An Approach to Agent-Based Service Composition and Its Application to Mobile Business Processes

Michael Berger, Makram Bouzid, Mark Buckland, Habin Lee, Nicolas Lhuillier, Dieter Olpp, Jérôme Picault, and John Shepherdson

Abstract—This paper describes an architecture model for multiagent systems that was developed in the European project LEAP (Lightweight Extensible Agent Platform). Its main feature is a set of generic services that are implemented independently of the agents and can be installed into the agents by the application developer in a flexible way. Moreover, two applications using this architecture model are described that were also developed within the LEAP project. The application domain is the support of mobile, virtual teams for the German automobile club ADAC and for British Telecommunications.

Index Terms—Business to worker, multiagent system, multiagent architecture, generic service component, field trial, virtual team.

1 Introduction

One of the basic requirements for agents is to accompany their human users, while fulfilling certain tasks for them. As the tendency to carry and use mobile devices increases, agents should be present on such devices too, given their ability to communicate via some form of mobile Internet. An area where this idea can be directly applied is the work of mobile, virtual teams. This was the starting point of the European project LEAP[1] (Lightweight Extensible Agent Platform) [5], which ran from January 2000 to June 2002. It consisted of three main activities:

First, the agent platform JADE [3], [6] was enhanced and modified to form JADE-LEAP, the first FIPA-compliant agent platform that runs on small devices like PDAs and mobile phones. This effort has been described elsewhere in detail [1], [2].

Second, an architecture model was developed that is based on "generic services." A set of such services for the application domain of mobile workforce support was implemented.

1. Supported by the European Commission by grant IST-1999-10211. Consortium members were: Motorola S.A., Paris, France; ADAC e.V., Munich, Germany; Broadcom Eireann Research Ltd., Dublin, Ireland; British Telecommunications plc, Ipswich, UK; TILAB S.p.a., Turin, Italy; University of Parma, Parma, Italy; Siemens AG, Munich, Germany.

- *M. Berger and D. Olpp are with Siemens AG, CT IC 6, Otto-Hahn-Ring 6, 81730 Muenchen, Germany. E-mail: {m.berger, dieter.olpp}@siemens.com.*
- *M. Bouzid, N. Lhuillier, and J. Picault are with Motorola Labs Paris, Parc Les Algorithmes, Saint-Aubin, 91193 Gif-sur-Yvette Cedex, France. E-mail: {makram.bouzid, nicolas.lhuillier, jerome.picault}@motorola.com.*
- *M. Buckland, H. Lee, and J. Shepherdson are with BT Exact, Intelligent Systems Lab., pp MLB1-12, Orion Building, Adastral Park, Martlesham, Ipswich, Suffolk IP5 3RE, United Kingdom. E-mail: {mark.buckland, ha.lee, john.shepherdson}@bt.com.*

Manuscript received 14 Jan. 2003; accepted 29 Apr. 2003.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number 7-012003.

Third, two applications were written on top of these services and tested under real-life conditions in two field trials in April and June 2002. These applications supported mobile, virtual teams of German automobile club ADAC, whose so-called "Yellow Angels" offer roadside assistance to club members, and of British Telecommunications plc (BT), whose "Survey Officers" evaluate the telecommunications infrastructure needed for new and existing BT customers.

This paper is about the generic services and the field trial applications. In Section 2, we explain the abstract architecture model and the process of building a multiagent application for the JADE-LEAP agent platform from the generic services. In Section 3, we present the generic services concretely implemented. Finally, we give an overview of the two applications and the set-up of the corresponding field trials for ADAC (Section 4) and BT (Section 5).

2 The Generic Services Architecture Model

The objective of the generic services is to ease the development of agent services by providing generic building blocks, which can be assembled and customized to create complex applications. These generic service components mainly handle most of the agent-related concerns (protocol, conversation, language, ontology, and errors), while allowing the developer to concentrate on the application logic.

The architecture of generic services relies on two main components: the *Initiator* and the *Respondent*. The main motivation is to decouple as much as possible the service from the application, to ease the integration of agent services within applications, and to increase reusability of service components. Note that the arity of the interaction

 Published by the IEEE CS, CASS, ComSoc, IES, & SPS

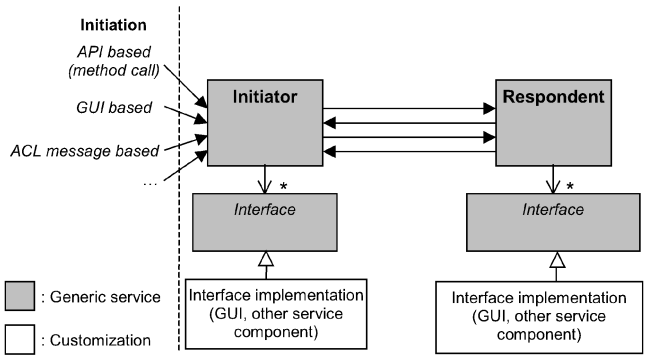


Fig. 1. Initiator/Respondent architecture.

between Initiator and Respondent is 1 to n, i.e., one Initiator interacts with one to many Respondents.

The Initiator is used to access the service. The term "Initiator" has to be preferred to "client" because the Initiator's role is first to start the conversation, but it can also be involved in the decision process and the service provision.

