Context management in an agent-based approach for service assistance in the domain of consumer electronics

Jürgen Schirmer[1], *Holger Bach*[2]
[1]*Computer Graphics Center, Darmstadt,*
[2]*Technical University Darmstadt, Interactive Graphics Systems Group*

Keywords

Consumer electronics, context, context-aware computing, situated computing, situation-aware computing, situation awareness, agent, multi-agent system, agent communication language, KQML, XML.

Abstract

In this article, we will give an overview of a context management facility that provides the required knowledge base and event mechanism to enable situation-aware assistance in the domain of consumer electronics. The automatic acquisition of information about the ecological and technical environment, as well as the »working« situation and preferences of the end-user, together establish the usage context and play an important role for a personalized, efficient information access and pro-active service offering. We propose an interface and architecture of a context management component (CONJURER)*, which is integrated into a multi-agent system. The context manager's main task is to provide the current usage context. Agents in the system query or change the context using appropriate KQML messages.*

1 Introduction

The growing intricacy and diversity of today's electronic world in regard to consumer electronics, light weight devices for personal information management, networked home and automotive environments, as well as terminal systems on the one hand, and the information and service overload on the Internet and Intranet on the other, require the end-user to have a more sophisticated form of electronic assistance in his »working« situation.

Taking the specific end-user situation into account, an electronic assistance system can anticipate the current information and service requirements of the end-user and provide an instant, efficient and pro-active information and service delivery.

What constitutes the »working« situation of the end-user? An overview of the end-user's situation can best be described using an extended context model[1] (see Figure 1).

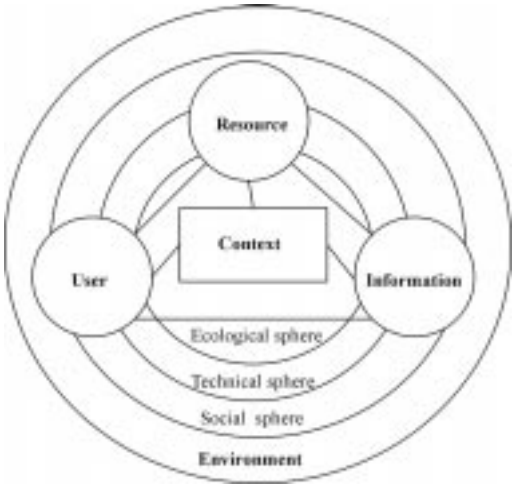

Figure 1: Enhanced context model.

The *user* and his goals, habits and preferences, electronic *resources* like, e.g. a mobile device, network connectivity and the consumable *information* and services, are interrelated and embedded within an *environment*. The environment influences this relationship and comprises the *social sphere*, the *technical sphere*, and the *ecological sphere*. The social sphere is concerned with the social contacts of the end-user and a possible shared task, as well as other people present. A reminder mechanism can be initiated, for example, in the event a specific person arrives with whom the end-user wishes to speak. The current location of the user, the brightness and vibration of the location, and the noise and temperature of the environment is reflected by the ecological sphere. The technical sphere reflects the available technical infrastructure.

All of these things form the current usage context. The context is continuously tracked by sensor or monitor agents. Further, it should be stressed that the usage context is permanently changing. Time is thus an important attribute

[1] A somewhat more simplified context model is first introduced within the MoVi Project [14].

of the context and enables the recognition of usage patterns.

The context manager should maintain the current context and provide as much information as necessary to interpret the current »working« situation and best assist the end-user in performing tasks. In order to be of use to the end-user, the situational information must be captured using appropriate sensor technology and stored in the context knowledge base, all automatically.

1.1 The MoVi Project

From our experience in mobile computing research in the MoVi project [14], we know that the provision of context-aware applications has a great impact on offering real intelligent assistance for the end-user, not only in the mobile area. During the research effort, the notion of mobile assistance was coined, which emphasizes a mobile device as a non-intruding companion and not as PC surrogate within a working situation [3]. The new interaction paradigm allows the end-user to focus on work, rather than on tools, to accomplish his tasks.

During the MoVi project, our primary concern involved the resource context of mobile end devices, network connectivity and visualization-specific services. Given the necessary information, a visualization pipeline that is best suited to the current resource context is established. As an example application, we have developed an architecture for the efficient access to any content in the WWW from a mobile device [5]. One perspective of the proposed system was to further extend the available context information, taking into consideration not only the device properties within the adaptation process, but the whole usage context as well, e.g. the current location.

