Realizing Teamwork in the Field: An Agent-Based Approach

An intelligent agent enhances the provision of cooperative services by adopting a component-based approach, increasing accessibility to mobile workers in the field.

Habin Lee, Patrik Mihailescu, and John Shepherdson
BT Group Chief Technology Office

The recent managerial trend of empowering workforces requires mobile workers to rely on a high degree of teamwork in a changing environment.[1] However, today's mobile workforce managers can't simply pick up an off-the-shelf solution to realize the teamwork concept for their teams. Current commercial, mobile solutions highlight their main features in terms of data collection, data delivery, and data synchronization between a mobile device and back-end systems.[2] There isn't much mention of effective cooperation among mobile workers, so presumably this isn't regarded as a must-have feature.

This omission might seem odd considering the rapid advances in *computer-supported cooperative work* in mobile computing during the last decade (see the related sidebar). However, most CSCW research focuses on in-office workers rather than on mobile workers, who work in completely different (behavioral and technical) environments.

To address this, researchers at British Telcommunication's Intelligent Systems Research Centre developed mPower,[3] a component-based framework for developing agent-based cooperation support systems for mobile workforces. In mPower, we designed an intelligent agent that supports cooperation between mobile workers by considering the specific characteristics of mobile workplaces.

A personal assistant agent

If we observe an average mobile worker, Bob, for a few days, we see that his nomadic workstyle prevents him from constantly monitoring a device screen for information updates (such as his job schedule). Furthermore, Bob uses his device mainly for peripheral activities, such as changing task status or entering job completion data, while he carries out primary tasks such as fixing wires and inspecting network equipment without using the device. This dramatically decreases Bob's degree of collaboration with his colleagues, unlike workers in office environments who normally execute their tasks using desktop computers. So, information systems intended to support mobile worker collaboration should support hands-free interaction and have the autonomy to minimize user intervention when executing services such as requesting assistance from colleagues.

Apart from that, managing the business rules guiding cooperation among mobile workers is a critical issue. For example, you might want to initially allow task reassignment via P2P negotiation (also known as *mini-trade*) between mobile workers for all types of tasks but later exclude tasks requiring specific skills. This requires efficient version management of deployed software to avoid inconsistent application behavior.

In mPower, developers can implement application-specific *cooperative services* (computerized services that semiautomate interactions among two or more users to achieve a common

1536-1268/07/$25.00 © 2007 IEEE ■ *Published by the IEEE Computer Society*

Related Work on Teamwork Using Mobile Devices

Initially, some projects researching the development of agent-based personal assistants considered only office-based users who interacted with their personal assistants on activities such as filtering emails and personalizing news items.[1–3] In recent years, the focus has not only shifted to mobile computing but also to providing more advanced services. For example, the Crumpet project[4] looked at developing a personal assistant to improve a user's experience while on holiday by providing location-aware tourism services.

Projects that share a similar theme to our work—that is, coordination of services between mobile users—include ActiveCampus[5] and MyCampus.[6] Both projects focus on developing context-aware services for students and staff within an education scenario. One key differentiator between these projects and our work is our emphasis on coordination between team members as opposed to between individuals.

A project that focuses on coordination between team members is Electric Elves.[7] It uses agent technology to automate commonly executed tasks such as organizing meetings. In the Electric Elves model, coordination between team members is offloaded to a separate proxy that's part of a domain-independent, team-based architecture called Teamcore. By using this architecture, personal assistants aren't tied to a particular coordination interaction protocol. Our work requires personal assistants to perform the coordination themselves, and we achieve independence of coordination protocols by using a component-based architecture that enables personal assistants to dynamically install new cooperative services.