As agents communicate on a peer-to-peer basis, it is possible (but not compulsory) for both Initiator and Respondent components to be installed within the same agent. Hence, the architecture of the application and the deployment of the service components are free to have whatever configuration is suitable for the application. This is particularly useful when a part of the application has to run on lightweight (i.e., computing resource constrained) devices because it allows deploying only some part of the interactions on the lightweight devices and the rest on more "heavyweight" (i.e., powerful) devices.

Note that, though these components are built to be independent from the agent they are plugged into, they cannot be used without being installed within an agent because they have no intrinsic communication capabilities and need to rely on those provided by the agent. FIPA agents [4] use an Agent Communication Language (ACL) to communicate, which defines the encoding, semantics, and pragmatics of message exchanges.

The functions performed by the Initiator are as follows:

- To find the appropriate responder(s), which can fulfil the service. By default, this is performed by searching the Directory Facilitator (DF) agent for a specific service registration. In fact, in FIPA specifications, the DF is the agent that provides a service lookup function for the agent platform.
- To return a directly usable result to the user of this service. This means hiding, as much as possible from the user, the complexity of the request and the different steps of the interaction. Using an API-based initiation (e.g., method call), the user requests the service and is returned a well-formed result. The generic service API allows all kinds of initiation (as shown in Fig. 1)—through a GUI, a method call, or after the receipt of a message.
- Finally, the Initiator component has to ensure that the agent that it is plugged into has the required capabilities to perform the request. In particular, this means the agent knows the appropriate content language and service ontology (which gives a meaning to the symbols in the content expression). In the best case, the service component should use the content language the agent already supports and install the missing resources if required.

From the other side, the Respondent component has to ensure the following roles:

- To advertise, via the DF agent, the kind of service it provides, so that initiator components are able to find and access it.
- To ensure that service requests are answered each time in the best possible way. Since the Respondent has no facility to receive messages itself, this implies modifying the agent behavior so that the agent effectively handles all request messages for the service, which will process them in turn.
- To make sure the agent this component is plugged into owns all the required resources (e.g., content language and ontology) to process the service requests. If possible, the component should provide the agent with the missing required resources or should return an error otherwise.

Note that the only prerequisite for an Initiator to interact with a Respondent is to have the information about the protocol, the content language, and the ontology used by the service. Hence, each generic service should include a detailed specification, at the FIPA level, to allow other standards-compliant parties to potentially interact with it.

3 Generic Services for Mobile Workforce Support

3.1 Customization of Generic Services

A key advantage that generic services offer is that they can be customized to fit an application's specific needs. Customization of generic services is performed through the implementation of the Initiator and Respondent interfaces. These interfaces are the means for developers to give the generic service all the information that the service cannot know beforehand because this information is application-dependent. The implementation of each interface can be by any type of object, including a GUI or even an initiator component.

3.2 Service Broker

In some architectures, it may be useful (or sometimes required) to use a *Broker* component between the Initiator and the Respondent. This can be:

- For performance reasons: There are many communications between broker and Respondent.
- For lookup specialization reasons: The broker knows about many more Respondents.
- For security reasons: The Initiator is not allowed to directly access the Respondents.

In our case, the broker is just a particular kind of generic service, whose role is to mediate the interaction between the

Initiator and the Respondent, especially in case of 1 to n interactions (i.e., *one* Initiator and n Respondents).

3.3 Composition of Services

By composition of services, we mean that the output of one service becomes the input for another service. This feature is a key to agent services as it allows developers to dynamically build groups of services in order to fulfil requests that were not considered beforehand. Note that this architecture does not provide the means to decide how and why a collection of services has to be built, but only defines the mechanisms by which elements of this collection, once defined, will be able to easily interact with one another. Service composition is achieved thanks to the Initiator and Respondent interfaces, which can be implemented by other initiator components, thus allowing the output of one service to be the input of another service.

3.4 Identified Generic Services

We have identified three categories of agent generic services relevant for the mobile workforce domain and, hence, used in the field trial applications, as follows.

3.4.1 Travel Management

Providing up-to-date information and guidance on travel planning. Ensuring travel time is minimized, thus saving resources and reducing traffic congestion. The Travel Management services anticipate a mobile worker's travel needs, providing guidance and time estimation so as to synchronize the movements of virtual teams working over vast geographic areas. Useful services that have been identified are as follows:

- *Plan Route*. Given two locations A and B, calculate a route between A and B, subject to any given constraints (e.g., shortest distance, least time taken, must pass through intermediate "waypoints," etc.).
- *Replan Route*. Following the initial generation of a route plan, the system identifies that the mobile worker is no longer on schedule. This may be due to a number of reasons: the work schedule being changed, new traffic information being received, the mobile worker being delayed, and so on.
- *Get Current Position*. Find the position of the mobile worker. This could be done by using a positioning system, e.g., GPS, triangulation, or by asking the user directly.
- *Estimate Route Cost*. Given a route consisting of a set of legs and using information about current conditions, calculate the cost of the route in terms of nominated dimensions such as time, mileage, etc.

3.4.2 Teamwork Coordination

Empowering individuals to collectively coordinate activities (e.g., by trading jobs, automatically negotiating for work, and expressing personal preferences) within an agreed policy framework; facilitating "buddying" between mobile workers where team members can exchange tacit knowledge, for example, between experienced and trainee workers. A list of services we have defined is given below:

- *Give Job*. Allows a mobile worker to give a job that has been assigned to him to another mobile worker based on a predefined policy (distance, skills, etc).
- *Swap Shift*. Each mobile worker has an attendance pattern that defines the shifts they will work. A mobile worker wants to swap a shift on some day for some other shift (on possibly the same day).
- *Auction Shift Swap*. The same as *Swap Shift*, but using an auction-based mechanism between mobile workers.
- *Request Expertise*. When a mobile worker has a problem that he cannot solve alone, this service will enable him to ask for help with the problem from an expert in the given problem area.
- *Make Collective Decision*. Called by other services in order to mediate the interactions between mobile workers when a collective decision is necessary.

