Smart Reminder - Personal Assistance in a Mobile Computing Environment

Frank Kargl, Bin Dong, Torsten Illmann, Michael Weber
Department of Multimedia Computing
University of Ulm
89069 Ulm
Germany
+49 731 5031310
{frank.kargl|bin.dong|torsten.illmann|michael.weber}@informatik.uni-ulm.de

ABSTRACT
In this paper, we describe the Smart Reminder agent that we developed using our CIA personal agent framework. We first motivate the need for personal assistance and how this need led us to the development of the CIA agent framework. Smart Reminder is composed of a set of personal agents that support their user in all situations where people come together spontaneously, e.g. an encounter on a corridor. Smart Reminder will notice the presence of another person and display identity information like name, affiliation, etc. In cooperation with other CIA agents it will also display information related to that person like common dates and tasks. After the introduction of this scenario and a description of the functional range of Smart Reminder we continue our paper with some technical aspects of the application like person detection and communication via Bluetooth. We conclude our paper by giving an outlook how Smart Reminder can be extended in the future.

Keywords

Personal assistance, personal agents, CIA framework, Smart Reminder

INTRODUCTION

Personal Assistance

In today's rapidly changing world, people spend more and more time with the coordination of information. In spite of the large amount of technical tools available, like phone, fax, email or the WWW, there are research studies showing that for many people this overload of information and communication activities is simply too much [1]. In contrast to the introduction of ever new technology, productivity of professionals sometimes even diminishes. So today's computers often fail to deliver their promise of being helpful productivity tools.

Personal Agents

Personal Agent Technology [2] offers a promising way out of this dilemma. Although the definition of what actually makes out a software agent varies between different fields of computer science, there is probably one common aspect:

> "*Software agent*: A computing entity that performs user delegated tasks autonomo usly."[2]

So like a good secretary they should allow their users to delegate routine tasks and get them finished without any further user intervention.

Collaboration and Coordination Infrastructure for Personal Agents (CIA)

Many agents today concentrate on a single task where assistance should be provided. So even when a user is assisted by a number of different personal agents, it is still up to him to coordinate these different agents. Think of a travel scenario, where a user needs to coordinate different tasks like selecting appropriate transport means like train or plane, booking a hotel room and arranging dates with business partners he wants to meet at the foreign location. Even if there are agents supporting any of these single tasks, it is still up to the user to ensure that the plane will arrive at the destination in time for the meeting or that there is enough time to get from the train station to the airport etc.

If we succeed in making the single agents cooperate, the benefits are clear: the user can simply tell his cluster of personal agents to make a date with some other persons and arrange a suitable accommodation and transport.

Example Scenario

Think of organizing a meeting in your company. Instead of sending a dozen mails back and forth or phoning all the participants a number of times to fix a date and reserve a suitable meeting room, you just tell your personal diary agent to do all this for you. All you have to do is to specify who should join the meeting, a timeframe when the meeting should happen and if a meeting room is required and which properties (e.g. a beamer) this room should have. The rest is up to the personal agents of the participants. After a few

minutes you will get a notification with the exact date and room.

The CIA Framework

In order to allow such scenarios the different software agents need to cooperate in a complex way. First the agents responsible of date organization (which we call Diary Agents) have to agree on a concrete date for the meeting. Afterwards the Diary Agents have to send a request to another agent responsible for room reservation. This Room Reservation Agent will then contact a room reservation service responsible for the building where the meeting is to happen and reserve a suitable room.

It is reasonable to encapsulate most of the basic functionality in a middleware component that relieves the agent programmer from dealing with details of the underlying communication system, database engine or application server.

It is the aim of the CIA project [3] to realize such a framework for personal agents that provides:

- Easy implementation and integration of software agents.
- Powerful but still easy-to-use communication features.
- Easy and location transparent access to agents of other users and to external services.
- Integrated support for commonly needed features like persistency, security or weak and strong mobility.
- Support for flexible and device independent user interaction [5].

All personal agents belonging to one person are concentrated in a virtual construct called Agent Cluster [6] that can be spread out over a multitude of different physical devices that are inter-connected by a communication network. Each user owns exactly one Agent Cluster that will support him every time and everywhere. The communication infrastructure may vary widely in availability and quality of service, as all communication is based on asynchronous message exchange. Within the cluster communication is possible via the so called Agent Bus which consists of several topic-based communication channels [4]. External functionality that is not related to a user is offered by so called Services. The location of suitable services or of clusters of other persons is done by a directory, broker and trader component (DBT). Figure 1 represents an overview of the CIA architecture with two agent clusters, two services and the lookup component.