1.2 The Embassi Project

Our research effort in context-aware computing is currently being further investigated in the EMBASSI project. The EMBASSI project is sponsored by the German Ministry of Science and Education (BMB+F). Its goal is to provide service assistance for consumer electronics in private households, individual traffic and in the terminal domain (automatic teller machine).

A personal agent [12] as a human-computer interaction paradigm helps the user to efficiently accomplish his tasks, such as the »control« of electronic systems like recording movies on a VCR, watching TV, or listening to music in cars. The EMBASSI system focuses on the most natural and user-oriented assistance.

If, for example, the user wants to record a TV movie on a VCR, he points to the movie and says »Record the movie, James«. Therefore, not only speech recognition and multimodal interfaces[2], but situation-awareness also play an important role. This is a prerequisite to delivering the necessary situational information for evaluating the sentence without querying the user too often. The context provides the missing information for the correct interpretation of the sentence.

As previously introduced, the context is established by information about the location of the user, the environment in which the user acts, the detection of a user's presence or absence, and additional information about physical objects in his surroundings. Situation-aware applications can best react to the user changing context automatically and instantly. For example, the detection of a user's presence in his home environment might lead to an evaluation process to determine his next relevant task, e.g. an important telephone call.

With regard to this situation awareness, one of the central components of the EMBASSI system is the context manager (CONJURER) (see Figure 2).

From a software engineering point of view, the most challenging approach in EMBASSI has to do with the connection of a number of independent and heterogeneous system components from the technical environment via an intelligent bus system. To speed up and simplify the integration process, an agent communication language (ACL) [9] was proposed and will be supported in the project work.

Accordingly, the CONJURER offers a KQML[3] interface and serves - in addition to its global knowledge capability - as the routing and service brokering facilitator.

---

[2] The user interface of the service assistant might have a three-dimensional avatar.

[3] The Knowledge Query and Manipulation Language (KQML) is the de facto standard for an ACL. For a detailed discussion of different ACLs see [9].