3.4.3 Knowledge Management

Anticipating a mobile worker's knowledge requirements by accessing and customizing knowledge (based on the mobile worker's skill, location, current job, and type of display) and providing access to collective knowledge assets in the team (e.g., by putting novices in touch with experts, as and when required). The following Knowledge Management services have been identified:

- *Find Relevant Information*. Called by other services in order to proactively provide mobile workers with information relevant to the performance of their work.
- *Update Knowledge Base*. Enable a mobile worker in the field to add knowledge to the knowledge base. The types of knowledge identified so far include feedback from the customer, work reports, technical experience, and information about the customer.
- *Find Expert*. Given a problem, use the knowledge base to identify a colleague who is likely to be able to help in the given problem domain.

In addition, we have identified a set of common generic services, among which we state essentially the *User Authorization* service component for password based authentication of a user.

3.5 Implementation through JADE-LEAP

This section concentrates on the implementation of the generic services with JADE-LEAP agents. JADE (Java Agent DEvelopment Framework) is a software framework for developing and running multiagent systems, which is fully implemented in the Java language. It was developed by TILAB and the University of Parma. JADE-LEAP is its extension developed by the LEAP project for the deployment of agents on lightweight devices. JADE-LEAP is a distributed platform; therefore, agents can be deployed on several different running entities called containers. Though generic services are designed to be independent from the agents they are plugged into, their tight coupling with the agent makes it so that generic services that are built for a particular FIPA agent platform will not work on another one (until standardisation applies at this level as well).

In the JADE-LEAP approach, all the actions of an agent are controlled by instances of the Behavior class [3]. Hence, generic services that are plugged into an agent must add the proper behaviors to this agent to be able to modify its activities.

3.5.1 Concurrent Sessions

In a service framework, the Respondent component must be able to handle several requests concurrently to allow scalability of applications in terms of number of users, and ensure an acceptable mean response time. Due to the behavior-scheduling scheme of JADE-LEAP, this has to be performed by adding ahead of time a preset number of Respondent behaviors. This number, described by a `Policy` object, defines the maximum number of concurrent sessions. This allows us to finely control the load of a service and to limit the potential overhead on the agent. Note that, for a given agent, requests will never really be processed simultaneously since the JADE-LEAP behavior scheduler within an agent is monothreaded at the Java level.

3.5.2 The "Future" Idiom

Interactions between service components rely on agent message exchanges, which are intrinsically **asynchronous**. In order to ease and leverage the use of the generic services within applications and to allow various kinds of initiation, it appeared useful to turn the asynchronous nature of the interactions into **synchronous** method calls (also called an API-based approach). This is done through a mechanism called the **Future idiom** [7]. Using this mechanism, the method called immediately returns something that is not the real result but a placeholder (i.e., the future variable) that will eventually contain the result that will be produced after an asynchronous computation. The caller can issue the call and do something else until it needs the real result, which it then extracts from the future variable (this is called "redeeming the future"). If the asynchronous computation ended before the redeeming time, fine; otherwise, the caller will block until the result is actually ready.

4 Application 1: Roadside Assistance

4.1 ADAC Business Case

The ADAC *(Allgemeiner Deutscher Automobil-Club)*, based in Munich, Germany, is the largest automobile club in Europe with nearly 14 million members. It was founded back in 1903. More than 6,000 employees offer a wide range of products for the members of the club. The most famous service is the Roadside Assistance, which is provided by 1,700 so-called "Yellow Angels," a fleet of yellow-colored service vehicles. In 2001, the "Yellow Angels" and their colleagues from subcontractors registered 3.5 million roadside assistance activities. In order to provide this service, the ADAC runs five call centers all over Germany.

The idea for agent technology, in general, and the LEAP project, in particular, to create an added value for ADAC, was to provide the Yellow Angels with mobile devices and develop an agent-based application that relieves them from some of their routine tasks and supports their decision-making processes.

From the many activities the Yellow Angels are involved in throughout the day, three were chosen to be supported by the agent application: planning a route to a customer, retrieving tourist information for the customer, and scheduling a meeting/agreeing on a venue (e.g., for lunch) with other Yellow Angels.

With this selection, we have covered three typical areas of activity of the mobile workforce: organization of the work itself, support for the customer, and social activities related to work. On the other hand, one main activity, namely, the assignment of jobs to the Yellow Angels, is *not* touched by our application—the corresponding processes between the call centers and the Yellow Angels are highly sensitive and crucial to the functioning of the Roadside Assistance service, hence we were not allowed to interfere with them.

4.2 ADAC Application Details

We now explain, in detail, the requirements and the resulting features of the application for the three main features it provided.

