An Architecture for Automated Information Retrieval

Hans-Günter Stein, Michael Breu

FAST e.V.
Arabellastr. 17
D-81925 München
{hgs | breu}@fast.de

Abstract

Mobile agent technology can be employed to do automated information retrieval. But there are not many practical models available on how a system of interacting agents and services can be organized. This paper introduces the architecture that is used in the FollowMe pilot applications to establish a system of agents, service providers and contents providers to assist the user in his information management.

Motivation

Efficient information management and retrieval is becoming the key success factor to almost any human activity – be it in business or private life. The well known problem of 'information overload' is omnipresent in the Internet. Thus the challenge we face is to structure the contents of the Web and to develop easy to use tools that assist the average information consumer in locating the information sources that best fit his needs.

Referring to this challenge, L. Daigle stated (Daigle 1995):

> ...we need to have standard mechanisms for encapsulating client information processing tasks.... Since the activity [*of information retrieval and processing*] is client-centered, however, not resource-centered, it is not appropriate for it to be buried in information servers, either. Thus, there is a new layer of information processing objects, acting as agents for clients....

Several architectures, based on intelligent agent technology are proposed to solve this problems.

Simple structured architectures propose application specific solutions that use personalized agent technology deployed either on the client or the resource provider host (Caglayan and Harrison 1997, Maes 1997).

Recently more sophisticated architectures based on multi-agent systems have been proposed that deploy several types of interacting agents (Genesereth and Ketchpel 1994, Sycara et al. 1996, Hermans 1998). In (Sycara et al. 1996) these agents represent the different partners in an information retrieval task (e.g. the user, or the information source), but there are also so-called task agents that are responsible for the mediation between both partners.

The approach presented in this paper is similar to that in (Sycara et al. 1996), however it adds another level of genericity to allow a dynamic plug-in of new services and even more of new service types. In this paper we outline an agent interaction framework that facilitates an information brokering and information retrieval structure. Users can customize agents that serve as interfaces to information services. These agents are potentially mobile so they may move whenever appropriate depending on the specific tasks they are designed to service. The system supports location transparent agent and data addressing giving both users and agent developers maximum flexibility with respect to mobility.

The project is building up a support technology for a mobile agent infrastructure and proves the technology in several major pilot applications. Partners in the project are APM Ltd., the University of the West of England (UK), Inria, TCM (F) and FAST e.V. (D).

In the following chapter we outline the technical mobile agent framework that is built up by FollowMe. This is followed by a presentation of the logical infrastructure of how agents interact with services over the network on behalf of their user. This is illustrated by an example scenario. The paper is concluded by an overview of the pilot applications under preparation.

The FollowMe Framework

In this chapter we give a short overview to the components of the FollowMe framework.

Design Guidelines

The Design of the FollowMe Agent System is guided by the following principles:

Portability: The mobile agents should be able to move between a variety of host types and operating systems, i.e. the code has to be as portable as possible. Therefore the coding language of choice was Java™ (for other examples of mobile agent systems based on Java™ see e.g. (White 1996, Lange and Oshima 1997).

Scalability: The applications may require a mass of agents to run on a single location. The system should be

able to cope with this and the overhead for running agents should be small.

Remote User Access: A mobile agent runs primarily unattended and must be able to contact his user while he is offline. This can be done by means such as sending a fax, an SMS message, or making a phone call with some synthesized audio information.

Autonomy: An agent is autonomous in its decisions, i.e. it is free to react on external events, e.g. a request to move to another location.

Security and Privacy: An agent may run in a location that is not necessarily trusted. On the other hand a host may not necessarily trust every agent. There must be mechanisms to identify trusted hosts, especially hosts that guarantee not to dissect an agent, and to identify trusted agents, that are allowed to use the resources of a host.

Interoperability: Agents can inter-operate with each other or with a (non-mobile) service either by moving to a common location or by communicating over the network.