REFERENCES

1. P. Maes, "Agents That Reduce Work and Information Overload," *Comm. ACM*, vol. 37, no. 7, 1994, pp. 30–40.
2. C. Thomas and G. Fischer, "Using Agents to Personalize the Web," *Proc. 2nd Int'l Conf. Intelligent User Interfaces*, ACM Press, 1997, pp. 53–60.
3. T. Haynes et al., "An Automated Meeting Scheduling System That Uses User Preferences," *Proc. 1st Int'l Conf. Autonomous Agents*, ACM Press, 1997; http://sigart.acm.org/proceedings/agents97/A166/A166.pdf.
4. S. Poslad et al., "Crumpet: Creation of User-Friendly Mobile Services Personalised for Tourism," *Proc. 3G 2001—2nd Int'l Conf. 3G Mobile Communication Technologies*, IEEE Press, 2001, pp. 282.
5. M. Ratto et al., "The ActiveClass Project: Experiments in Encouraging Classroom Participation," *Computer Support for Collaborative Learning 2003*, Kluwer, 2003, pp. 477–486.
6. N. Sadeh, F. Gandon, and O. Kwon, *Ambient Intelligence: The MyCampus Experience*, tech. report CMU-ISRI-05-123, Carnegie Mellon Univ., School of Computer Science, 2005.
7. H. Chalupsky et al., "Electric Elves: Applying Agent Technology to Support Human Organizations," *Proc. Int'l Conf. Innovative Application of Artificial Intelligence* (IAAI 01), AAAI Press, 2001, pp. 51–58.

goal) as components that an agent can plug and play as needed. We've incorporated multiagent systems in mPower because of the extensive research indicating the relative ease with which you can achieve software autonomy using asynchronous, message-based cooperation mechanisms (such as an Agent Communication Language[4]) and a belief-desire-intention architecture.[5] We adopted the JADE-LEAP (Java Agent Development Framework-Light Extensible Agent Platform) multiagent system platform[6] as mPower's core, owing to its desirable features in usability, device adaptability, and communication efficiency as well as its small deployment footprint.[7] JADE-LEAP provides the agent execution environment for our agents as well as the mechanism for them to communicate with each other.

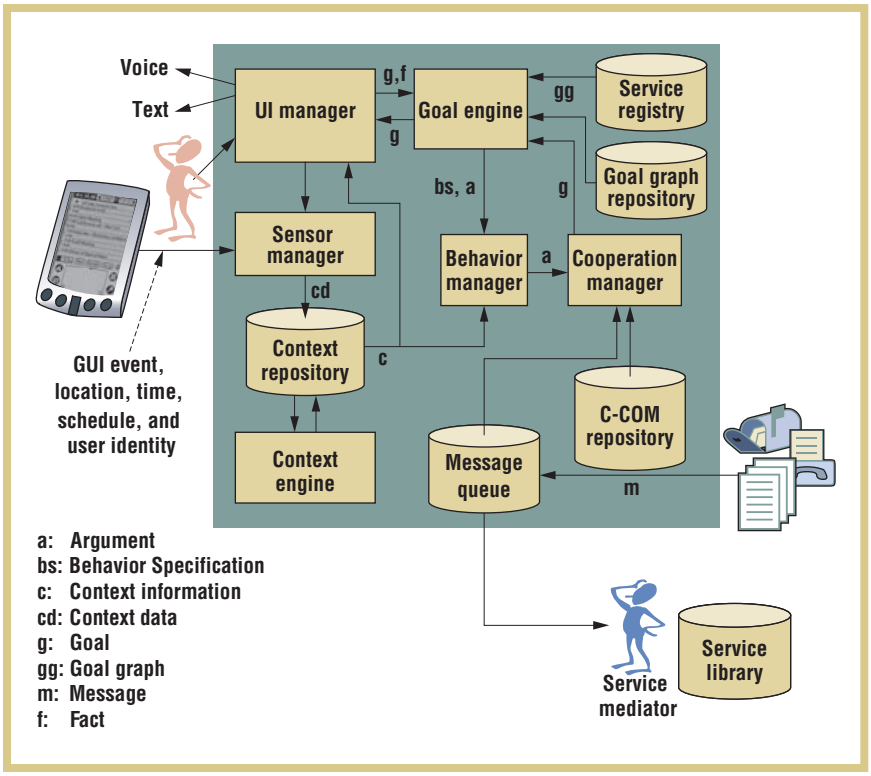
The personal assistant agent, this article's main focus, plays an important role in mPower for the installation, update, and execution of cooperative services. A personal assistant's cooperation support capabilities include

- providing P2P-style services via pluggable software components,
- performing version management of installed components for cooperative services,
- minimizing user intervention in a cooperation process by understanding the working context, and
- configuring the agent-human interface on the basis of the identified work context to increase access to users.