4.2.1 Route Planning

Here, the scenario is as follows: The Yellow Angel receives a job request from a call center and has to find the shortest route from his current whereabouts to the location of the stranded motorist. As there is no direct interface to the data transferred from the call center, the start and destination addresses must be entered by hand. However, the mobile device is equipped with a GPS receiver by means of which the current location can be discovered and entered into the route request automatically. The route is then computed by the central ADAC route planning system and is displayed in text form on the mobile device. Of course, a more sophisticated presentation of the route, for instance, by a map that is updated while driving, is imaginable. Yet, it could not be realized due to limited resources both in terms of performance of the mobile device and in terms of manpower for the development.

The "Route planning" feature is an application of the "*PlanRoute*" and "*GetCurrentPosition*" generic services.

4.2.2 Tourist Information

This feature mainly serves to support the customer, but also the Yellow Angel himself. Suppose that, due to the delay caused by the breakdown, a motorist is looking for a place to stay for the night; or that the Yellow Angel wants to refill his tank on the way to his next job and, therefore, needs to know the filling stations along the route. These and other kinds of tourist information are provided by an ADAC database that the application can access, either along a given route or around a given location, for instance, where the breakdown occurs. The "tourist information" feature is an application of the "*FindRelevantInformation*" generic service, partly combined with the "*PlanRoute*" generic service.

4.2.3 Meeting Scheduling

Suppose a Yellow Angel who is on the road wants to meet with several others for lunch or in order to have a group meeting. Then, he issues an invitation that is propagated to all invitees. The invitees (including the Initiator) are notified by their mobile devices and can reply to the invitation if they want. Then, they have the opportunity to enter their

preferences, e.g., the maximum time they want to spend traveling to the meeting, the current location (alternatively detected via GPS, again), and the lists of "buddies" of whose intentions they want to be kept informed. Each time some preferences are changed, the system estimates the distances of the invitees' locations to the possible venues, which are given by a predefined set. On this basis, the system makes a suggestion for the venue by choosing the one that is most convenient to all invitees in terms of the maximum driving time. Every invitee gets informed about the suggestion, but can freely indicate an intention to go to a different place. The system doesn't make a definite decision about the venue to be chosen, but invitees can change their intentions until the invitation expires. Finally, their intentions will converge to one place to go to—or they may split up into subgroups and go to different places, if this seems more convenient to them.

The "meeting scheduling" feature is a direct application of the "*MakeCollectiveDecision*" generic service, which, in turn, makes use of the "*EstimateRouteCost*" and "*GetCurrentPosition*" generic services.

4.3 Field Trial Set-Up, Execution, and Evaluation

In a one week long field trial in April 2002, the system was tested under real-life conditions. The trial took place near ADAC's headquarters, in the area around Munich. For cost reasons, only three Yellow Angels took part, each one accompanied by a member of the LEAP project as a "supervisor." Interviews about their experiences during this week were conducted afterwards with all participants.

4.3.1 System Architecture and Implementation

Hardware and Software. The mobile devices used in the field trial were the Siemens SX45 and Compaq iPAQ Pocket PC's, running Windows CE 3.0 and Windows Pocket PC 2002, respectively. Communication was established over GPRS, using the T-D1 network of German telecom provider T-Mobile. Here, the Siemens SX45 offered the most elegant solution as it features a built-in GPRS phone, whereas the Compaq iPAQ accessed the mobile network via a Bluetooth connection to an Ericsson T39m mobile phone with GPRS support. Each mobile device was equipped with a Pretec WorldNavigator Teletype GPS receiver.

The mobile devices hosted the agents representing the user of the device (see below), whereas the service-providing agents were hosted by a PC running Linux at the Motorola Research Centre near Paris, France. These agents contacted a special ADAC gateway server, which had, in turn, access to a server farm with ADAC's route planning system and ADAC's tourist information database running under Microsoft SQL Server 2000. All communication except to the mobile devices was done over the public Internet, to which the GPRS network provided access, and over the ADAC Intranet.

We used Jeode EVM 1.9.1 as the Java Virtual Machine on the mobile devices and Java 2 Standard Edition on the PC.

Containers and Agents. We used the JADE-LEAP container concept in a very straightforward way: There was one peripheral container on each mobile device, whereas the main container resided on a PC at the Motorola Research Centre. The multiagent system was composed in the following way.

There were two agents living in the main container, namely:

- the User Admin Agent responsible for accepting a user login to the system;
- the Travel Resource Manager Agent acting as a responder to route planning, tourist information, and driving time estimation requests.

Moreover, there were five agents living in each peripheral container on the mobile devices:

- the Coordinator Agent, coordinating the activities of the other agents in the container and "proxying" the communication to outside agents;
- the User Manager Agent, providing the interface to the GUI;
- the Position Finder Agent, a wrapper agent for the GPS system;
- the Meeting Initiator Agent, sending invitations for new meetings on the user's behalf and generating suggestions about the venue;
- the Meeting Respondent Agent, guiding its user through the process of subscribing to an invitation and agreeing on a suitable venue together with other invitees and their agents.

4.3.2 Field Trial Results

The evaluation of the field trial was based on feedback from the users and logging information generated during usage. Feedback was obtained for each of the three features: Route planning, Tourist information, and Meeting scheduling. An overview of user opinions follows: The analysis of the log files showed that all services were used successfully, were reliable, and functioned as expected during the trial. Tourist information was not used stand-alone very often, but was included in every usage of Route planning. Meeting scheduling was used only a few times because of the very long response time over GPRS. Besides this, bad GPS reception and bad readability of the displays in the bright sunlight were some other issues. These results and the difficulties in handling such compact devices were the most critical general points for the Yellow Angels.