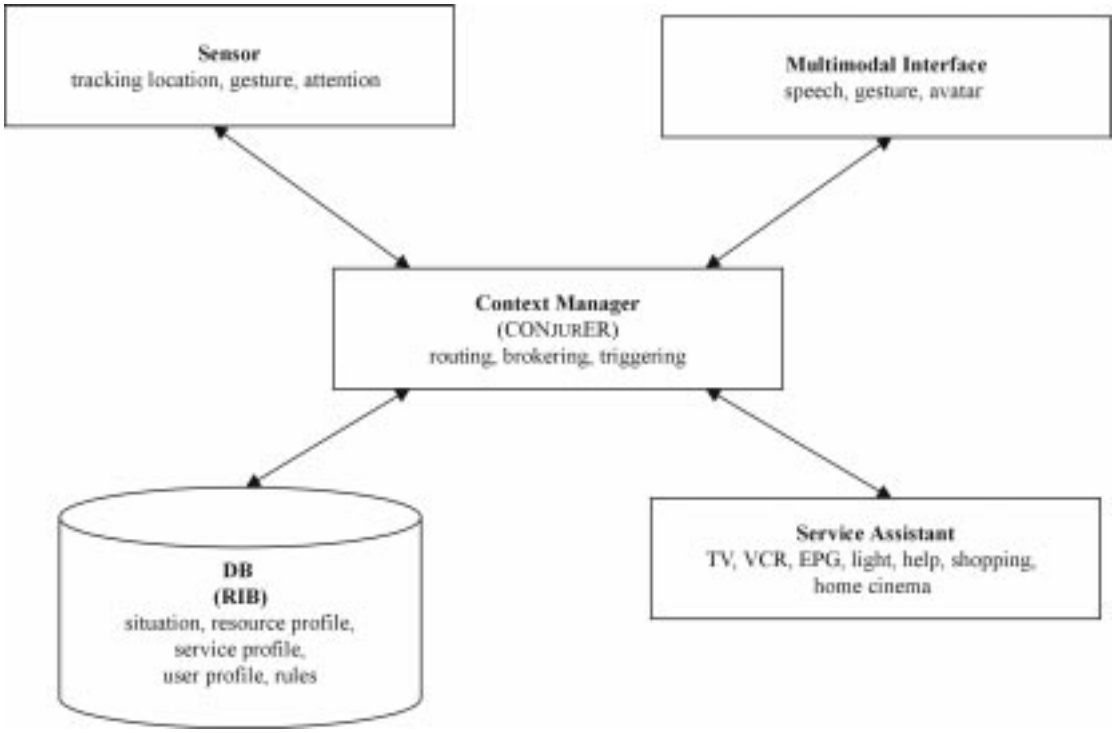

Figure 2: The CONJURER **as a global knowledge base and facilitator service embedded within a multi-agent architecture.**

Figure 2 shows a simplified system architecture of the EMBASSI system, in which the CONJURER is embedded.

Sensor agents are responsible for acquiring row data from hardware or software sensors, for interpreting the data input so that the information retrieved is meaningful to the application, and for storing the information in the context database using the CONJURER interface. A user-tracking sensor agent, for instance, monitors a specified room and reports the arrival and departure of people at that location.

Service assistant agents work as personalized agents. For example, a user assistant agent can track the behavior of the user and insert usage patterns in the context database. Furthermore, personal agents can aggregate and combine the information delivered by sensor and other producer agents, (e.g. agents from the multimodal interface module), and store more sophisticated information in the database (e.g. »Holger is watching TV«).

Each module in Figure 2 contains several loosely coupled agents, who collaborate with each other on top of KQML. Every agent in the multimodal interface, as well as in the service assistance module, can produce or consume context information. The hardware device agents that produce process and state information (e.g. »TV channel ARD is activated«) and operate the consumer electronics are incorporated in the service assistance module.

1.3 Overview

The article is structured as follows. Section 2 provides a short introduction and overview of existing research activities and sensor technology for situation-aware computing. Section 3 continues with the description of the context-management component. Here, a suitable data model and data management facility, as well as a notification service, are sketched. Finally, an example application is given to validate the CONJURER approach and a conclusion completes the article.

2 Situation-Aware Computing

Currently, a great deal of research work and experimentation is underway in the domain of situation- or context-aware computing. All projects revolve around the mobile situation, in which the device or the user is a mobile entity. In most systems, the location context is the major criterion for efficiently supporting the user. Among these are, for instance, a mobile museum guide [10] and a mobile tourist guide [4], which have both been presented at the IMC'98 workshop, and the stick-e-notes system [1], which is a generalized digital equivalent of Post-It notes.

In addition to the location information, user preferences, cultural event information, the actual time and opening hours, and other environmental information are often considered added value information, which constitutes the actual user context. Based on this information, the user is guided and assisted in a sophisticated manner. The tourist-guide application, for instance, queries the user about his preferences, takes the weather and opening hours into account, and offers its user a context-aware sightseeing tour through a town.

To track the location of the user, positioning systems like GPS receivers for outdoor »job« and the Active Badge systems for indoor »work« (office or home environment) are used. Gesture recognition or eye tracking can also be integrated to account for the physical and cognitive situation of the user. One of the central problems in regard to user tracking lies in interpreting the sensor data, so that it is useful to the application (cp. [7][13]).

AdaWeb [5] is a prototyped system developed in our MoVi project [14] for the adaptation and efficient access of Internet services and information from a mobile device. In the system, device and network resources, as well as user profiles, constitute the context information, which in turn guides the adaptation process. In Figure 3, the proposed architecture for a context-aware adaptation within a mobile WWW environment is shown. The primary concept behind AdaWeb is to exploit meta-information about the resource environment and user preferences in regard to user adaptation requirements, in order to find a suitable adaptation pipeline to overcome most limitations of mobile information access.[4]

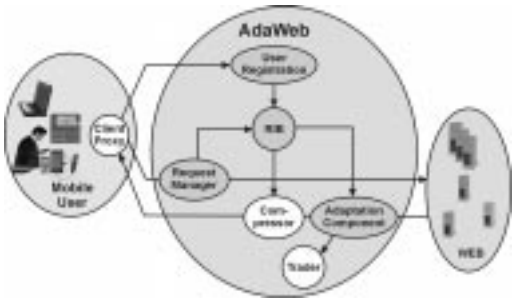

Figure 3: Architecture of the AdaWeb system.