The FollowMe Architecture

Logically the FollowMe architecture consists of the following modules (figure 1):

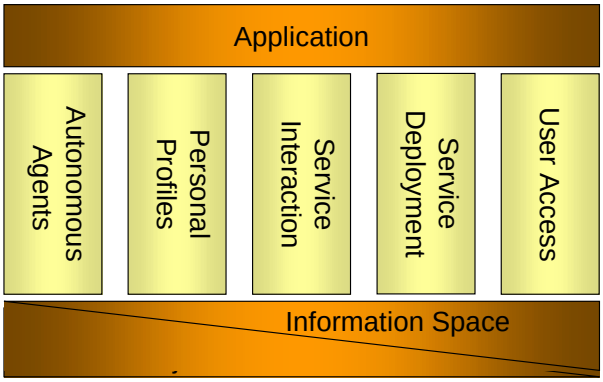

Figure 1: Architectural Framework Components

The *mobile object workbench* (MOW) provides the basic infrastructure for mobility. A mobile object resides at a "place". A place provides the minimal environment for a mobile object to live in. It also provides mechanisms for location transparent references for mobile objects.

A place itself can be mobile, e.g. when it resides on a notebook, it can disconnect from the internet and reconnect at another geographical location. This can be a typical case for a mobile user. For a detailed description of the MOW see (Hayton et al. 1998).

On top of the workbench *autonomous agents* are running. Autonomy means that an agent is free on how to react on external events. The only decision a place can enforce on an agent is its destruction. The mission of an agent is defined in an agent script. An agent is equipped with a *profile* that holds its characterizing data.

Agents interact with each other. The interaction with (non-mobile) services is provided by service interaction interfaces that act as service proxies. These proxies can be contacted by agents to query a service. An agent can also subscribe to a service, in order to be informed when a certain state change occurs.

A special type of agent is a *personal assistant*, which is directly associated to a user. The personal assistant manages a *personal profile* that holds all relevant information about the user. The personal profile contains simple items like the user's name and address, but also a diary of its user, i.e. a schedule defining via which devices a user can be contacted, in order to send him a fax or an SMS message. The persistent data components of all profiles in the FollowMe environment are implemented in XML (W3C 1997), all agents are capable of parsing XML and all instructions on how to interpret XML data stored in the profiles are implemented using a scripting language. User interfaces to XML data are described by XSL style-sheets (Microsoft 1998).

The i*nformation space* component provides services to persistently store objects and retrieve them in a distributed environment. The Information Space may provide replication services for objects in order to optimize access from different locations. The information space also provides store and backup facilities to be able to recover mobile agents after a system crash.

The s*ervice deployment* component applies data mining techniques to optimize the availability of objects at different locations, i.e. to exploit the replication service provided by an information space.

Finally the *user access* allows a mobile agent to contact its user. Communication media could be e.g. a Web browser, a personal digital assistant (PDA), a fax, or phone. Two types of devices are distinguished:

- Output-only devices: These devices only support system output (i.e. sending a fax or a phone message), thus the agent is able to send reports to the user or just the request to contact him via an interactive device for further interaction. These devices are used for off-line communication with the user.
- Interactive devices (i.e. Web browsers): These devices are used to interact with FollowMe agents, e.g. via HTML forms. Through these devices users can typically command their agents.

Communication is always initiated by an agent, i.e. the agent decides to send the user a message (e.g. a fax, or a Web page). The only exception is that if the user is on-line, he is able to send the agent a "I'm on-line"-event through dedicated entry-points of a FollowMe place.

As all other system components, the user access component is designed with respect to scalability. Generic methods allow the integration of new device types without major changes to the system design.

The applications themselves are built on top of these components. In the next section we will outline a logical architecture of how a service infrastructure can be built up and exploited by mobile agent technology.

Agent Interaction Framework

In this section we describe the concepts that form the framework for agent interaction in a FollowMe scenario. The FollowMe agent interaction framework was designed to enhance information retrieval and information filtering in large scale, heterogeneous networks like the Internet.

