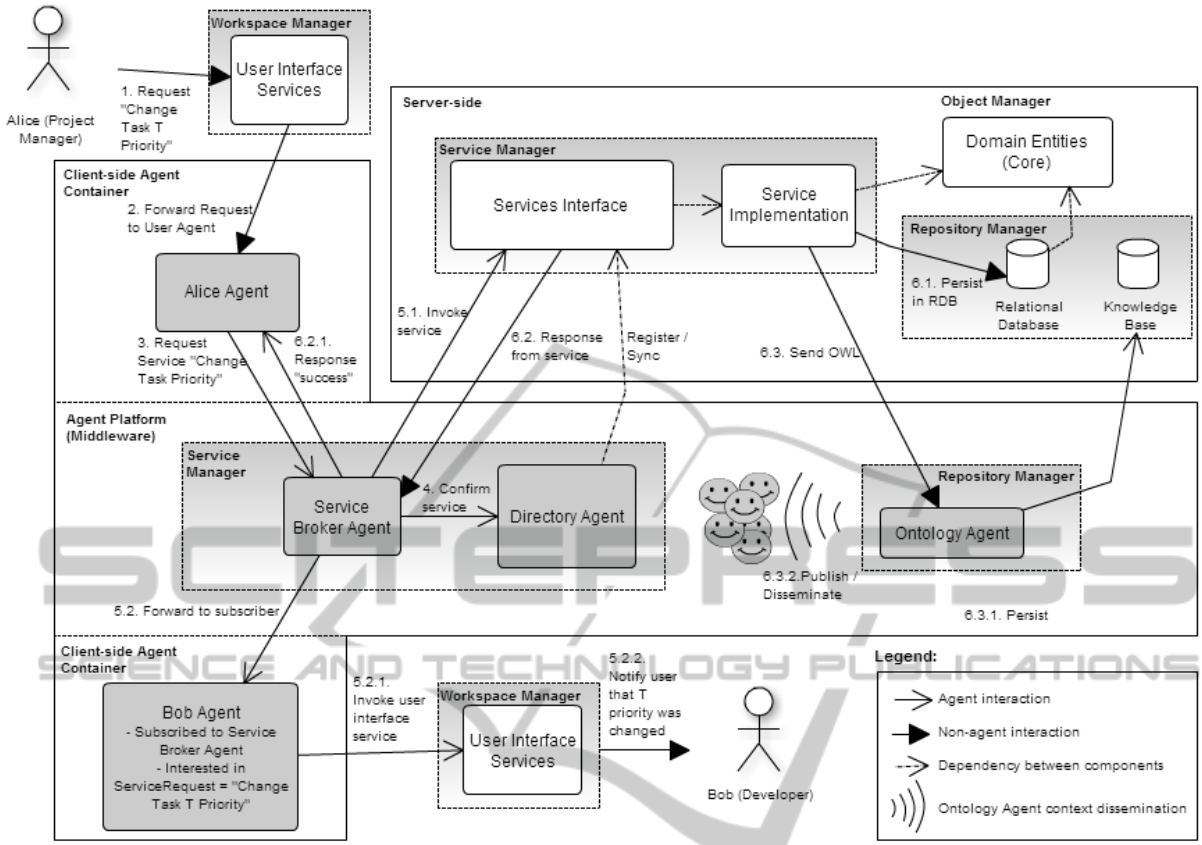

Figure 6: DiSEN with CAKMAS scenario.

5.2 Description

The following is the exact description of the interactions that take place in our scenario:

1. Alice, the project manager, interacts with the environment and asks its personal agent AliceAgent to change task T priority;
2. AliceAgent receives this request and looks for the adequate service.
3. AliceAgent creates a service request following a specified standard (e.g. SOAP) and sends it to SBA;
4. SBA takes the required actions as described in Section 4.3, and then invokes the service requested. It also publishes the request to all subscribers, which includes BobAgent;
5. BobAgent receives the message containing the service request from AliceAgent, becoming aware of Alice's wish to change T priority. It knows that the message is interesting based on its interest, or desire, as explained in Section 4.2;
6. BobAgent notifies developer Bob about the priority change properly. Such action - the notification - is triggered when its decisory component reason about the fact that one of the tasks Bob is working had its priority changed;
7. While Bob becomes aware of Alice's action, the request is processed by the service provider, i.e., the service implementation. The context knowledge is also generated and sent to the OA, which will check consistency, persist and disseminate it. In case of success, SBA can confirm the action to BobAgent. Such confirmation may also end up being made by the OA, which will disseminate the new context information later. Otherwise, SBA can inform BobAgent that the operation triggered by AliceAgent was not successful and advice Bob properly about it.