Route Planning/Tourist Information. This feature was used most during the field trial. It worked without problems and received positive comments. It was thought to be more useful to those Yellow Angels who were new to the job or spent time in areas that they were unfamiliar with. The additional information accessible from the mobile device was useful in situations were the car could not be repaired and the customer needed additional services like finding a hotel, a garage, or a filling station. Although these scenarios were not tested with real customers, the application was used and received very well by the Yellow Angels.

Meeting Scheduling. The meeting scheduling feature was not used very often during the field trial because of problems with GPS and the long response time over GPRS. Furthermore, the Yellow Angels rarely meet for lunch and there was no sign that they wanted to change that. Currently, if they want to contact each other, they use their

mobile phones. But, they can imagine using other devices and applications such as those used in LEAP in the future.

5 Application 2: "Surveying for Network Service Provision" Process

5.1 BT Business Process

5.1.1 Description

The generic service component architecture has been applied to the "surveying for network service provision" process of BT, which is executed by mobile workers, called Survey Officers.

The function of the Survey Officers is to provide relevant technical information when queries relating to provision of fixed telecom services for both residential and business customers arise. Typically, surveys are initiated as a result of a request for service provision from a customer where there was inadequate data about the site, plant, or capacity, or where there had been an unsuccessful attempt to install equipment to provide service because of plant or site difficulties. Survey Officers are organized according to geographical areas delineated by telephone exchanges. Each Survey Officer works within a certain area known as his "patch" which comprises a number of exchanges. Patches vary in size and are grouped in areas. The Survey Officers self manage their work through a queuing system. Each officer has ownership of a queue for the patch in which they work. Jobs are allocated to the queue by a system called "Job Management" which is responsible for decomposing service provision into a number of coordinated tasks for the relevant parts of the process to ensure that final completion is achieved. New survey requests are regularly allocated to the queues as customer orders demand. The number of jobs within a queue at any one time is dependent upon the geographical area and the size of the Survey Officer's patch.

5.1.2 Issues

Three main requirements to support the process have been identified. First, the main information resource is the telecommunications network maps, which show cables routes and the location of joints and access points. Currently, all the related maps are stored on a server located on the company's Intranet. Each Survey Officer copies the maps for their patch on to their laptop computer and takes them to their work location. However, the maps are sometimes not up-to-date and the laptop is difficult to carry on to a Customer's premises as it is heavy. Getting the maps online and displayed on a lightweight device like a PDA was expected to improve work performance dramatically. Second, support for dynamic job trading within a team of Survey Officers is needed. One or more urgent jobs which need to be finished that day might appear in a Survey Officer's queue at any point in the working day. In order to ensure timely completion of urgent jobs, the Survey Officer has to negotiate with other team members to balance the day's workload. Currently, this is done by contacting team members via their mobile phones, which takes time and is expensive. The use of autonomous agents which negotiate with each other on behalf of their users was expected to greatly reduce these coordination costs. Last, as Survey Officers frequently travel to new job locations, the plan route functionality was expected to reduce the travel costs.

5.2 Customization of Generic Service Components

Three generic service components have been customized to satisfy the requirements identified in Section 5.1.2. The "*FindRelevantInformation*" service component was customized to provide a "Retrieve telecommunications network maps" service. The "KnowledgeHunter" interface provided by the Respondent component of "*FindRelevantInformation*" was implemented to find and retrieve a telecommunications network map from a server residing on the company's Intranet, using the job's postcode or GPS location as input. Second, the "*PlanRoute*" service component has been customized for the process by implementing the "GetRoute" interface to contact a route planning system to get the shortest route, taking into account current traffic conditions.

Third, the basic Contract Net-based "*GiveJob*" service component was customized as Extended Contract Net, in order to provide a dynamic job reallocation service. The basic Contract Net protocol is composed of two phases: bids collection and awarding. In the bids collection phase, the Initiator sends a call for bids (CFB) to multiple Respondents. Then, the Respondents send their bids to the Initiator. In the awarding phase, the Initiator evaluates the received bids and selects the winner based on the evaluation criteria. The winner gets an award notification and the other Respondents get reject notifications. Extended Contract Net uses the same logic for the bids collection phase. However, the awarding phase is changed to negotiate with each Respondent for an award until anyone receives the award. It means that the Respondents can refuse the award according to their situation. The "*GiveJob*" process is performed as follows: First, the initiator of the service sends a CFB, which contains the job details including the postcode and address of subscribed Respondents. Then, the Respondents get their current locations and calculate the estimated distances from their locations to the job location. Then, they create bids based on the distance and return their bids to the Initiator. The Initiator evaluates the bids and sorts the bids according to the specified policy (such as "shortest distance"). In the awarding phase, the Initiator negotiates with the Respondent that offered the best bid. The Respondent shows the offer to the User and receives the response to return to the Initiator. If the User refuses the offer, then the Initiator contacts the next best bidder. Otherwise, the process is finished.

For the customization of the "*GiveJob*" service component, the Respondent component is provided with the customized implementation of the "Prepare Bid" interface, which gets the current location of the mobile worker using the *GetCurrentPosition* service component. It then executes the Initiator component of the "*EstimateRouteCost*" service component to get the distance from this location to the location of the new job. The estimated distance is used to create a bid for the CFB. Then, the offer is evaluated by the predefined interface "Evaluate Proposal." The implementation of the interface shows the proposal to the User via a GUI component and receives the User's response to the proposal.