[4] Furthermore, we won a thorough insight to location tracking using position receivers like GPS within our system Loci [6]. We also carried out some experiments with the Active Badge system.

Most systems support a central remote global database for the storage of context information. This database can be queried for situations or situations can be forwarded to registered components based on an event propagation mechanism. As a matter of fact, the data model and data access mechanism, which is one of the most important aspects of a context database - the common base of context-aware capabilities - is presented only rudimentary and application-specific as in [13]. A proposed data model and access mechanism will be discussed in more detail in the following section. Thereby, the concept of the CONJURER is comparable to what Pascoe et al describe as a context information service [11].

3 Context Management

The CONJURER serves as a central[5] knowledge store, in which registered components provide facts about the current world. Facts about the world that are consumed by more than the producer agent alone are the only reasonable ones to store in the context base.

The CONJURER should not only maintain the current context, but should also provide the following functionality:

- Asynchronous coordination mechanism between independent system components;
- Registration mechanisms and service brokering, as well as an event propagation mechanism;
- Routing mechanism to deliver KQML messages based on symbolic names.

Each component added to the EMBASSI system needs to register and advertise its service to the CONJURER. The CONJURER is responsible for registering (white page service) the system components, e.g. VCR service assistance to record a movie and to route messages based on symbolic names to their respective receivers. It is also responsible for service brokering (yellow page service). In addition to these facilitating services, the CONJURER offers persistent insertion and reading of situational information. In this article, we discover the data store and event propagation mechanism in more detail.

A brief final remark for *not* using JINI as the enabling technology for networked consumer electronics shall be given at this point:

[5] Regarded from the logical viewpoint. From the technical viewpoint the CONJURER is distributed. Distributed context-management was investigated within the database group of the MoVi project.

- JINI is a Java technology. In Embassi, we have to support heterogeneous systems from Java, C/C++ to LISP like languages. Even non object-oriented languages must be supported.
- JINI is a technology for spontaneous networks. Services are allocated and are only available for a short period. In Embassi, agents are also integrated in an ad-hoc manner, but are available for a long time.
- JINI is restricted to a rigid client server principle, whereas, in Embassi, every agent is both a client and server at the same time.
- JINI supports only the registration, brokering and event propagation mechanism. The CONJURER provides a global knowledge base, incorporating an efficient rule-based conditional notification mechanism. One strength of this mechanism is the homogeneous view of all the context data using the flat data model approach.

The data store mechanism of the CONJURER must fulfill the following requirements:

- Global persistent knowledge base. It stores important information, which will be consumed by more than one component;
- Distributed on mobile and stationary systems, small footprint;
- Usable from heterogeneous systems and programming languages;
- System-wide homogenous view to data;
- Simple yet efficient access mechanisms.

Taking into account these requirements and the previously mentioned research interests originating from the MoVi project, we decided to establish the context management facility upon the resource information base (RIB), the resource-context management component of the realized system AdaWeb [5].

3.1 Classification of Context Data

In regard to all the distinct information, such as static resource profiles and user profiles, as well as dynamic agent state and environmental information, which is to be stored in the context, we will first introduce a classification scheme. This should allow us to get a better understanding of what kind of information is important to maintain and help to better organize the data in the context knowledge base.

We can differentiate between *static* or *dynamic* context *knowledge*. The former incorporates resource profiles of hardware devices, software services, or static user profiles. The latter comprises, for instance, the processing state of resources or the user location (see Figure 4).

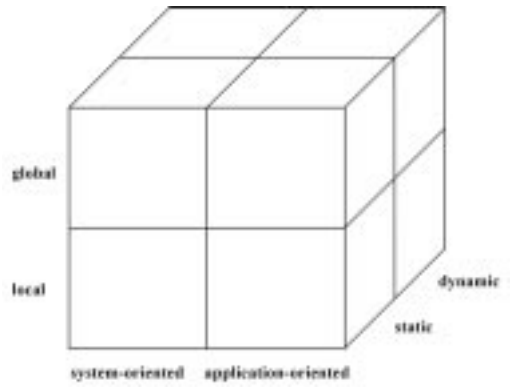

Figure 4: Classification scheme.