The Internet's nowadays major application, the World Wide Web, is a loosely coupled and fairly unstructured environment composed of millions of information consumers and providers. Anyone familiar with the Web knows of its major contradiction: *Whatever information you look for is available somewhere on the net, but locating the appropriate information sources is extremely time consuming and therefore very costly.*

Any solution to this contradiction should aim to leverage the existing infrastructure of the Web to the worlds largest, truly distributed database. In approaching a solution towards this aim, we identified the following core issues:

- The use of any database is not only defined by the sheer amount of collected data, but by the applications or services that operate on the contents of the database to provide information in form of customized results to the users of such systems.
- In order to enable the development of useful database applications, any database needs to provide a meta-model describing the structure of the offered data.
- To gain the most from largely distributed databases, the development of database applications should be de-coupled from the maintenance of the databases themselves. That way, the contents of data sources can be re-used and re-combined when developing new applications according to the needs of information consumers.

A first analysis of above issues leads to the most important design decision for the FollowMe agent interaction framework: *to de-couple the roles of service providers and content providers*. *Service providers* implement applications that make use of raw data offered by *content providers*. They define meta-models describing the data structures their applications are capable of dealing with. In order to enable the *service providers*' applications to make use the data offered by a *content provider*, the data needs to be structured according to meta-models that form supersets of the meta- models of the *service providers*. This decision leads straightforward to the following core axioms:

- Users of a FollowMe system will no longer (as is with the Web) address *content providers* to obtain information in raw or proprietary data format. Instead they will address services that provide them with already refined information according to their individual needs. The services therefore need to be customizable by the individual user.
- Services do not operate on a predefined or hardcoded set of data sources, but on specific data structures. They may use any data source available at runtime that offers relevant data as long as it offers an interface the service knows how to use.

These axioms impose the introduction of components *that glue things together*. On the one hand, users need to have an effective way of locating services that fit their needs. On the other hand, there need to exist mechanisms that enable services to locate relevant information sources at runtime. The appropriate concepts to fulfil these requirements are the ideas of brokers, matchmakers or – more simply – directory services.

Thus the core components of the FollowMe system have been identified as (see figure 2):

- *content providers* (offering access to data-objects),
- *service providers* (providing services operating on data available from the *content providers*),
- *information consumers* (users of available services) and
- *directory services* (mediating between the other components).

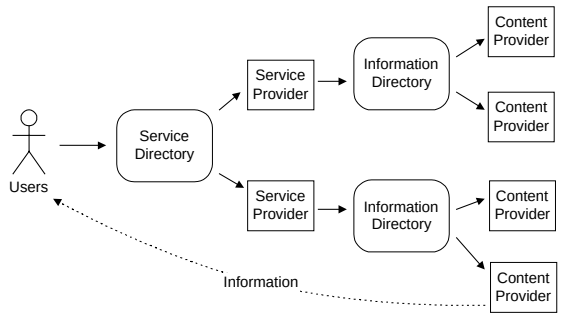

Figure 2: Core components of the FollowMe system

FollowMe uses agent, scripting and profiling concepts to build up an agent interaction framework that enables the deployment of agent driven applications meeting the above stated needs.

In FollowMe a service is composed of a component related to the information consumer (referred to as *task agent*) and a component implementing an interface to content providers (referred to as *service interaction interface*). A special user related agent (referred to as *personal assistant*) assists the user in organizing the usage of services and handling personalized information.

In most client/server or network centric applications currently available on the Internet users connect to remote applications and specify their interests in form of application specific parameters. All computation is done on the server side and results are delivered to the clients. This enables application providers to gather lots of personalized information about users of their system. In contrast, within the agent based approach of the FollowMe system, the user's privacy is respected by hosting all user related information and computation in a location close to the user and in an environment trusted by the user (see concepts of *profiles* and *information spaces* in the architectural framework section). All agents acting on behalf of a user in

FollowMe are instantiated on request by downloading the respective Java classes from so called *agent factories* to the user trusted environment (referred to as *FollowMe places*). The environment is located on a host with permanent online connection (i.e. a local ISP).

Scenario description

To illustrate the basic concepts of the FollowMe system we describe a fairly generic scenario. We assume, that the user in the described scenario already owns a *personal assistant* and is now on his way to select one of the services offered by FollowMe's *service providers*. As an example we introduce a service that delivers information on regional events (i.e. concerts, cinemas, markets, exhibitions). This service has already been deployed in a large scale practical field test (see *conclusion* section).