5.3 Field Trial Set-Up, Execution, and Evaluation

A field trial of the application was conducted with a team of Survey Officers (SO). Within operational constraints, each Survey Officer's patch was chosen to make full use of the "*GiveJob*" service (i.e., as many adjacent patches as possible). The field trial covered an area of about 80 miles in diameter and six patches.

5.3.1 System Architecture and Implementation

Hardware Set-up. Each Survey Officer was equipped with a palm-sized Personal Digital Assistant (PDA) with network connectivity enabled by a PCMCIA GPRS (General Packet Radio Service) card using an expansion sleeve for the PDA or via Bluetooth to a GPRS enabled mobile phone. A GPS (Global Positioning System) PCMCIA card provided positioning data.

In addition to this, the Survey Officers had access to their existing technology, which aids them in the day-to-day management of their jobs, a laptop with PSTN network connectivity and a mobile phone to contact coworkers and supervisors.

Software Set-up. Information servers behind the company firewall were accessed by client side agents using a secure, IPSec-based Virtual Private Network (VPN), and a Java Virtual Machine (JVM) adhering to the 1.1.x Java Specification ran the application.

Each PDA (an iPAQ Pocket PC) ran a lightweight LEAP container with the Personal Agent responsible for the "*GiveJob*," underground network map, "*PlanRoute*" and "*EstimateRouteCost*" Initiator service components, and a "*GiveJob*" Respondent service component. The main container resided within the company Intranet, running the Agent Management System, and Directory Facilitator agents. Two other containers held agents on the server side: One container held the "*PlanRoute*," "*UserAuthorization*," and "*FindRelevantInformation*" Respondent service components. Another container on a separate machine hosted two agents: JobAgent and NotificationAgent. The role of JobAgent is to extract jobs assigned to a Survey Officer and pass the jobs to the worker via interaction with a PersonalAgent on his device. The NotificationAgent notifies a worker if any urgent jobs are created for him. For this, all the PersonalAgents subscribe to the agent to receive the notification service. Fig. 2 shows this configuration. The agents were located on different machines to remove performance bottlenecks—e.g., stopping an agent locking up a container when undertaking heavyweight database access.

5.3.2 Field Trial Results

All the services were used successfully within the trial. Feedback was obtained from the users, for which an overview of each service follows.

PlanRoute. This was perhaps the most simple of all the services. The service ran without problems and received positive comments regarding the functionality of the service. The service was thought to be more useful to those Survey Officers who spent time in areas that they were unfamiliar with.

Telecommunications Network Maps. Again, the intelligence of this service was implicit in the application rather than the agent-based interaction, that is, being able to download job specific maps asynchronously at a press of a button. The service worked well in the field and under real GPRS conditions. This service reduced the search time for a map, as the *GetCurrentPosition* generic service was customized to integrate a GPS system, allowing automatic identification of the Survey Officer's current location, which was used by the respondent agent to get the map corresponding to the User's current location. A minor inconvenience was the smallness of the PDA screen that was not user-friendly for map viewing.

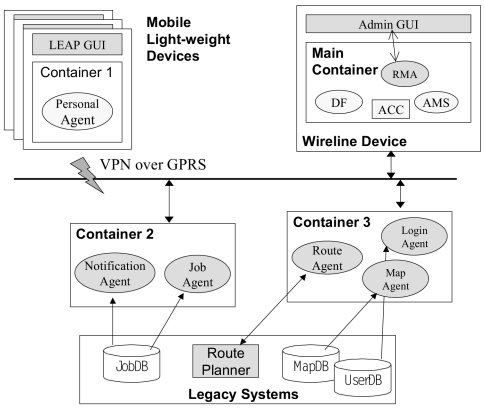


Fig. 2. Application architecture to support "survey process for network service provision."

GiveJob. Technically speaking, the major accomplishment of the field trial was that a successful agent based negotiation was performed within a real-world environment and "off-the-shelf" hardware (GPRS, GPS, Bluetooth, Java on small devices, etc.). The service released the Survey Officers from time-consuming job coordination activities with the help of the autonomous agents. However, there are still issues with the third party technology that we used—for example, GPRS coverage and signal strength was found to be inconsistent across the trial patches. Some of the GPS receivers were not 100 percent reliable. Such problems with ancillary technologies had an impact on the Survey Officers' perceptions of how robust the agent services were.

A user quote for a solid benefit for the business process:

> *The ability of being able to send a job and for LEAP to find the next nearest person, rather than have to ring around, especially when I am running two queues* [patches] *as I am at the moment.*

However, there are "people" issues too:

> *... it is all too easy to refuse the job at the press of a button. When you have someone you know on a phone line, it is human nature to be more likely to say yes, but if it is a machine that will not say anything back if you press no, then I suspect that is what will happen in most cases.*

For the success of real-world implementations of agents on mobile devices, we need to see the robustness of mobile networking improve and, over time this will, especially with the advent of 3G. Also, researchers and developers should be very aware of the "soft" issues related to new technology and business process improvement.

6 Related Work

In summary, the LEAP project has resulted in three important realizations.