Figure 1 shows a personal assistant's main components and the information flows between them. In a normal scenario, a GUI event (for example, a user requesting a service via a mobile device) is passed to a personal assistant's UI manager. If the UI manager interprets an event as a service request, it passes the event to the goal engine. The goal engine initiates and controls a process that provides the service. The goal graph repository contains a goal-processing graph that defines the service provision process. Executing a goal-processing graph leads to executing agent *behaviors* (agent capabilities or actions) to receive user input, to show output to the user, or to perform message-based interactions with other devices' personal assistants.

The behavior manager maintains a description of all agent behaviors and either executes one or more behaviors or calls the cooperation manager. The cooperation manager installs and updates cooperative services by collaborating with the service mediator agent. The sensor manager collects data from external

Figure 1. The personal assistant architecture.



sources such as GPS, PIMs (personal information management systems), and the system clock and inserts it into the context repository. Upon insertion, the context engine converts the data into meaningful behavior that represents the user's current working context, on the basis of predefined inference rules. The context repository contains the accumulated, processed context information.

During a service-provision process, the goal engine might wish to receive further information from a mobile worker. If so, the UI manager decides which user interface to use to collect the required information on the basis of the context repository's state. The goal and context engines use a modified version of the Zeus Rete engine,[8] which can be used in a lightweight-computing environment to schedule goal-achieving actions and derive high-level context information.

Component-based management

mPower implements each cooperative service as a pluggable component and dispatches it to the mobile device for dynamic installation and execution. The cooperation component (C-COM)[9] uses asynchronous message exchange among multiple agents. It consists of two types of role components—*initiator* (starts an interaction) and *respondent* (reacts to an initiator's request)—each of which plays a specific part on behalf of its user in a cooperation context and encapsulates the cooperation logic. For example, an initiator that enables the FIPA Contract Net-based[10] mini-trade service performs all of a contract manager's required actions. It finds engineers with the right skill set, prepares a call for bids that contains relevant information (for example, job location and required skills), evaluates received bids (such as the distance between the job location and the bidder's location), contracts with the winner, and so on. A personal assistant dynamically installs each role component on a mobile device according to the roles the user played in the business process. Any component in a personal assistant that needs a cooperative service simply treats a C-COM (see the cooperation manager in figure 1) as a black box.

Using C-COMs provides easy version management of deployed services (via dynamic overwriting of obsolete components) and allocation of the right cooperative services to the right users (based on their roles).

In figure 1, the service mediator deploys new cooperative services. It manages a service library that holds all application cooperative service packages. A service package consists of a service description, one or more role components, a goal-processing graph that guides service provision, and behaviors that retrieve user input or present service output. A service description contains versioning information, the roles that are authorized to execute the component, a device profile specifying the minimum computing resources a target device needs to execute each role component, and so on.

The personal assistant periodically contacts the service mediator at predefined times (such as on daily initial log in) to check if any new components or services are available for the user role and device combination. If so, the personal assistant downloads and installs a new package automatically. When a new C-COM is available, the cooperation manager registers it in the C-COM repository, adds a definition of its goal-processing graph to the goal graph repository, and passes the associated behaviors to the behavior manager. In mPower, a Rete engine evaluates a set of production (if-then) rules representing a goal-processing graph. Consequently, the goal graph repository contains sets of rules corresponding to the goal-processing graphs. Finally, the UI manager includes the new service name in the list of available services on the device GUI so that users can access it.