Furthermore, a *local* and a *global situation* are discernible. A local situation is associated with a single object in the world and therefore resembles the situation of a micro world. In contrast, a global situation is a fact about the whole world, that correlate multiple objects – a macro world. Local and global *situations* are both global *information*, which is consumed by more than just the producer agent alone and therefore must be maintained in context. An example of a global situation is the fact »Currently, the kitchen is occupied by Holger«. A local situation is the fact »Holger is an excellent cook«.

Last, but not least, *system-* and *application-oriented facts* can be distinguished. The first is only of importance for the system components, such as the volume range of a TV. The latter is meaningful to the end-user.

In Figure 4, a cube visualizes the classification scheme. For example, the context information »Currently, the kitchen is occupied by Holger« is categorized to be a global situation, an application-oriented fact and dynamic knowledge.

3.2 Data Model

We are proposing a flat data model for the storage of context objects. This model offers the necessary specialization mechanism and a homogeneous view to all context objects. This is a prerequisite in order to easily apply the rule-based conditional notification functionality of the CONJURER.

Information is inserted in the form of a tuple of data values – the context objects. A database query is performed using a template of a tuple, which can hold data values and wildcards. The

CONJURER searches the database and delivers the first or all entries matching the query while replacing wildcards. In our system, a context object consists of a tuple of a *layer, section, key* and *value,* as well as an optional *time* flag entry. An example XML context-object representation is given below:

```
<contextobject
   layer="situation"
   section="TV_Loewe"
   key="volume"
   begin="2000-07-10 11:00:00"
   end="2000-07-10 11:30:00">
20
</contextobject>
```

The data model offers a unified view of all context information provided. In order to provide a hierarchical information model, inheritance and specialization of properties must be supported. Therefore, we are using a layered approach in which the database is organized as a stack of ordered individual *layers*. Each layer is a grid in which rows are *sections* and columns are *keys*.[6] The information or *values* are stored in the grid cells.

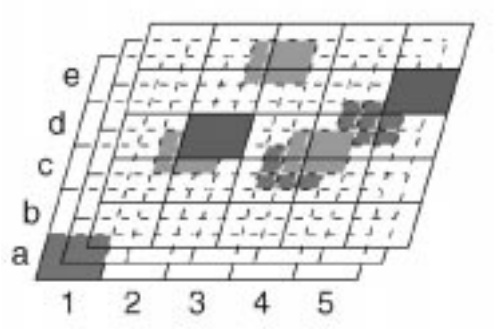

Figure 5: Layer concept of the CONJURER**.**

This data model is comparable to tuple spaces [2], introduced for parallel computing and currently proposed as a simple yet powerful blackboard coordination mechanism for agents. The subtle distinction between a tuple space (e.g. JavaSpaces) and the data model of the CONJURER is the following:

- The CONJURER provides a simple inheritance mechanism based on the layer concept;
- Tuples within CONJURER are given a time flag, which enables a history mechanism and secures consistency of context data. Due to a possible unreliable message delivery channel, without a time flag, there would be no guaranty that the newest situation is maintained in the tuple space.
- It is not possible to automatically remove a tuple upon a read access. Furthermore, tuples are not deleted explicitly, but implicitly, using a start- and end-time attribute. This is necessary, because more than one agent is allowed to consume the context data. Thus, only the producer agent could have full access rights to delete the data. However, the automatic deletion of context data after an elapsed time period reduces message traffic and management effort on the agent side. Furthermore, given the time mechanism, temporally valid profiles can be maintained, as well.

3.3 Data Management

The CONJURER provides persistent storage of information, which is inserted and queried by the registered system components using KQML messages. Information is written into the CONJURER explicitly by the usage of an appropriate insert message. In order to avoid the message overhead regarding the explicit deletion of information and in order to avoid an inconsistent database, information is deleted implicitly. The first method for removing entries is to evaluate the time flag provided in the tuple of the insert message and delete the entry after the elapsed time period. The advantage of this method lies in the association of some context information with time information which is used for automatic deletion. With appropriate time flag values, a history of context information can be established which has value for some components, such as a user assistance agent learning user behavior.

In the event the time flag is omitted, which is the second method, the CONJURER overwrites old entries with new ones. This is useful for time-independent context information, which must be available during the whole session. The CONJURER distinguishes two services for information access: polling and notification.