CIA is based on a number of technologies that largely influence its capabilities:

- Message Oriented Middleware: this provides location-transparent, asynchronous communication channels. Addressing of messages is not done directly but instead semantic topics are used to deliver messages to all interested participants [4]. Agents can subscribe to channels. Whenever a message is posted to a channel, it is delivered to all subscribers. We support different MOM implementations using a factory pattern, ranging from a simple local dispatcher that runs on very small devices within a Java VM to multicast-based message delivery in arbitrary IP networks. A segment of an Agent Cluster running a specific AgentBus implementation is called a subcluster. Different subclusters are bridged by Cluster Routing Agents (CRA) which implements advanced store-and-forward capabilities like expiration, prioritization, etc. These mechanisms cover short network outages or overload of lines with small bandwidth. The messages can contain any kind of data like serialized objects or FIPA ACL messages [28].
- Mobile Ad-Hoc Networking: The topology between the components of an Agent Cluster is not fix. So advanced topology discovery and routing for mobile radio networks is integrated [7][8].
- Discovery Service: We use Sun's Jini [9] for easy location of Agent Clusters and services. This is augmented by broker and trader services that e.g. find a suitable room reservation service for a given building.
- Services: Most services in the CIA framework are realized as Java Enterprise Beans [10] that are embedded in J2EE application servers. Nevertheless, the framework is open to easily integrate services of different technologies like Web Services, RMI or CORBA.
- Device Independent User Interaction: In [5] we present the concept of the User Communication Agent (UCA) which is responsible for all interaction between agents and users. Agents don't implement their user interface themselves but instead merely present an abstract description to the UCA. Users can freely connect or disconnect to the UCA with whatever device they are presently using. It is the responsibility of the UCA to generate a user interface fitting the device in use and to relay events between the agent and the user. This way only a small subset or even none of the agents in use needs to run on the user's current device.

This flexible framework allows the easy implementation of complex cooperation scenarios that are distributed, mobile and multi-user capable. In order to demonstrate this, we developed the Smart Reminder Agent.

The Smart Reminder Agent

Introduction

All of us usually have social connections to a lot of other people. We have dates with others, common interests, work together in projects etc. Sometimes these social connections simply become too many. Often when encountering a person in a crowded area, we can't remember the name although we are sure to know this person. Or we meet someone, talk to him and five minutes after we left him we suddenly remember that we should have discussed with him a very important matter. This is where the Smart Reminder helps and supports.