Figure 2 details further how a mobile worker receives a cooperative service when the UI manager passes a service request to the goal engine as a GUI event. The goal engine consists of the inference control, inference engine, rule base, and working memory. Inference control

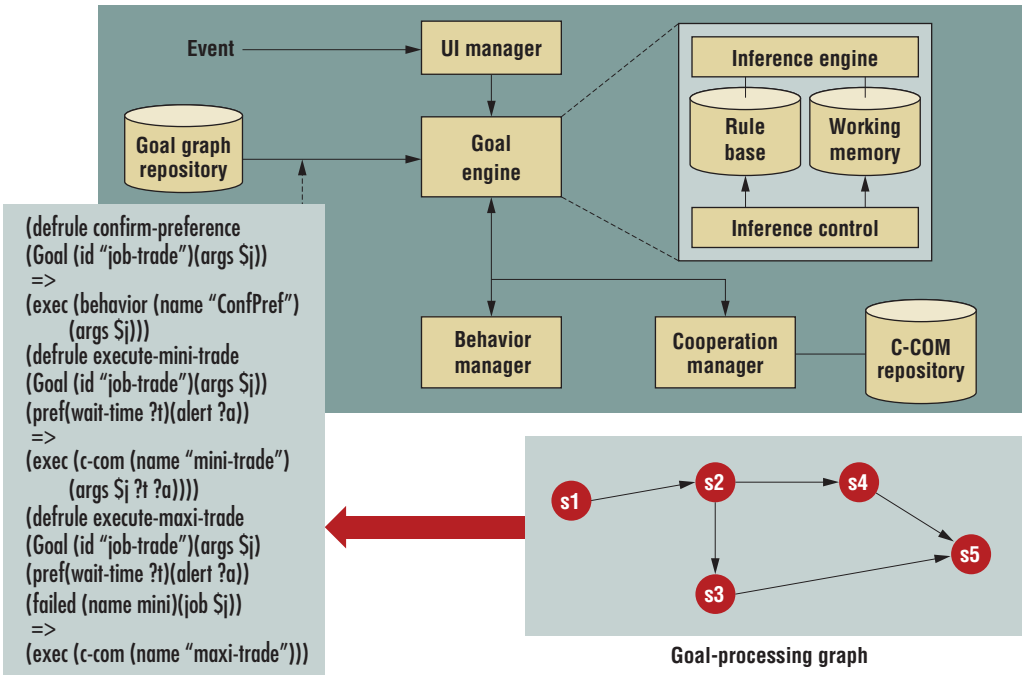


Figure 2. A goal-achieving process for cooperative service provision.

transforms the GUI event into a goal fact by binding goal attribute values from event attributes (such as a service name and a job identifier) the user selected when requesting the service. It also retrieves all relevant rules that define the service's goal-processing graph and inserts them in the inference engine's rule base. The source code in figure 2 is part of the simplified rules for the job-trading service's goal-processing graph (shown in the figure). In the graph, each node represents a processing point wherein a behavior or C-COM is executed, and each arc represents conditions that should be met to reach a succeeding node. Each rule represents a route from one node to another.

For example, the rule **execute-mini-trade** specifies that the goal engine can execute the C-COM **mini-trade** if the working memory contains a goal for **job-trade** and that the working memory contains user preference data as a result of executing the behavior **ConfPref**. The example rules also indicate that the personal assistant can provide a service by executing multiple C-COMs in sequence. That is, the goal engine should execute the **mini-trade** C-COM before the **maxi-trade** (job reassignment via a team leader's mediation) C-COM. Inserting a goal fact in the working memory triggers a chain of rule firing, which will progress the traversal of the goal-processing graph.

Minimizing manual intervention

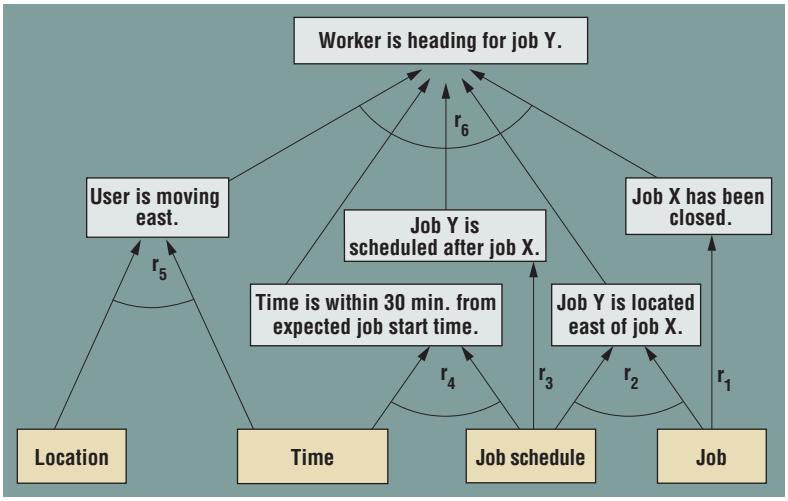
In some cases, a personal assistant can execute cooperative services without user involvement at the decision-making points. For example, an implementation of the job-trading service might require workers' context information (such as skill sets and work schedules) to form the content of the respondent's bids. In this case, the respondent role components can autonomously collect such context information to compose the bids.