The first one is the JADE-LEAP agent platform, respecting the FIPA standard, which is lightweight—meaning that it can be executed on various devices, varying from mobile phones, and PDAs to classical desktops. JADE-LEAP is also generic for the design and development of intelligent software agent applications. In addition, it supports various communication mechanisms (GPRS, TCP/IP, etc.) in a transparent fashion to the user. Some agent platforms are proposed in the literature, like ZEUS [13], FIPA-OS [12], etc., but few of them can run on small devices and support multiple communication protocols as well. MicroFIPA-OS [18] is an extension of the FIPA-OS agent platform developed in parallel with JADE-LEAP during the CRUMPET [17] project to fit small devices.

The second achievement was the set of designed and developed generic services for mobile workers assistance and the two applications developed to support BT Survey Officers and ADAC Yellow Angels. These applications were built quite easily thanks to the generic services architecture, and we were even able to change (redesign and implement) the BT application in about a week. Some agent applications exist in the literature, like the ADEPT project [8] for business process management, but, unfortunately, it is restricted to the development of applications for a subset of business processes (that exclude those used in the LEAP field trials). Another proposal based on the InteRRaP [11] agent architecture to build a multiagent system for virtual enterprises management appeared in [10], but it remains theoretic and incomplete to develop an entire multiagent application for real-world problems. The WHAM [14] system is a prototype supporting mobile workforce and applications in workflow environments, which is implemented using a client-server architecture. But, since this prototype is not based on multiagent technology, which allows many advantages (see [15] for a complete description), it should be less flexible and generic to be extended to other types of application. In [16], there is an architecture proposal to support nomadic agent-based applications, based on the MicroFIPA-OS platform, like CRUMPET for location-based services for tourism.

The third achievement is the two field trials carried out successfully in Germany and England, with real mobile workers being able to use the applications without the need for in-depth technical knowledge of agents. At this time, few real-world tests and evaluations of agent applications have been performed. We can state the ADEPT project realization as an example, which implemented and tested (in a real environment) a business process management case study [9].

7 Conclusion

The paper described specific results of the project LEAP. In detail, these are:

- a generic service composition framework,
- the integration of the framework into the multiagent system LEAP,
- two applications in the area of mobile workforces, and
- the results of the corresponding field trials.

With the generic service architecture, we have addressed the need for reusable software components, which exists in the design of multiagent systems just like in that of any complex software system. Our experience was that it is sometimes hard to separate the generic aspects of a service from the application-specific ones and that it requires some overhead to implement the services independently from the agents in which they will live later on. Yet, it is an approach worth considering as it yields a more modular structure of the system and saves much time of rewriting services for similar applications.

The field trials showed that agent technology is ready for use in commercial applications. In particular, restricted functionality can even be delivered on mobile phones and Pocket PCs. However, several severe technical problems mainly occur because of external influences and lacking product quality. The technical problems outside the LEAP consortium, for example, result from: GPRS coverage problems, VPN problems, bad GPS signal reception in city areas, the usage of emerging technologies and hardware, or usage of Pocket PCs in outdoor areas with short battery life and display readability problems.

Most of the agent services and applications developed by the LEAP consortium have shown their benefit during the field trials. The usability, stability, and performance of the system have to be improved for commercial use. However, these are points for product development and were to some extent outside the focus of LEAP.

It has been shown that the applied approach was the right one and the obtained results are promising for further research and, in particular, for commercialization. Both field trial enterprises have recognized the potential for improvement and will use the results of the field trials to optimize their processes and increase the satisfaction of the workers and customers. Siemens AG and ADAC were discussing further developments in the direction of providing assistance applications on small devices for ADAC customers.

Altogether, one can say that the field trial participants were more or less in favor of the new services and usage possibilities and would freely embrace such a new technology.

Acknowledgements

The authors acknowledge the contribution made by all members of the LEAP project consortium, without which the work described here would not have been possible. They are also grateful to the European Commission, who partially-funded this work.

References

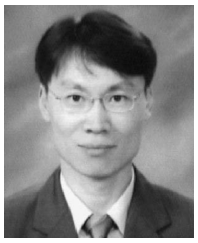
[1] M. Berger, B. Bauer, and M. Watzke, "Towards an Agent-Based Infrastructure for Distributed Virtual Organisations," *Proc. IEEE 10th Int'l Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '01),* 2001.

