

# Contents

Session I This is a technical session

1  
CIA - an Infrastructure for Personal Agents  
*Frank Kargl, Torsten Illmann, Michael Weber*

3



Session I

This is a technical session



# CIA - A COLLABORATION AND COORDINATION INFRASTRUCTURE FOR PERSONAL AGENTS

Frank Kargl  
Torsten Illmann  
Michael Weber

Distributed Systems Department  
University of Ulm, Germany  
Albert-Einstein-Allee 11  
89081 Ulm, GERMANY

frank.kargl@informatik.uni-ulm.de  
torsten.illmann@informatik.uni-ulm.de  
weber@informatik.uni-ulm.de

**Abstract:** Software agents are one of the means to relieve users from searching, sorting and filtering information in their daily work. For fast and efficient implementation of agents there needs to be a powerful infrastructure supporting collaboration and coordination between agents and their users. In this document we propose such an infrastructure where the main focus is on modularity, efficiency, realtime-communication and resilience. Typical scenarios that can efficiently be realized with our system are diary-systems with integrated room-reservation service, team-assembly and workflow systems or even video-conferencing with advanced conference establishment. We introduce the concept of a so called Agent Cluster which contains all personal agents of one user and can be segmented into subclusters. This cluster is build around a software bus and contains many supportive agents for standard tasks. The current status and the on-going development of our project are discussed at the end of the paper.

**Keywords:** Distributed Systems, Middleware, Software Agents, Software Bus, Computer Supported Cooperative Work (CSCW)

## 1 INTRODUCTION

With the advent of modern communication systems, users often face a dramatic change of their working behavior. Today the question isn't anymore *if* some kind of information is available somewhere or *if* a communication partner can be reached online but rather *where* the information is stored or *where* the partner is located. A lot of time is spent with browsing, searching, sorting and filtering of information. It is commonly believed that intelligent and personalized software agents are a way for solving the resulting problems and making daily work more efficient. Agents are proactive, personalized, adaptive and autonomous software entities that work at request of their user, take the initiative and submit intelligent proposals for problem solving [1]. Especially in environments with a lot of cooperation between participants their agents need to cooperate themselves. In order to facilitate the cooperation and ease the design and implementation of single agents, a common environment for collaboration and coordination as proposed within this paper would obviously be helpful. Solutions to common tasks should be delivered by the infrastructure instead of being reinvented within each agent.

All agents belonging to one user form the so called Agent Cluster where every single agent represents a certain aspect or skill of its user. Within such a cluster there are powerful services that support the developer of agents at various common tasks like communication, persistent storage of objects, security aspects etc.

## 2 SCENARIOS

While developing our architecture we have several application scenarios in mind which influence our design decisions. A very basic scenario that doesn't impose too many difficulties is a distributed calendar service where people use diary agents to negotiate dates with each other. Additional value is added by services like automated conference-room reservation using a room-reservation service. Figure 1 shows a typical communication pattern in such a scenario. First the originating Agent Cluster (B) contacts a directory service in order to lookup the cluster of user C (step 1). Then a direct connection between the two clusters can be established and a date is arranged (step 2). Next cluster B asks a trader service to lookup the room-reservation service (step 3). Finally cluster B can request a room reservation (step 4).

Next we plan to realize a team-assembly- and workflow-scenario where agents can be used for fast and dynamic assembly of workgroups that can be coordinated using the workflow service. This requires intelligent white- and yellow-page services for location of task specific services and users, based not only on names but also on capabilities and dynamic roles within the workflow.

Finally we will integrate audio-/video-conferencing and cooperative application support in our system. This requires efficient communication systems, so we have to deal with real-time and QoS enabled middleware. Again we need naming services and search strategies for assembling conferences and the like.