The personal assistant's context engine and context repository play key roles in collecting and maintaining context information about each mobile worker's environment. The context engine collects three types of sensor data from a mobile device: *Physical data* provides geographical and time-related information for a user's working area. *Task-oriented data* provides information on duties—what day they're assigned to a worker and their execution schedule. Finally, *behavioral data* includes chronological data on the user's interactions with the application, such as GUI events and services requested, along with the result from executing the requested services.

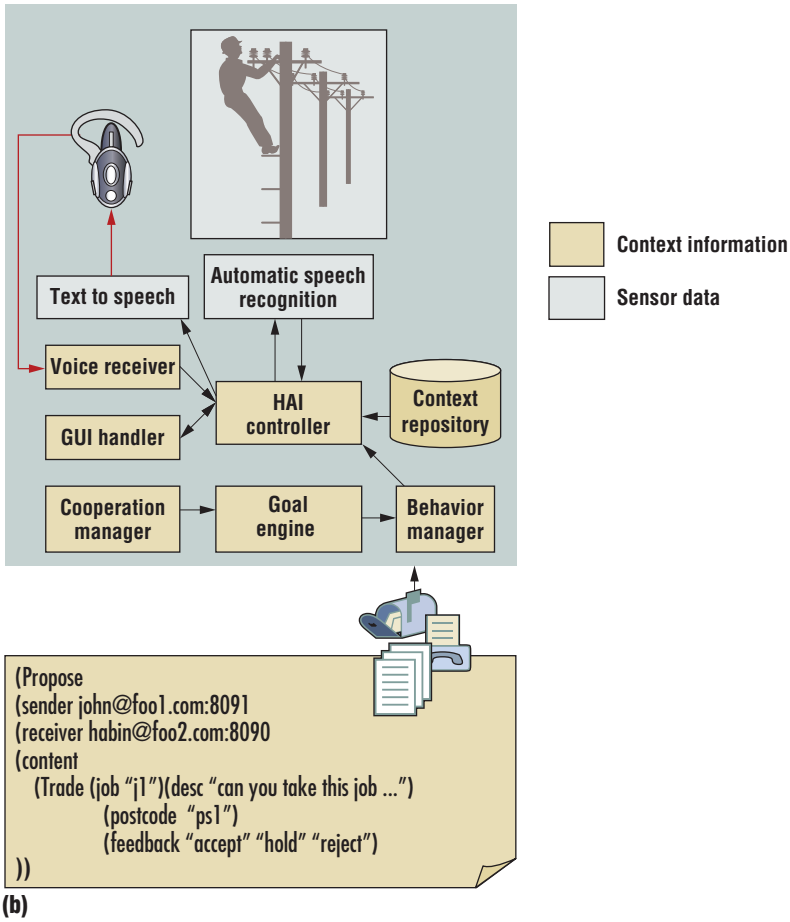
The context engine manipulates the collected sensor data to produce value-added context information that applications can directly use. It achieves this based on a set of production rules that produce new contextual information on the basis of the attribute values of the sensor data and their relationships.

The context engine dynamically loads the rules, which are stored in the local file system. The process for setting up an inference engine and loading facts (input context data) and rules resembles the goal engine's process. We can add the new sensor data schema and corresponding inference rules to the context engine via a pre-

Figure 3. (a) Context-reasoning rules and (b) a voice- or text-based user interface for cooperative services.



(a)



(b)

defined interface of the sensor manager (see figure 1).

Figure 3a shows how we can derive value-added context information from raw context data to implement a customer-relationship-management campaign called Ring Ahead Ring After.