Polling

The context information can be queried by the system components using the polling functionality. In the case of time-dependent information, even a history can be inquired given the time flag. Due to the communication costs associated with permanently polling the database for new information, the CONJURER provides an event propagation or notification mechanism (see the following section also).

[6] Likewise the associated time flag is stored within the key.

The insertion and polling functionality of the CONJURER, which correspond *write* and *read* operations of tuple spaces, is performed using the following KQML performatives[7] within the messages: *insert*, *ask-one*, and *ask-all*.

The message *ask-one* will deliver the first matching entry in the database, whereas *ask-all* will return all of them. For example, an agent who learns the behavior and preferences of the user inserts a user profile by sending the message[8]

```
insert("Holger" "Fernsehen"
"Nachrichten" "(Tagesschau ARD
20:00)")
```

to the CONJURER. The CONJURER can than be queried with the message

```
ask-one("Holger" "Fernsehen"
"Nachrichten" "?x")
```

in which `?x` is the wildcard which will deliver the entry given above. If a component is interested in all users with the same preferences, the query must be given in the form

```
ask-all("?x" "Fernsehen"
"Nachrichten" "(Tagesschau ARD
20:00)").
```

Likewise, the CONJURER can be queried for all users and their preferences regarding the genre »Nachrichten«[9] or against all preferences of a user associated with TV programs. In either case, the tuple of the query incorporates two wildcards.

Notification Service

In order to reduce message traffic among the system components and to provide an instant reaction to an environmental situation change, the CONJURER offers an event propagation or notification mechanism[10]. Two operations for notification are distinguished: *notifiy* and *conditional notify*[11].

The former offers a simple event propagation mechanism, which informs every subscribed system component about the change within the knowledge base according to a specified event. The event resembles the context object going to be updated. The latter takes into account that additional situational facts of the current context are related to the event.

The agents of the system can insert a notification rule into the CONJURER. The premise of a rule is defined by a Boolean expression of situational information. When the premise of a rule is evaluated and the condition is fulfilled, the actions of the rule are executed. The actions of the rule resemble a *tell* KQML message that has situational information as message content.

The approach of a conditional notification provided by the CONJURER moves the logic of the situation interpretation from the system components (agents) to the CONJURER. This reduces message traffic and latency dramatically, because all the necessary context information is computed at its source. Simple situation-action rules are used which can be inserted into the context database using a *subscribe* message with the rule and its triggering event as its content.

4 Application Example

As application example to test the CONJURER approach, we have established a simple In/Out Board and an environmentally sensitive switch for the TV. For the realization of the sensor agent functionality, we used the ActiveBadge system, which tracks the location and movement of department staff members. We have installed ActiveBadge readers in three labs of the department, as well as on the floor between the labs.

The In/Out Board visualizes the location and motion of staff members (see Figure 6). In order to do so, it makes use of the event mechanism and queries the CONJURER for the current situation. The environmentally sensitive switch changes the current volume and channel of the TV in case more than two persons are in the lab. If the department chief is in the lab with the TV, the program switches to a news channel and lowers the volume. Should he leave the room, the program

---

[7] Performative is the name of the speech-act used for KQML messages.

[8] For the discussion of the CONJURER KQML interface, we are using the short form *performative*(X), where X is the content. The sender and receiver fields are omitted because the sender is the program sending the message and the receiver is always the CONJURER. Likewise the ontology and language fields are fixed. Every message needs a message identification in order to relate an answer to a query. Furthermore, the content itself is delivered using the XML syntax.

[9] German expression means news.

[10] cp. distributed events, remote event notification

[11] cp. notification filter

switches to a sports channel and raises the volume again.

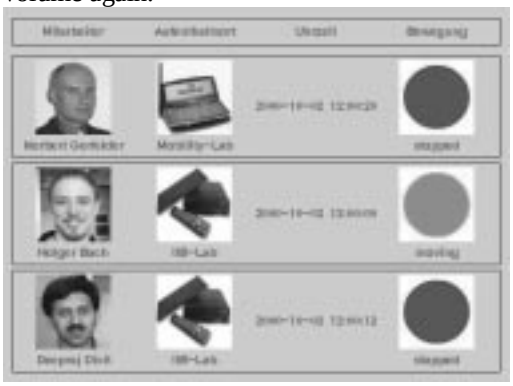

Figure 6: The In/Out Board.