Step 1:

A user connects to the FollowMe system by contacting his *personal assistant* (*PA*). The user may then make changes to his personal *diary*, i.e. by stating, that he will be reachable by e-mail during working hours and by fax otherwise (see figure 3).

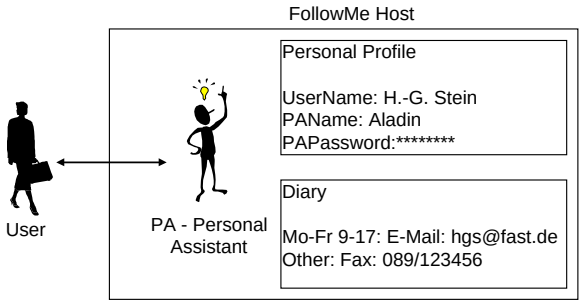

Figure 3: Contacting the *personal assistant*

Step 2:

The user wants to use one (or more) of the services offered by FollowMe.

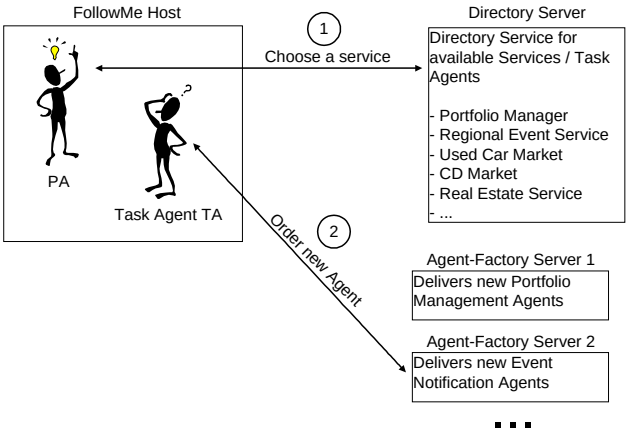

Figure 4: Instantiation of a new *task agent*

The *PA* connects the user to a *service directory* that allows the user to select a specific service the user is interested in. Note that the *PA* does not require any knowledge about specific services. This de-couples the user specific components (such as the *PA*) from the rest of the system (available information sources and services). After selecting a service to subscribe to, the *directory service* links to the appropriate *service provider* (*agent factory*) and an instance of a *task agent* representing the respective service is created on the FollowMe host to service the individual request of its owner (see figure 4).

Step 3:

The user may now provide the new *task agent* with personalized parameters. In case of an event notification service, parameters might include specific event types of interest to the user and location and date of events. Moreover the user specifies a time schedule defining when he wants the results of the service to be delivered (see figure 5).

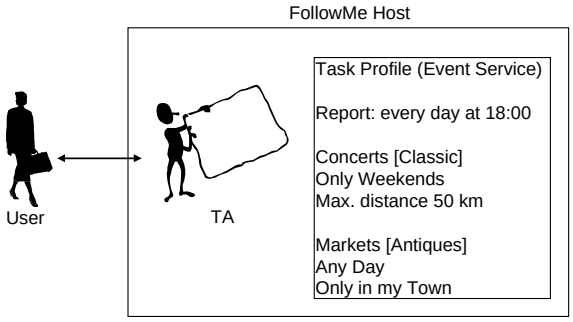

Figure 5: Defining a *task profile*

Step 4:

After specifying these input parameters the user may disconnect from the system. The specified service will execute automatically and in regular intervals according to the user defined schedule. The agent in charge of executing the respective task will contact another *broker* (*directory service*) that links it to *service interaction interfaces* at *content provider* sites relevant to the application domain of the *task agent* (in our example these are providers of information on regional events). All the agent needs to know is the type of interfaces it is capable of connecting to. There is no need to hardcode the addresses of *content providers* within the agent. This de-couples *service providers* (*agent providers*) from *content providers*. That way, new *content providers* could join the system by registering at the *directory service* without the need for changes to existing services. The same holds for the integration of new services operating on data offered by existing *content providers*. In our example the agent connects to servers providing information on regional events and queries these servers according to the user specified parameters.