In RARA, average mobile worker Bob must contact his customers before each visit to advise estimated arrival time and after the visit to receive customer feedback. We can configure a personal assistant to remind Bob to ring ahead as he's traveling to the next job location. So, context information such as "the worker is heading toward next job" is crucial for RARA. The simplified rules r1 to r6 in figure 3a show how the personal assistant can infer the context information "worker is heading for job Y" from raw context data such as time, location, job schedule, and job information. Each rectangle represents either deduced context information (grey) or sensor data (gold). An independent line linking two rectangles represents a rule saying that the personal assistant can derive context information from a sensor data item (or other context information). Lines linked by an arc represent a rule saying that the personal assistant can derive context information if and only if all the descendent sensor data or context information exists (the context information "User is moving east." can be derived if and only if both "Location" and "Time" data are available). For instance, figure 4 shows how we provide rule r6 to the personal assistant. In this example, the terms preceded by "?" represent a variable that can match with any relevant facts in the Rete engine's working memory.

Accessible anywhere, anytime

In addition to a text-based interface, mPower's personal assistant has a voice-based interface, which facilitates communication in situations where workers don't have their hands free. Figure 3b shows how a personal assistant can provide Bob with voice-based cooperative services via a mobile device in his pocket. We've implemented voice-based interaction by integrating third-party voice solutions: automatic speech recognition (ASR) and text-to-speech (TTS)

```
(defrule r6
(worker is moving in ?direction)
(jobschedule (prior ?jx)(posterior ?jy))
(job (id ?jx)(status closed))
(located_to (source ?jx)(destination ?jy)(direction ?direction))
(job ?jy should be started within 30)
=>
(assert (worker is heading toward job ?jy))
```

Figure 4. The content of rule r6 in the personal assistant.

Figure 5. mPower deployed in various real-world scenarios.

engines.[11] The *human agent interface* controller mediates interaction between Bob and his personal assistant and manages both the GUI and voice-based interfaces.

Suppose that a colleague sends Bob's PDA a job-trading offer message that requires a real-time response (see figure 5). A respondent role component in the job-trading service handles the message and passes it to the goal engine. The goal engine then sets up a Rete engine session by loading the corresponding rules from the goal graph repository plus the message contents into working memory.

On the basis of the job-trading offer's details, a node of the goal-processing graph needs Bob's input. The behavior manager (on the goal engine's request) invokes the HAI controller, which contacts the context repository to obtain the mobile device's current state and Bob's interaction history. Because Bob hasn't interacted with the device for some time and the response deadline is tight, the HAI controller activates the TTS and GUI components simultaneously. The TTS engine reads the descriptive part of the message content to Bob. After that, the HAI controller activates the voice receiver because Bob will likely give a voice command. If Bob responds to the offer by using a GUI component (for example, by clicking the button of the PDA's pop-up dialog box in figure 5), the GUI delivers the response to the HAI controller via the GUI handler, and the HAI controller deactivates the voice receiver. If Bob responds to the offer using the Bluetooth headset, the HAI controller forwards the voice input to the ASR engine along with context information—in this case, a list of possible responses: accept, hold, and reject. The ASR engine uses the context information to simplify translation and to increase the match's accuracy by reducing the size of the voice recognition dictionary.

mPower in the real world

A team of several BT engineers used a trial application based on mPower technology. They were responsible for maintaining and installing network services in the UK during 2005. The trial aimed to assess the feasibility of moving from centralized job management to team-based job management, in which teams rather than individuals receive job assignments and team members must cooperatively manage the assigned jobs. Before the trial, they used laptop computers to check the jobs assigned to them from a centralized job-scheduling system via a tethered dial-up connection made either at a BT building or a customer's job site.

We selected trial participants from a pool of volunteers. The official trial period was three months; however, the participants used the system unsupported for several months after the official end date because they liked the new way of working.