[2] M. Berger et al., "Porting Distributed Agent-Middleware to Small Mobile Devices," *Proc. First Int'l Joint Conf. Autonomous Agents and Multi-Agent Systems (AAMAS '02), Workshop 16 (Ubiquitous Agents on Embedded, Wearable, and Mobile Devices),* 2002.
[3] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE—A FIPA-Compliant Agent Framework," *Proc. Fourth Int'l Conf. Practical Applications of Intelligent Agents and Multi-Agent Systems (PAAM '99),* pp. 97-108, 1999.
[4] The Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org, 2003.
[5] Lightweight Extensible Agent Platform (LEAP), IST-1999-10211, http://leap.crm-paris.com, 2003.
[6] Java Agent Development Framework (JADE), http://sharon.cselt.it/projects/jade, 2003.
[7] M.L. Scott, *Programming Language Pragmatics.* Morgan Kaufmann, pp. 20-21, 1999.
[8] N.R. Jennings et al., "Autonomous Agents for Business Process Management," *Applied Artificial Intelligence,* vol. 14, no. 2, pp. 145-189, Feb. 2000.
[9] N.R. Jennings et al., "Implementing a Business Process Management System Using ADEPT: A Real-World Case Study," *Applied Artificial Intelligence,* vol. 14, no. 5, pp. 421-463, June 2000.
[10] K. Fischer et al., "Intelligent Agents in Virtual Enterprises," *Proc. First Int'l Conf. Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM '96),* 1996.
[11] J.P. Müller, "The Design of Intelligent Agents: A Layered Approach," *Lecture Notes in Artificial Intelligence,* vol. 1177, Berlin: Springer, 1997.
[12] FIPA-OS Agent Platform, Emorphia, http://fipa-os.sourceforge.net, 2003.
[13] ZEUS Agent Platform, BT Exact, http://www.btexact.com/projects/agents/zeus, 2003.
[14] J. Jing et al., "WHAM: Supporting Mobile Workforce and Applications in Workflow Environments," *Proc. 10th Int'l Workshop Research Issues in Data Eng.,* pp. 31-38, 2000.
[15] *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence,* G. Weiss, ed. Cambridge, Mass.: MIT Press, 1999.
[16] M. Laukkanen, H. Helin, and H. Laamanen, "Supporting Nomadic Agent-Based Applications in the FIPA Agent Architecture," *Proc. First Int'l Joint Conf. Autonomous Agents and Multi-Agent Systems (AAMAS '02),* 2002.
[17] S. Poslad et al., "CRUMPET: Creation of User-Friendly Mobile Services Personalised for Tourism," *Proc. Second Int'l Conf. 3G Mobile Comm. Technologies (3G 2001),* 2001.
[18] S. Tarkoma and M. Laukkanen, "Supporting Software Agents on Small Devices," *Proc. First Int'l Joint Conf. Autonomous Agents and Multi-Agent Systems (AAMAS-02),* 2002.

Michael Berger received the diploma in electrical engineering and the PhD degree in computer science from Dresden University of Technology. He has been involved in computer science research for the last eight years, specializing in Computer Supported Cooperative Work (CSCW), distributed systems, and multiagent systems research. Since 1997, Mr. Berger has been a member of the Siemens Intelligent Autonomous Systems Research Group in Munich and involved in technical and team management functions. He was also the technical manager as well as work-package leader of the European research project LEAP. Currently, he is leading projects with Siemens's mobile communications and automotive divisions focusing on middleware and applications for mobile devices and future mobile networks. He is also actively involved in the agent standardization initiative FIPA (Foundation for Intelligent Physical Agents) as a member of FIPA's board of directors and chair of FIPA's Technical Committee "Ad-hoc."

Makram Bouzid received the PhD degree in computer science in multiagent systems modelling and simulation in 2001. He performed his PhD work in the Loria/Inria-Lorraine laboratory. He is a researcher within the Networking and Applications Lab of Motorola in Paris. He joined Motorola in August 2001 and he is interested in intelligent agents domain, including multiagent planning, cooperation, and coordination. He participated in the LEAP and Agentcities EU projects.

Mark Buckland received the BEng degree in software engineering from UMIST and the MSc degree in intelligent systems from Sussex University. He joined BT Exact in 2000 and is a senior research engineer with the Future Technologies Group. He led BT's technical contribution in the EU-funded LEAP project and is the author of two patents in the areas of mobile devices and agent systems. His main research interest is in situated agent (animat) perception and control architectures that pay more attention to agent/environment interaction.



Habin Lee received the BSc degree in management from Hankuk Aviation University, the MEng in management science and PhD degrees in management information systems from KAIST (Korea Advanced Institute of Science and Technology). He joined BT in 2001, and is a senior research engineer with the Intelligent Business Systems Research Group in BT Exact. He had previously worked as an assistant professor at Paichai University and for the foundation of Korean Software Component Consortium. He was a team leader of the project that developed the GIGA EXCELLENCE AWARD winning K-WFMS (knowledge-based workflow management System) at KAIST. His main research interests include the application of multiagent technology to eBusiness, design of multiagent architectures, knowledge management in business processes, and simulation of consumer markets in the presence of network externality.

Nicolas Lhuillier received the engineering diploma from the Ecole des Mines, Nancy, France. He joined Motorola Labs in 2000 and he was the technical coordinator of the LEAP project from June 2001 until its end.

Dieter Olpp received the diploma in mathematics from the Technical University of Brunswick, Germany, in 1993. He also holds a certificate of advanced study in mathematics from the University of Cambridge, UK. In 1996, he received the PhD degree in Brunswick again, specializing in discrete mathematics. In the same year, he joined the corporate technology department of Siemens in Munich, first working with the Discrete Optimization Group. In 2001, he switched from mathematics to computer science and now works with the Software Agents Group as a senior research scientist. Here, his research includes agent applications for mobile devices and agent-based adaptive office assistence.

Jérôme Picault received the software engineering diploma from the Institut National des Télécommunications, Evry, France. He joined Motorola Labs in September 2001. He is interested in distributed environments, especially in multiagent systems. He was a contributor to the European projects LEAP and Agentcities.

John Shepherdson received the honors degree in electronics and communications engineering from the University of North London in 1987 and joined BT the same year. He went on to receive a master's degree in artificial intelligence from Kingston University in 1993. He is a technical manager in BT Exact's Intelligent Systems Laboratory at Adastral Park, Suffolk, UK. He leads the Intelligent Business Systems Research Group, and is instrumental in generating value from multiagent and semantic integration technologies. He is a recognized expert in the field of advanced wireless applications and services (he chaired the Foundation for Intelligent Physical Agent's Gateway technical committee, which developed standards for interworking between software agents in wireline and wireless networks), a named inventor, and author of many papers. He is a member of the Institution of Electrical Engineers.