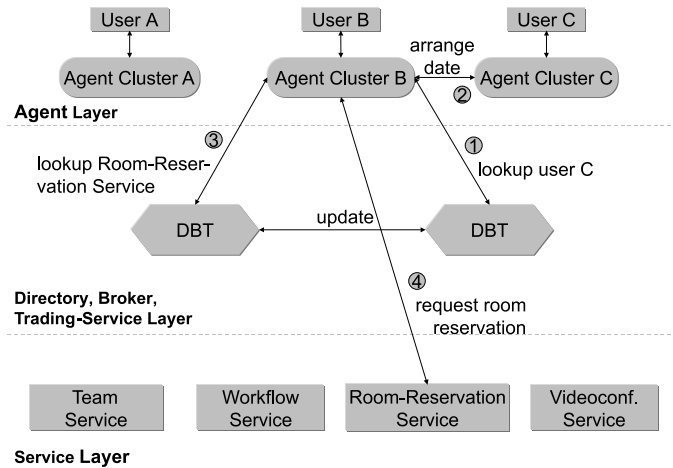


Figure 1 The 3-Layer Model

### 3 SYSTEM ARCHITECTURE

#### 3.1 3-Layer Model

As shown in Figure 1 our agent infrastructure is composed of three logical layers that will be discussed in the next subchapters. All agents belonging to one user are combined in a so called Agent Cluster that resides in the agent layer. The directory, broker and trader services are situated in the so called DBT-Layer. Finally there's a service layer, where task specific services like the room-reservation service mentioned above can be found.

#### 3.2 The Agent Cluster

Each user builds his own Agent Cluster tailored to his specific needs. For example most users will probably want to use a diary agent whereas not everyone needs an agent for management of a frequent flier account. Agents can dynamically enter or leave the cluster, driven by user request or by one of separate control agents present in each cluster.

Within a cluster agents can communicate via a software bus which we call Agent Bus. The bus is divided into separate subject-specific channels. Clients can join or leave these channels in a dynamic manner and events sent to a specific channel are seen by all its participants. Synchronous request-reply style communication is implemented on top of the Agent Bus. Currently we examine different realizations for such a software bus e.g. based on CORBA's Event Channel [2], Java's Jini Technology [3] or iBus [4] particularly with regard to performance, realtime aspects, resilience and expandability.

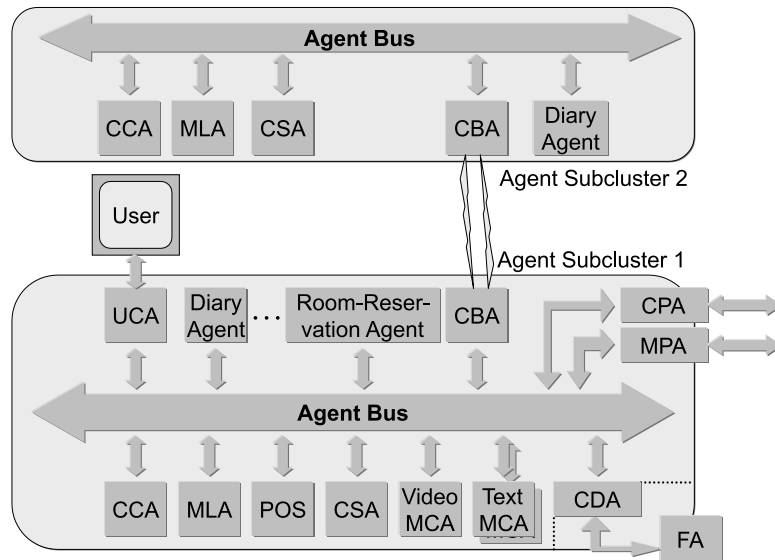


Figure 2 The Agent Cluster

At the same time we investigate whether we should use events or text message for agent communication and for communication between clusters and services. Our first prototype uses a message format similar to IETF's RFC822 used with SMTP. Now we are building a second prototype based on the iBus technology so we will switch to an event oriented communication scheme within the whole system.

### 3.3 Supportive Agents

Supportive agents e.g. for management, logging or persistent object storage are provided to keep the functional agents small and easy to implement. We try to keep the whole architecture as modular as possible and use a separate agent for every basic task. As every single aspect of our system is represented by its own agent, we can best describe the functionality of our architecture by describing some of the different agents contained.

The *Cluster Control Agent* (CCA) manages the Agent Bus. It controls the initialization of the whole system, starting of new agents or stopping of unneeded ones. When e.g. a user wants to negotiate a date and no diary agent is running, the CCA detects the request, starts one and asks the MLA (see below) to deliver the message once again.

There's a *Message Log Agent* (MLA) responsible for collecting and logging all events on the Agent Bus. The MLA can replay selected events which is important in respect to delayed agent startup (see above) or cluster splits discussed later.

Agents can store themselves or the data they produce using the *Persistent Object Space* (POS), which is again another agent responsible for storing serialized objects. We consider using JavaSpaces as a technology base for this task.