For the trial, we gave each engineer a PDA running the Microsoft Pocket PC 2003 operating system, which included 55 Mbytes of available RAM. We equipped each PDA with an external GPS receiver and a third-party satellite navigation application. We installed a third-party Java Virtual Machine and used the Standard Widget Toolkit to implement the GUI components. Each mPower-based PDA client included a personal assistant, and we ran job agents responsible for collecting jobs assigned to the team and job-related knowledge on a server on the corporate intranet. Communication between PDA clients and the server took place via GPRS (General Packet Radio Service) secured by a virtual private network connection.

Each time users logged in to the client, their personal assistant connected to the service mediator to check if service com-

ponent updates were available for download. We set up weekly meetings with the trials' manager primarily to discuss overall progress and any effects on the performance of routine business due to introducing new technology. We collected any technical concerns directly from the participants and dealt with them immediately by updating the relevant service components, which were deployed via the service mediator.

We offered three types of cooperative services to each engineer: market-based job assignments, job trading, and urgent job notifications. Market-based job assignments let team members choose which jobs to execute depending on their circumstances. For the trial, we put all jobs in a team job queue, and each engineer could reserve preferred jobs, which then disappeared from general view to prevent duplicate reservations. A job agent managed team jobs, and each personal assistant communicated with the job agent to obtain a list of available jobs. Each engineer could remove any job reservations that couldn't be finished before the customer's deadline. However, certain important *gold jobs* (jobs that strongly impact revenue, customer satisfaction, or both) required engineers to use the job-trading service to meet the deadline of the jobs. The gold job notification service let engineers record their preferences regarding job selection with the job agent, which matched user preferences to attributes of team jobs in the queue and then notified the engineer.

Technical issues

The component-based provision of cooperative services particularly helped with managing deployed services on mobile devices. Trial participants provided additional requirements for using cooperative services during the trial. We quickly integrated the requirements in the C-COMs and dynamically installed them on their mobile devices. Downloading a new version of a C-COM over GPRS typically took less than 10 seconds.

The ASR feedback wasn't as positive as that for TTS, and we saw limited ASR use. Conversely, TTS use was quite high, especially when the engineers were driving or working up poles or in ducts.

The GPRS connection's much-vaunted "always on" property didn't live up to its name in rural areas, and frequent VPN reconnection (in particular, two-phase authentication) irritated users. So, we updated the personal assistant to include connection-management functionality, which autonomously tried to regain a lost GPRS connection and prompted users to enter their VPN login details when it obtained a new GPRS connection. Also, many PDAs, including those used in the trial, automatically go into hibernation mode after a certain amount of time with no user input. The hibernation mode froze the application, which then suspended the personal assistant until the next user input, decreasing access to users. We resolved this through a fix supplied by the PDA manufacturer, but this greatly increased battery consumption.

To accommodate the devices' small screens (320 by 420 pixels), we implanted a highly effective, simplified job closure process. In the simplified version, the personal assistant guided users through this complex data entry task using progressive screen builds based on their most recent input, using XML-to-Java interpretation. However, at the trials' beginning, many participants complained about having to enter job-related notes on the PDA because they didn't feel that the virtual keyboard or phone-style keypad was satisfactory for inputting large amounts of free text. So, we updated the GUI to contain a drop-down list of frequently used phrases, and participants could choose one or more and edit them as needed. This increased the system's user acceptance.

Process findings

The trial had very positive results in terms of business process performance. The market-based job-scheduling mechanism increased productivity (the average number of jobs closed per day by each engineer) by 10 percent. We might attribute this to the reduced delay between a job creation and an engineer's awareness of it, which resulted from using the notification service.

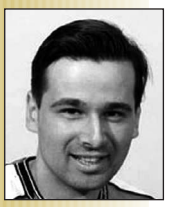
On average, each participant withdrew reservations from 20 percent of their reserved jobs, and about 15 percent of these jobs were then reassigned to other members via job trading. It was hard to measure the impact of using the voice-based user interface and context-awareness functionality on productivity. The participant feedback hinted that such functionality contributed to an increased perception of teamwork due to their colleagues' increased accessibility.

> The strong involvement in decision making resulted in participants having a more proactive attitude toward the execution of assigned jobs.