5 Conclusion

In this article, we have proposed a simple yet powerful mechanism to store and retrieve context information based on KQML message interchange and the idea of tuple spaces. First test drives have shown the usefulness of the CONJURER mechanisms and interface, as is the case for the interface and mechanisms already in use in the AdaWeb system. Of course, in this application, we do not use KQML and have somewhat different context information. In addition, the interconnection and discovery of remote context managers, which is essential for mobile clients and terminal applications, as well as replication mechanisms, must be further investigated. Nevertheless, situation awareness is becoming a prerequisite for efficient and effortless assistance to the end-user. The consideration of situation-aware systems is a step beyond location-aware systems, which only account for the location of the user. Situation-awareness allows for an instant and sophisticated information processing, helping the end-user to perform his daily work optimally.

Acknowledgement

The authors would like to thank Deepraj Dixit for the excellent implementation work covering the time functionality, fine-tuning and debugging of the CONJURER, as well as the TV switch application. Norbert Gerfelder and Deepraj Dixit also deserve thanks for providing us with such a nice photograph.

References

[1] Brown, P., Bovey, J., Chen, X., "Context-Aware Applications: From the Laboratory to the marketplace", IEEE Personal Communications, 1997 4(5): 58-64.

[2] Carriero, N., Gelernter, D., „Linda in context", Communications of the ACM,Vol. 32, No. 4, April 1989.

[3] Chavez, E., Ide, R., Kirste, T., *„SAMoA: An experimental platform for Situation-Aware Mobile Assistance"*, Proceedings of the 2nd Workshop on Interactive Applications of Mobile Computing IMC'98, Rostock, Germany, November 24-25, 1998.

[4] Cheverst, K., Mitchell, K., Davies, N., *„Design of an Object Model for a Context Sensitive Tourist Guide"*, Proceedings of the 2nd Workshop on Interactive Applications of Mobile Computing IMC'98, Rostock, Germany, November 24-25, 1998.

[5] Freytag, C., Kumpf, C., Neumann, L., *"Resource Adaptive WWW Access for Mobile Applications"*, Proceedings of the 2nd Workshop on Interactive Applications of Mobile Computing IMC'98, Rostock, Germany, November 24-25, 1998.

[6] Freytag, C., Neumann L., "Location-Aware Computing", Computer Graphics Topics, 2/98.

[7] Hull, R., Neaves, P., & Bedford-Roberts, J. (1997), *"Towards Situated Computing"*. In Proceedings of the International Symposium on Wearable Computers, (pp. 146-155). Los Alamitos, CA, USA: IEEE Computer Society.

[8] Kanter, T., Frisk, C., Gustafsson, H., "Context-Aware Personal Communication for Teleliving", Personal Technologies (1998) 2:255-261, Springer-Verlag London Ltd.

[9] Labrou, Y., Finin, T., Peng, Y., "Agent Communication Languages: The Current Landscape", IEEE Intelligent Systems, March/April 1999.

[10] Oppermann, R., Specht, M., *"Adaptive Support for a Mobile Museum Guide"*, Proceedings of the 2nd Workshop on Interactive Applications of Mobile Computing IMC'98, Rostock, Germany, November 24-25, 1998.

[11] Pascoe, J., Ryan, N., Morse, D., *„Issues in Developing Context-Aware Computing"*, in Hans-Werner Gellersen (Ed.): Handheld and Ubiquitous Computing, First International Symposium, HUC'99, Karlsruhe, Germany, September 27-29, 1999, Proceedings. Lecture Notes in Computer Science, Vol. 1707, Springer, 1999, ISBN 3-540-66550-1.

[12] Riecken, D., "Intelligent Agents", Communication of the ACM, July 1994, Vol.37, No.7.

[13] Salber, D., Dey, A., Abowd, G., *„The Context Toolkit: Aiding the Development of Context-Enabled Applications"*, in Proceedings of CHI'99, Pittsburgh, PA, May 15-20, 1999, ACM Press.

[14] Prof. Dr. Heidrun Schumann, Prof. Dr.-Ing. José L. Encarnação: "Visualisierung multimedialer Informationen mit mobilen Computersystemen", DFG-Projekt Mobile Visualisierung (MoVi), Zwischenbericht, 31.Juli 1998.