The so called *Cluster Security Agent* (CSA) implements all authentication, authorization and encryption functionality needed for secure and private communication.

In special cases it is desirable to move agents to other clusters, so they can fulfill their function at the remote place and come back later with a compressed result thus reducing the network load. There's a special *Cluster Docking Agent* (CDA) that allows a secured access for mobile agents (we call it Foreign Agents - FA) similar to the sandbox concept for Web-Applets.

Special *Media Conversion Agents* (MCAs) will solve the problem of incompatible media formats. There's one special agent for any major media category (like text, video etc.). When connecting to other clusters the relevant MCAs first try to negotiate the available media formats available on each side of the communication. The "best" format available on both sides is chosen using a quality metric.

All the communication between the user and his agent cluster takes place by way of the *User Communication Agent* (UCA). This agent provides a very flexible interface that other agents can use to interact with their user. In our prototype we use a simple WWW-server and a specific Java applet for implementing the UCA. Agents can dynamically describe and modify their part of the graphical interface displayed by the applet. They can upload intrinsic GUI-components to the UCA. Events generated by user input in the applet are sent through a chain of event proxies that either handle the event locally or forward them to the next proxy in the UCA or finally to the agent. We have modified the Java event model so that remote event handlers can be called using RMI.

When an agent cluster consists of several agent subclusters these are connected via *Cluster Bridge Agents* (CBAs). The CBAs constantly monitor the links between them so they can detect net-splits instantly. In case of such splits they notify the CCAs so special services like the CSA that are vital for functionality are restarted in the vacant subcluster. After re-join, the CBA initiates the re-synchronization of all agents.

There are several *proxy agents* for communication between clusters or even for converting cluster events like dates from the diary agent into emails (e.g. with dates converted to vcalendar standard). These are called *Cluster Proxy Agent* (CPA; for cluster to cluster or cluster to service communication), *Mail Proxy Agent* (MPA; for email conversion) and the like.

### **3.4 The DBT and Service Layers**

In the DBT-Layer you find distributed white and yellow page services. Using these services, agents within a cluster or services can locate other agent clusters searching for special usernames or specifying certain skills that the communication partner should have. Of course services can be searched for, too.

All services not bound to a specific user are situated as external services in the Service Layer. Different scenarios where such services are used were described in section 2.

#### 4 STATUS AND ON-GOING DEVELOPMENT

Up to now we have designed the major parts of our infrastructure and implemented a very basic prototype consisting merely of a CCA, a CPA and a dummy-agent for performance tests etc. This prototype is implemented in Java and is using the CORBA event channel as the implementation base for the agent bus.

Our ongoing design- and implementation-work will lead to new prototypes implementing more of the mentioned supportive agents and thus adding more functionality to the framework. By spring '99 we want to finish a first real-world application, a distributed calendar with room-reservation functionality based on a diary and room-reservation agent and service. Advanced scenarios like workflow management and a conferencing system will be studied afterwards.

Our focus for the ongoing research will be on security issues (authentication and esp. authorisation for agents), on resilience and distributed operation of our agent cluster (applying networking techniques like repeaters, bridges and routers to software buses) and finally on the implications of real-time processes (videoconferencing, workflow) for agent systems.

#### References

- [1] Maes, P. *Humanizing the Global Computer*. IEEE Internet Computing, July-August 1997.
- [2] Object Management Group *CORBA services: Common Object Services Specification*. available from <http://www.omg.org/library/>.
- [3] Sun Microsystems *Jini*. from <http://java.sun.com/products/jini/index.html/>.
- [4] Softwired AG *iBus*. from <http://www.softwired.ch/products/ibus/>.

#### Biography

**Frank Kargl** studied Computer Science at the University of Ulm with focus on distributed and parallel systems. From 1997 to 1998 he worked for the University Computing Centre as network administrator. In 1998 he started to work for the Distributed Systems Department. His major research interests are agent systems and software buses.

**Torsten Illmann** studied Computer Science at the University of Ulm with focus on distributed systems and artificial intelligence. From 1997 to 1998 he worked for the company debis Systemhaus GEI GmbH as software developer. In 1998 he began to work for the Distributed Systems Department. His major interests are agent and workflow management systems.

**Michael Weber** Michael Weber holds a Ph.D. in computer science. Since 1994 he is professor for distributed systems at the university of Ulm. Previously he has been in industry for several years. His research interests are in distributed computing, middleware, multimedia systems and CSCW.