Finally, we observed a change in the participants' attitude toward the business process. In the trial, the market-based job assignments let engineers make the decisions rather than rely on a centralized workflow system, especially with respect to job selection. This strong involvement in decision making resulted in participants

the AUTHORS

Habin Lee is a senior research fellow in Brunel University's Brunel Business School. He contributed to the mPower project at BT for more than five years before joining Brunel. His research interests include agent technology for business processes, information systems development methodology, and computer-supported cooperative work. He received his PhD in management information systems from the Korea Advanced Institute of Science and Technology. Contact him at Business & Management, Brunel Business School, Brunel Univ., Uxbridge, Middlesex, UB8 3PH, UK; habin.lee@brunel.ac.uk.



Patrik Mihailescu is a software engineer at ShoZu. Before that, he was a key contributor to mPower as a research engineer at the BT Intelligent Systems Research Centre. His research interests include multiagent systems, peer-to-peer technologies, and ubiquitous computing. He received his PhD in computer science from Monash University. Contact him at ShoZu, the Space 57-61 Mortimer St., London W1W 8HS, UK; patrik.mihailescu@googlemail.com.

John Shepherdson is a technical manager in the BT Group Chief Technology Office's Intelligent Systems Research Centre and leads the Intelligent Business Systems Research Group. His research interests include multiagent systems technologies and wireless applications and services. He received his MS in artificial intelligence from the University of Kingston. He's a chartered engineer, a member of the Institution of Engineering and Technology, and a certified leadership coach. Contact him at PP12/MLB1, B62, Adastral Park, Martlesham Heath, Suffolk, IP5 3RE, UK; john.shepherdson@bt.com.

having a more proactive attitude toward the execution of assigned jobs.

We believe features that enhance the accessibility and autonomy of mobile workforces should be present in any system designed to support their decision making. mPower's successful trial has resulted in a deal between BT and a third party to commercialize mPower technology. P

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their comments, which were crucial in enhancing this article.

REFERENCES

1. D. Bowen and E. Lawler, "Empowering Service Employees," *MIT Sloan Management Rev.*, vol. 36, no. 4, 1995, pp. 73–84.
2. "Enabling the Mobile Workforce: The Issues and Trends," white paper, TechRepublic, Feb. 2006; http://whitepapers.techrepublic.com.com/whitepaper.aspx?docid=165945.
3. J. Shepherdson, H. Lee, and P. Mihailescu, "mPower: A Reusable Framework for Building Multi-Agent Systems to Automate Business Processes," *BT Technology J.*, vol. 21, no. 4, 2003, pp. 92–103.
4. Y. Labrou, T. Finin, and Y. Peng, "Agent Communication Languages: The Current Landscape," *IEEE Intelligent Systems*, vol. 14, no. 2, 1999, pp. 45–52.
5. M. Georgeff and F. Ingrand, "Decision-Making in an Embedded Reasoning System," *Proc. 11th Int'l Joint Conf. Artificial Intelligence* (IJCAI 89), Morgan Kaufmann, 1989, pp. 972–978.
6. F. Bellifemine et al., "JADE—A White Article," *Telecom Italia Lab J. EXP*, vol. 3, no. 3, 2003, pp. 6–19.
7. P. Mihailescu et al., "MAS Platforms as an Enabler of Enterprise Mobilization: The State of the Art," *Proc. IEEE Int'l Conf. Systems, Man & Cybernetics* (SMC 04), vol. 2, IEEE Press, 2004, pp. 1889–1894.
8. J. Collis et al., "The Zeus Agent Building Tool-Kit," *BT Technology J.*, vol. 16, no. 3, 1998, pp. 60–68.
9. H. Lee, P. Mihailescu, and J. Shepherdson, "Conversational Component-Based Open Multi-Agent Architecture for Flexible Information Trade," *Proc. 1st European Workshop Multi-Agent Systems* (EUMAS 03), LNAI 2782, Springer, pp. 109–116.
10. *FIPA Contract Net Interaction Protocol Specification*, Foundation for Intelligent Physical Agents, Dec. 2002; www.fipa.org/specs/fipa00029/SC00029H.pdf.
11. J. Page and A. Breen, "The Laureate Text-to-Speech System: Architecture and Applications," *BT Technology J.*, vol. 14, no 1, 1996, pp. 84–99.