The directory service holds meta-data for each service. Based on this meta-data a task agent can optimize the

selection of content providers. In the domain of regional events the meta-data of a service may contain geographical information on what region is served. By this information the agent could rule-out content providers that do not serve events relevant for its query.

As described in the architectural framework, all objects and thus agents are potentially mobile. *Service interaction interfaces* are capable of providing an agent runtime environment (a FollowMe *place* – see architecture framework). Whether an agents makes use of these mobility features depends on the type of service the agent is representing. In applications where communication between a *task agent* and a *service interface* is very intensive (i.e. sophisticated negotiation processes) the agent could be designed to move to the interface instead of remotely connecting to it (see figure 6):

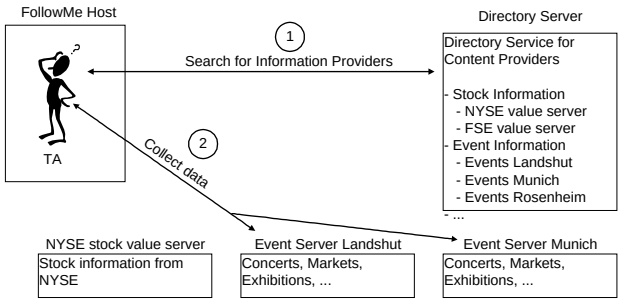

Figure 6: Task execution

Step 5:

After the *task agent* finished its task of information retrieval and refining, it stores the information in the user's *information space* for future reference either by the user or by the agent itself (see architecture framework). In addition the agent might be instructed to deliver reports on new information to its user. In our example the user instructed the agent to send a report every day at 18:00. Reporting might as well be triggered by specific changes to data values or other events (i.e. a stock portfolio management agent might be instructed to report to the user immediately when a share value exceeded a specified limit).

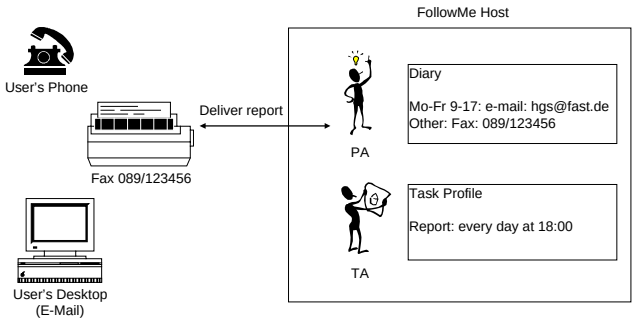

Figure 7: Delivering a report

To send reports to its user, the *task agent* uses the *user access components*, which provide gateways to a variety of devices like mail boxes, phones, pagers or fax machines (see architecture framework). The kind of device to be used for report delivery is stated in the diary section of the user's *personal profile*. The *task agent* consults the *personal assistant* to obtain this information. In our example the appropriate device for delivering reports at 18:00 is a fax gateway (see figure 7).

System components

With the previous scenario description in mind we now take a closer look at FollowMe's system components with special focus on agents (figure 8).

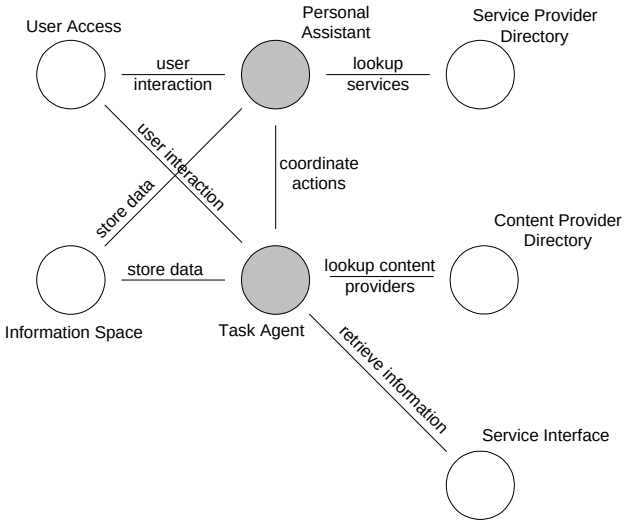

Figure 8: FollowMe system components

Personal Assistant: The *personal assistant* in figure 8 is in fact composed of a number of components (figure 9).