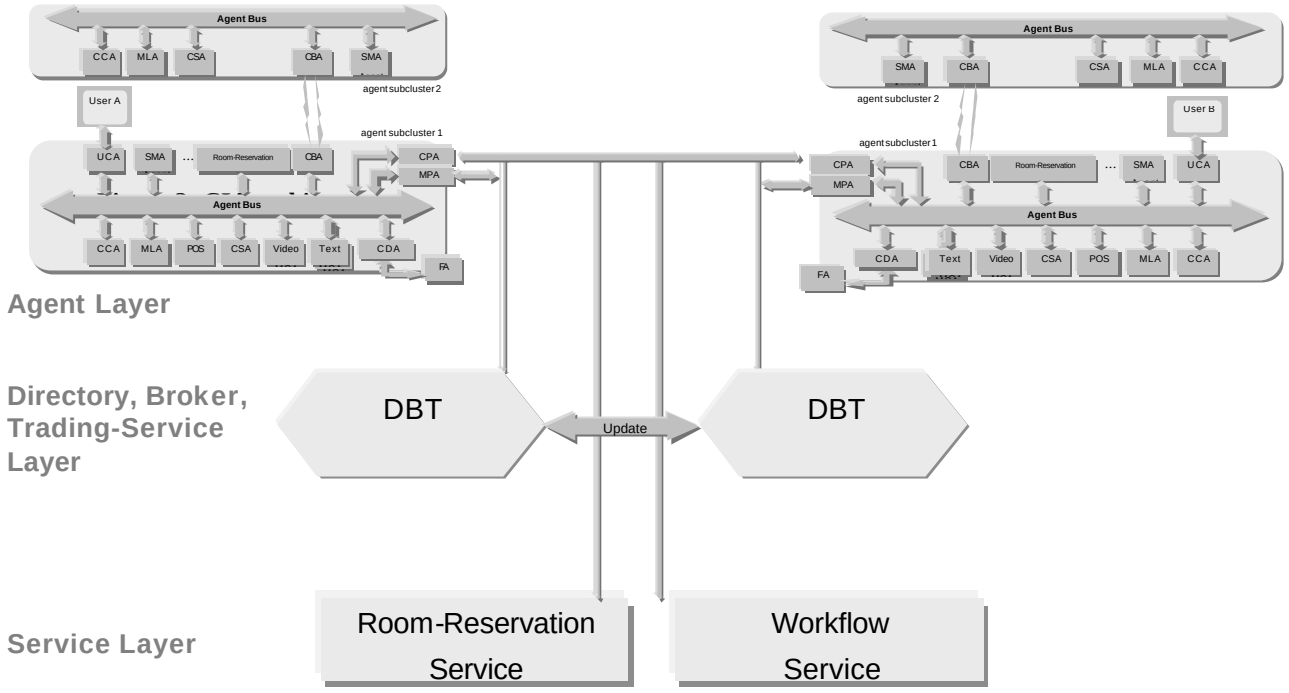

Figure 1: CIA Architecture

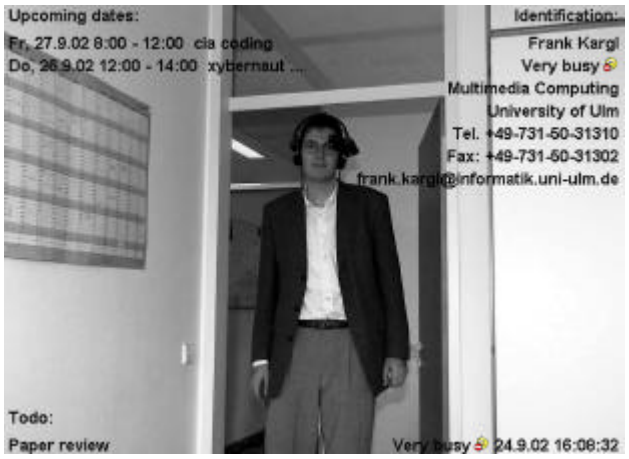

Figure 2: Smart Reminder Display

The Smart Reminder is a personal agent in the CIA framework which offers a reminder service based on the current context of its user. As you can see in figure 2 the Smart Reminder Agent (SRA) displays information on the identity of another person. Additionally the SRA is able to gather related data from other agents in the own or a remote Agent Cluster. E.g. the SRA can search the diary agent for dates or the task agent for tasks related to that person.

Example Scenario

Whenever a user Alice meets another SRA user Bob e.g. on a corridor (see figure 3) or in an office the two SRAs involved will detect this encounter and exchange identity information (name, affiliation …). This information is then presented to Alice and Bob e.g. via a head-mounted display or as speech output using a small headset.

In a next step, Alice's SRA agent contacts her other personal agents searching for dates and tasks related to Bob. That way Alice has all the relevant information regarding the person met immediately available. She will be informed about upcoming dates like a planned lunch with Bob next day and may then ask him if this date is still valid. By displaying tasks related to the encountered person she will never forget to ask Bob important things like the status of the development progress of a joint project or socially relevant things like: "How is Your family?".