5.3 Post-condition

The project manager Alice made his priority change of T successfully and the developer Bob could be aware of this context change immediately.

5.4 Implementation and Experimental Study

The given scenario, as well as a preliminary version of CAKMAS Platform, was implemented in order to foresee the improvements of using the proposed approaches for context capturing. As a continuation of the previous CAKMAS implementation, it was implemented in Java, based on the framework SemantiCore, using Jena to handle ontologies.

The scenario was run with and without the mechanism introduced in CAKMAS Platform. In the case without the SBA, BobAgent gets informed when the OA disseminates the new context in the environment, while it persists it in the knowledge base. Each case was run 40 times to take averages. The detailed experimental results and analysis are out of scope of this paper given the paper length limitation. However, just to give a glimpse of the results, the average time BobAgent took to be aware of Alice's action without the SBA was 636,03 milliseconds, whereas the same with SBA was 58,29 milliseconds. The improvement in the time to be aware of context is significant for the scenario, and we can deduce that it must reflect in a real environment.

6 CONCLUSIONS AND FUTURE WORKS

Proposing solutions for context aware virtual environments is justified by the increasing demand for tools to support collaboration among people geographically distributed. In such cases, the lack of a common environment in which people can act, interact and become aware of the situation demands some kind of software system to improve coordination and awareness. CAKMAS Platform is proposed as an extension of CAKMAS architecture taking into account the concept of environments, agent platforms and context capturing. As presented in Monte-Alto et al. (2013), previous work has focused on reasoning, persistence and dissemination of context information, but lacked proper ways of capturing it.

The proposed solution was analysed and validated by means of a case study and an experimental study using a prototype. Such empirical studies allowed us to figure out in more detail the functionality and effectiveness of CAKMAS Platform. However, further validation including a more robust implementation and experimental study is necessary to give strength to our proposal, which will be covered in future works.

REFERENCES

Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277.

Benford, S., Greenhalgh, C., Rodden, T., and Pycock, J. (2001). Collaborative virtual environments. *Communications of the ACM*, 44(7):79 85.

Blois, M., Escobar, M., and Choren, R. (2007). Using agents and ontologies for application development on the semantic web. *J. Braz. Comp. Soc.*, 13(2):35–44.

Chen, H. (2004). *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD thesis, University of Maryland, Baltimore County.

Churchill, E. F. and Snowdon, D. (1998). Collaborative virtual environments: an introductory review of issues and systems. *Virtual Reality*, 3(1):3–15.

FIPA (2000). FIPA agent management specification. Specification 23, FIPA. http://www.fipa.org/specs/fipa00023/SC00023K.pdf.

Khedr, M. and Karmouch, A. (2005). Acai: agent-based context-aware infrastructure for spontaneous applications. *J. Network and Computer Applications*, 28(1):19–44.

Monte-Alto, H. H. L. C., Biasão, A. B., Teixeira, L. O., and Huzita, E. H. M. (2013). Multi-agent and context-aware solutions for a global software development environment. *International Journal of Artificial Intelligence*, 11(A13):115–129.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education Inc, Upper Saddle River, NJ, 2 edition.

Tapia, D. I., Rodríguez, S., Bajo, J., and Corchado, J. M. (2008). Fusion@, a soa-based multi-agent architecture. In *DCAI*, volume 50 of *Advances in Soft Computing*, pages 99–107. Springer.

Viterbo, J., Mazuel, L., Charif, Y., Endler, M., Sabouret, N., Breitman, K., Seghrouchni, A. E. F., and Briot, J.-P. (2009). Managing distributed and heterogeneous context for ambient intelligence. *Context-Aware Self Managing Systems, CRC Studies in Informatics Series*, pages 79–128.

W3C (2001). W3c semantic web.

Wooldridge, M. (2002). *Introduction to MultiAgent Systems*. John Wiley and Sons.