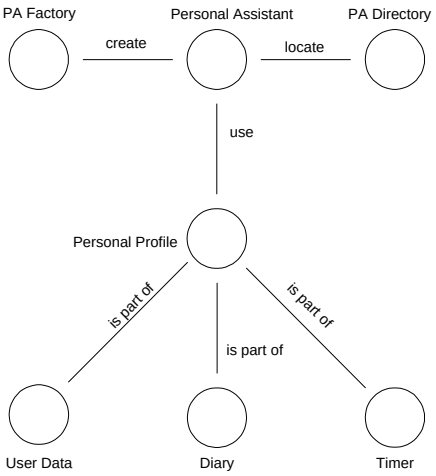

Figure 9: Personal Assistant Components

A *personal assistant* (*PA*) represents a single user of the FollowMe system. The role of the *PA* is to assist the user in organizing the usage of services. Attached to the *PA* is the user's *personal profile*. This profile is used to store persistent data about the user. Basically user related data consists of the user's name and address, the user's system access password and a list of the services the user is currently subscribed to. Since all persistent data is stored in XML (see architecture framework), profiles can be easily extended according to the needs of the evolving system.

Besides basic persistent data about the user, the *personal profile* provides a *diary* functionality that enables the user to specify how the system may contact him at different points in time (see description of the *user access component*).

Agents in FollowMe act on behalf of the user while the user is offline. The system therefore provides mechanisms that enable the scheduling of task execution and reporting. This mechanism is provided by a *timer* component as part of the *personal profile*.

New users might join the FollowMe system by requesting the instantiation of a new *personal assistant*. A new *PA* is created by instantiating the respective Java classes from a (remote) *personal assistant factory*.

To allow users to address their *PA* by the *PA*'s name (i.e. *MyAgent Aladin*), there exists a *PA directory* mapping *PA* names to object references.

Task Agent: The *task agent* in figure 8 is composed of components similar to those of the *PA* (see figure 10).

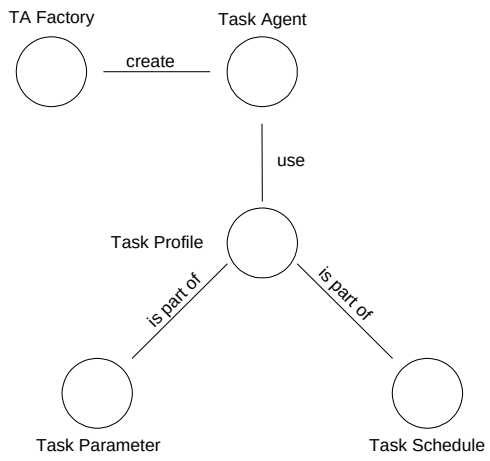

Figure 10: Task Agent Components

Task agents are agents offering application domain specific functionality. They are capable of using services with domain specific interfaces. The tasks of an agent are described in terms of *missions* programmed in a scripting language. *Task agents* can communicate using the message passing protocols provided by the underlying *mobile object workbench* (see architecture framework).

Interfaces to services are described in CORBA-IDL. Besides specifying interface types, the interface descriptions provide information on update schedules of the *content providers*' databases (i.e. it doesn't make sense for an agent to query a *content provider* for new information every five minutes when the data sources are updated only once a day).

Attached to a *task agent* is a *task agent profile*. The profile contains user specified task parameters and scheduling information defining when to execute which tasks.

Task agents are created by instantiating the respective Java classes from a (remote) *task agent factory* located at a *service provider* site.

Component Interaction: As outlined in the scenario description, the user communicates with agents via the *user access* through a variety of different device types. User interaction is required to provide the agent system with personalized instructions on how and when to execute certain tasks on behalf of the user.

Both agent types, *personal assistants* and *task agents*, may store the results of their information retrieval activities (that is data objects retrieved from *content provider* sites) in a user's *information space*. Access to stored objects is via location transparent object references. The *information space* provides access control mechanisms to ensure that data can be accessed only by authorized agents.