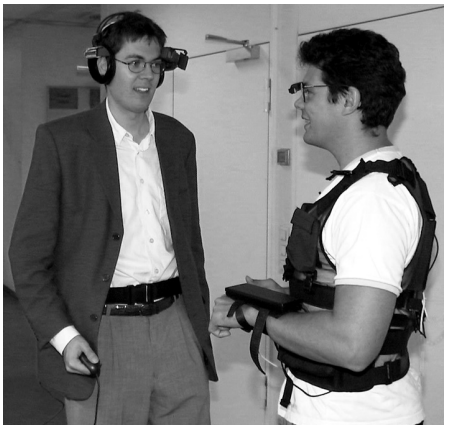

Figure 3: Spontaneous Meeting

Participating Agents

As said before, the different functions that overall form the Smart Reminder are distributed to several agents. The Smart Reminder Agent (SRA) instructs the Bluetooth Proxy Agent (BPA) to constantly inquire the neighborhood for new nodes. Whenever such a new node appears, the BPA sends a DISCOVERED event to a communication channel where it is received by the SRA. The SRA then sends his cluster ID to the remote SRA. All communication between the clusters is relayed via the BPA. Next both SRAs contact their VirtualMeAgents (VMAs) which administrates the identity information of the cluster owner and preferences like work hours, preferred lunch time, which information to transmit to other SRAs etc. Up to the time of writing this information has to be entered by the user. In the future we intend to extend the VMA to learn this from the behavior of the user. Based on the ID of the remote cluster the VMA also decides what information is to be delivered. This allows to distinguish e.g. between members of a related company, friends from a sport club and completely unrelated persons. Whereas people from a partner company will get e.g. by business phone number, I might decide not to give it to completely unrelated persons. Furthermore the VMA calculates a so called busy-level.

The VMA queries the Task and Diary Agent for upcoming tasks and dates. Depending on deadlines and duration the VMA can estimate how busy the user currently is. The SRA can then present this information e.g. to a colleague who can in turn decide not to disturb the user with unimportant things if he is extremely busy.

The level of busyness is computed as follows:

Let

now = the current time

T = $(t_1 ... t_n)$ a list of tasks to be done from now ordered by deadline

A_{now} = set of all appointments ending after now

wt_{now} = set of all working hours ending after now

The available worktime from now till the end of task t_j is:

$$wt_{avail,\,j} = \sum_{h=now}^{t_j.deadline} h \mid h \in wt_{now} - \sum_{a \in A_{now} \mid a.end \le t_j.deadline} a.duration$$

The busy-level b_j considering tasks t_1 to t_j is:

$$b_j = \frac{\sum_{k=1}^{j} t_k}{wt_{avail,\,j}}$$

The resulting overall busy-level b is the highest busy-level:

$$b = \max_{1 \le j \le n} (b_j)$$

Whenever a cluster receives information from a remote cluster, this information is automatically sent to the Address Book Agent (ABA) which stores it for further reference.

As soon as the identity information is exchanged, the SRA displays this information to the user. The SRA uses the User Communication Agent (UCA) for this task. All user interaction in CIA is handled by the UCA. The agents only describe their "user-interaction-interface" to the UCA. Whenever a user connects to the UCA (using e.g. a web-browser or a telephone-interface) the UCA renders this description e.g. to a graphical user interface or generates a speech dialogue. Input, events etc. are delivered back to the agent. Using this mechanism, the system can adapt to different input/output devices.

In order to display more useful information, the UCA sends a request to an Agent Bus channel asking all the subscribed agents to deliver information regarding the met person. So e.g. the diary agent will send dates and the task agent tasks that are related to that person. Again this is displayed to the user.

Implementation Details

Programming Language

All components of the CIA framework and the personal agents are written in Java [11] ensuring easy portability to a large range of devices.

Communication Infrastructure

The Smart Reminder Agent needs to recognize an encounter with another person. There are several possibilities to achieve this. One of the options is using RFID tags [12] and corresponding RFID readers. This allows the detection of other persons but we then need a separate communication channel for inter-cluster communication.

We could also use IEEE 802.11 Wireless LANs [13] in ad-hoc mode to detect neighbors. But as the range of these radios is quite large (up to a few hundred meters in open space), we will probably detect too many persons e.g. inside other buildings. A combination of RFID encounter detection and WLAN communication could solve all the problems but has the disadvantage of needing separate devices.

For this reason we decided to use Bluetooth for both encounter detection and communication. As Bluetooth Class 3 has a limited range of up to 10 meters, this fits pretty well with our requirements. However, Bluetooth has also one disadvantage: the inquiry process may take up to 10 seconds to detect a new device so our prototype sometimes reacts a little bit slow on new encounters. On the average, new devices are discovered after 5 seconds which is tolerable. After a new device has been discovered, the Bluetooth Proxy Agent establishes a connection to the other device which is then used to transmit all data. The theoretical data transmission rate is 1 Mbps so even with the overhead introduced by the communication protocols

there is still enough bandwidth for the Smart Reminder application.