The activities of a user's *task agents* are coordinated by the *personal assistant*. The *PA* keeps a record of all active *task agents*. Tasks are triggered by the *timer* component in the *personal profile*.

Personal assistants link their users to the directory of available services whereas *task agents* contact *content provider directories* to determine which data sources (in form of *service interfaces*) to use. This again demonstrates the concept of separating *service providers* from *content providers* and thus freeing the user from having to deal with widely distributed raw data.

Conclusion

This paper described the overall architecture and the agent interaction framework developed as part of the FollowMe project. It can be a basis for a clearer structuring of information available on the Internet. In particular, the separation between service providers and content providers allows a more flexible way to tailor information services, and to provide adaptation between information sources and user needs.

The concepts developed in FollowMe are currently evaluated in a large scale practical field test. Fast e.V. developed and recently (January 99) deployed two pilot applications in cooperation with the Bavarian Online user organization. The Bavarian Online user organization was chosen as a partner since they provide a large user base and a distributed network of places where FollowMe agents can run.

The intention of the pilot applications developed by Fast e.V. is to demonstrate the practical applicability and impact of such a technology and to validate the agent interaction

framework. Usage monitoring and collecting of user feedback is currently in progress and will be made publicly available as part of the FollowMe project's final evaluation report.

The first pilot application is a stock portfolio management system. Users can invoke a task agent that manages their stock portfolios. They can define upper or lower limits of share values. If theses limits are met, the user is informed by a fax or phone call. Share values can be retrieved by a variety of content providers as e.g. quote.yahoo.com for the NYSE, and finanzen.yahoo.de for the European stock exchanges.

The second pilot application is a regional events notification system as described in the *scenario description* section of this document. Local providers advertise their events in their regions (e.g. concerts, exhibitions, trade fairs). Users can engage a task agent to regularly lookup events of a certain type in a certain region.

The Bavaria Online user organization is planing to implement further FollowMe-based applications even beyond the scope of the FollowMe project (ending March 99). Application domains of interest are i.e. market places for books, CDs, used cars, real estates or any other domain where information of a specific type (or data structure) is distributed over multiple information sources.

Since FollowMe is designed to be truly scalable, follow up projects could use our agent interaction infrastructure to implement more complex scenarios that require more agent to agent or agent to service communication. FollowMe agents could not only act on behalf of user but as well on behalf of other agents or legacy systems.

Acknowledgements

The work described in this paper is based on the results of the FollowMe project. Therefore we want to acknowledge the work of the whole FollowMe Team. FollowMe is sponsored by the European Commission in its ESPRIT program under project number 25,338. Information about FollowMe is available under:

http://hyperwave.fast.de/FollowMe.

References

Caglayan, A., and Harrison, C. 1997. *Agent Sourcebook*. Wiley.

Daigle, L. 1995. Position Paper. ACM SigComm'95 – MiddleWare Workshop.

Genesereth, M.R., and Ketchpel, S.P. 1994. *Software Agents*. Communications of the ACM, 37(7):48-53.

Hayton R., Bursell M., D. Donaldson, A. Herbert 1998. *A Mobile Object Workbench*. Mobile Agent Workshop '98. Stuttgart.

Hermans, B. 1998. *Desperately Seeking: Helping Hands or Human Touch*.
http://www.hermans.org/agents2/index.html

Lange, D., and Oshima, M. 1997. *Java Agent API: Programming and Deploying Aglets with Java*. Addison Wesley.

Maes, P. 1997. *On Software Agents: Humanizing the Global Computer*. IEEE Internet Computing, I(4): 10-19.

Sycara, K., Decker, K., Pannu, A., Williamson, M., D. Zeng 1996. *Distributed Intelligent Agents*. IEEE Experts Dec. 96.

White, J.E. 1996. *Mobile Agents*. Menlo Park, CA, AAAI Press/MIT Press. in Bradshaw, J.(ed.). *Software Agents.*

XML Specification Version 1.0 1997. W3C.
http://www.w3.org/TR/PR-xml-971208

XSL Tutorial 1998. Microsoft.
http://www.microsoft.com/xml/xsl/tutorial/tutorial.htm