Hardware and Bluetooth Stack

We are using Acer USB Bluetooth Dongles with Bluecore Chip from Cambridge Silicon Radio [14]. As all Bluetooth hardware has a well standardized Host-Controller Interface (HCI) it is pretty easy to integrate different Bluetooth devices. On the host side, we need a so called Bluetooth stack which handles the different communication profiles like RFCOMM for serial line emulation or PAN for personal area networks. As most of our implementation is done on the Linux operating system, we tested different Linux Bluetooth stacks for their suitability. After testing the OpenBT [15], the BlueZ [16] and the Affix [17] stack we finally chose Affix because it showed the best stability of the candidates.

Up to now no standardized Bluetooth API for Java is available (although a standardization effort has just produced a final proposition [18]), so we use a self-written native JNI adaptor to access stack functions from the Bluetooth Proxy Agent. As Bluetooth support becomes integrated in the standard Java APIs, we will omit this Linux-dependant component from our application.

Other Hardware

Our demonstration prototype uses Linux (SuSE 8.0) on MA V wearable PCs from Xybernaut. We also have a number of different I/O devices available, like wearable flat-panel displays, a number of head-mounted displays, wearable keyboards etc. Using this equipment we are at the beginning of testing and improving the usefulness of the Smart Reminder using different set ups in user trials. Important questions being investigated include:

- How do users in a wearable scenario interact with their Agent Cluster?
- What impact have the properties of the communication infrastructure (e.g. Bluetooth inquiry time) for the application?
- How could the Smart Reminder be extended to support more than two persons in an encounter?

Related Work

Projects in the area of personal assistance may be divided in three major categories:

- Context-aware information gathering
- Optimization and filtering of messages
- Delegation of personal tasks

Context-aware systems mainly assist by considering time- [19] [20], location- [21] [22], activity- and/or persons-dependent situations. Reminder agents like this work may be counted to context-aware systems.

When looking at projects using a reminder functionality, the work of the agent group at the MIT Media Lab attracts attention. They produced a number of different reminder agents, like Streetwise, which provides location-dependent information to its user [22].

Another relevant project of the MIT Media Lab is called Memory Glasses [23] which is a wearable computer-based context-aware reminder system. It uses time, location, and activity context to deliver reminders to its user. The projects' main focus is on personal context by using body-worn sensors to determine what the user is currently doing (e.g. walking down stairs or taking part in a conversation). Memory Glasses then presents reminders associated with that activity using audio output. The main idea is that the context information lets Memory Glasses determine when it is appropriate to interrupt the user with a reminder.

Other similar projects include Lifestreams [24], comMotion [25] or Proem [26].

The CybreMinder [27] of Georgia Tech uses a so called Context Toolkit to deliver reminders to a user based on time-, location- and co-location context.

Whereas all of these systems consist of a single component that stores, transmits and delivers reminders, our system uses the cooperation of a large number of agents working together to deliver aggregated information to the user. Moreover, it combines context-aware assistance with the possibility of delegating personal tasks in one system.

Summary and Future Development

It is our strong believe that personal assistance systems will become more and more ubiquitous in the future. The key-point is that these systems must work as autonomous as possible so that the time and effort saved by these systems is not consumed by additional time and effort in coordinating these different services. So the CIA system tries to establish a framework where different autonomous agents can easily work together to deliver aggregated benefits to their users. The Smart Reminder is a demonstration that shows how the interoperation of different agents focused on single tasks can realize complex applications.

In the future we plan to extend CIA and the Smart-Reminder in two major directions:

First we are currently changing the communication framework in order to use messages based on FIPA [28] and DAML/OIL [29] in order to allow the agents to better interpret the content of messages [30]. This will allow the Smart-Reminder to send a generic request to all agents in the cluster asking for information about a met person and display it to the user.

Another future research direction will be on the integration of speech recognition. The Smart-Reminder agent will overhear normal conversation between persons. If e.g. one person tells another person "Let's meet for lunch tomorrow" the Smart-Reminder can instruct the diary agent of the Agent Cluster to arrange a date tomorrow noon with the other person. The CIA framework will assist the speech

recognition engine by providing the context of the talk (e.g. names of persons, location etc.)

ACKNOWLEDGMENTS

We thank the Xybernaut corporation for providing us some of the necessary hardware to implement our prototype and esp. Torsten Bergander from Visions2Wear for his assistance in the creation of the SmartReminder.

REFERENCES

1. SBT Accounting Systems: The PC Futz Factor, 1993.
2. Caglyan, Harrison. The Agent Sourcebook. Wiley & Sons. 1997.
3. F. Kargl, T. Illmann, M. Weber. CIA - A Collaboration and Coordination Infrastructure for Personal Agents. Proceedings of the IFIP TC6 WG6.1 Second International Working Conference on Distributed Applications and Interoperable Systems DAIS'99. Helsinki, Finland. 1999 .
4. F. Kargl, T. Illmann, M. Weber. Evaluation of Java Messaging Middleware as a Platform for Software Agent Communication. Java Informationstage JIT'99. Düsseldorf, Germany. 1999.
5. F. Kargl, T. Illmann, M. Weber, S. Ribhegge. Dynamic User Interfaces with Java. Proceedings of Webnet'99. Honolulu, USA. 1999.
6. T. Illmann, F. Kargl, M. Weber. Design of an Agent Cluster as Integrative Environment of Personal Agents. Proceedings of ICIIS'99. Washington, USA. 1999.
7. C. Perkins (ed.). Ad hoc networking. Addison Wesley. 2001.
8. C.-K. Toh. Ad hoc mobile wireless networks: protocols and systems. Prentice Hall PTR. 2002.
9. Jini. http://java.sun.com/products/jini/.
10. Java 2 Enterprise Edition. http://java.sun.com/j2ee/.
11. Java. http://java.sun.com/.
12. K. Finkenzeller. RFID Handbook: Radio-Frequency Identification Fundamentals and Applications. Wiley & Sons. 2000.
13. IEEE 802.11. http://standards.ieee.org/wireless/overview.html#802.11.
14. Cambridge Silicon Radio. BlueCore 2External Product Data Sheet. http://www.csr.com/media/pdf/bc02-external.pdf.
15. Axis OpenBT Stack. http://sf.net/projects/openbt/.
16. BlueZ Stack. http://bluez.sourceforge.net/.
17. Affix Stack. http://affix.sourceforge.net/.
18. Java Specification Request 82: Java™ APIs for Bluetooth. http://jcp.org/jsr/detail/082.jsp.
19. E. Chang, P. Maes: Hanging Messages: Using Context-Enhanced Messages for Just-In-Time Communication, May 2001.
20. B. Rhodes and P. Maes: Just-in-time information retrieval agents, IBM Systems Journal special issue on the MIT Media Laboratory, Vol 39, Nos. 3 and 4, 2000 pp. 685-704.
21. P. Persson, F. Espinoza, C. Elenor, GeoNotes: Social Enhancement of Physical Space, Design-Expo at CHI'2001, Seattle, April 2001.
22. J. Youll, J. Morris, R. C. Krikorian, P. Maes. Impulse: Location-based Agent Assistance, Software Demos, Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000), Barcelona, Catalonia, Spain, June 3 - June 7, 2000.
23. R.W. DeVaul, B. Clarkson, A. Pentland, The Memory Glasses: Towards a Wearable Context Aware, Situation-appropriate Reminder System. CHI 2000 Workshop on Situated Interaction in Ubiquitous Computing. 2000.
24. S. Fertig, E. Freeman, D. Gelernter. "Finding and Reminding" Reconsidered. SIGCHI Bulletin. Vol. 28. 1996
25. N. Marmasse. comMotion. Extended abstract in Proceedings of CHI'99. 1999. pp 320–321.
26. G. Korteum, Z. Segall, T.G.C. Thompson. Close Encounters: Supporting Mobile Collaboration through Interchange of User Profiles. Proceedings of HUC'99. 1999. pp 171–185.
27. D. Anind, A. Gregory. CybreMinder: A context-aware system for supporting reminders. In Proceedings of Second International Symposium on Handheld and Ubiquitous Computing, HUC 2000. pp 172-186.
28. Foundation for Intelligent Physical Agents. FIPA Communicative Act Library Specification, Version H, http://www.fipa.org/specs/fipa00037/.
29. J. Hendler and D. McGuinness. The DARPA Agent's Markup Language. IEEE Intelligent Systems, 15(5):34-43. May 2000.
30. M. Schalk, T. Liebig, T. Illmann, F. Kargl. Combining FIPA ACL With DAML+OIL – A Case Study. Proceedings of the 2nd International Workshop on Ontolgies in Agent Systems, Bologna, Italy, July 2002.