TeamWorker: An Agent-Based Support System for Mobile Task Execution

Habin Lee, Patrik Mihailescu and John Shepherdson

Abstract. An organization's mobile workforce is a vital asset, as it is in the front line liaising with customers, and driving the sale of an organization's products and/or services. Despite this, the IT support provided to mobile workers is often inferior to that available to office-based workers. Mobile workers operate in an unreliable environment, and as such require differing types of support. Within this paper we present a multi-agent based computer cooperative support system known as TeamWorker which can help overcome the difficulties faced by mobile workers. Within this system, each mobile worker is assigned a personal agent that can assist her/him during the working day through appropriate service provision (based on current work context), and through monitoring work progress to anticipate and undertake required actions on the user's behalf. A detailed presentation of the TeamWorker system is given, including the benefits provided for a real life mobile business process.

1 Introduction

A mobile workforce differs significantly from an office-based workforce. Mobile workers operate within an uncertain computing environment where both network and computational resources are lower, when compared to an office-based workforce that typically works within a reliable computing environment [2]. Mobile workers also behave differently to office based workers due to their frequent movements, and physical isolation from other workers. As a result mobile workers require additional IT support, as they are typically in the front line interacting with customers and driving the products/services offered by an organization.

Despite this, the support provided for a mobile workforce is often substandard when compared to that given to an office based workforce. Agent technology has been used to support business processes for the last decade. Huhns and Singh [3] summarize the state

of the art in agent-based workflows. Jennings et al. [4] insist that a multi-agent system has the necessary features for the support of modern dynamic business processes and propose a suitable multi-agent system architecture. Khoshafian and Buckiewicz [5] highlight some of the limitations faced by a mobile workforce. Firstly, the intelligence of an agent can be used to transparently monitor the actions of the user, and determine an appropriate course of action based on the current work context. Secondly agents have the ability to support different types of interaction protocols that can be used to coordinate the activities of distributed users. Finally, the autonomy of an agent can be used to increase the usability of a system for a user by optimizing the interaction types based on the constraints of the mobile computing device.

This paper proposes a multi-agent based Computer Supported Cooperative Work (CSCW) system called TeamWorker that supports the coordination of mobile workers. A personal agent plays a central role within the TeamWorker system, in that each mobile worker is assigned their own personal agent which assists them during their working day. Each personal agent is deployed on a mobile computing device which enables a mobile worker to access the system regardless of his or her location. In this paper we will discuss all aspects of the TeamWorker system, including both technical and non-technical details such as how the system improves the working practices of mobile workers.

In the following section we will introduce the TeamWorker system, and follow up in section three with a discussion of the mobile business process which this system supports. In section four we present an architectural overview of the TeamWorker system, and provide technical details such as the structure of the personal agent. In section five we provide a detailed explanation as to how the TeamWorker system supports the specified mobile business process, and then we conclude this paper.

2 Introduction to the TeamWorker System

The TeamWorker system has been designed to support a business process where mobile workers are the primary participants. Mobile workers are grouped together into teams to complete jobs within a certain geographically area. Each team has a leader who oversees the progress of their team, ensuring that jobs are completed, and acting as the first point of contact for team members. Each mobile worker is assigned their own personal agent which will guide them during the execution of their jobs during the working day. A personal agent resides within a mobile computing device and provides a number of services such as job retrieval, job update, and job closure. The TeamWorker system has not been developed from scratch, rather it has been derived from a platform called mPower [6]. The mPower platform is a component-based framework that can be used to support a mobile business process by modeling the interactions amongst participants via a linked set of conversational messages. The mPower platform has been implemented using an existing multi-agent system called JADE-LEAP [1] that has been designed to operate on a variety of mobile computing devices, such as a cell phone or a personal digital assistant (PDA). The mPower platform is composed of four layers:

1. Foundation: This layer provides infrastructure support for the upper three layers. It relies on the functionality provided by the JADE-LEAP platform such as message

composition, message delivery, ontology/language support, and agent management.

2. Component: This layer provides reusable components known as conversational components (C-COMs) which provide a common set of functionality that can be used within any type of application domain. For example, a C-COM that provides the ability to retrieve and add information from/to a knowledge source regardless of the information type. A C-COM is made-up of two parts, an interaction protocol that defines the sequence of asynchronous messages between participating roles, and role components that undertake the necessary actions to fulfill the provided service. Further information on C-COMs can be found within [7].
3. Generic workflow: This layer provides a generic set of pre-defined high-level workflows that can be applied to business processes in similar domains. Each generic workflow aims to complete a higher-level business goal as opposed to an individual C-COM which completes a specific service, and as such a generic workflow is composed of a set of linked C-COMs.
4. Application: This layer is where applications can be composed such as the TeamWorker system. An application can be developed by selecting only the required features (such as ontology support, used to devise domain-specific ontologies) from each of the lower three layers.

Further information on the mPower platform can be found within [6].

3 Mobile Business Process

Within this section we will present an overview of a real life mobile business process within British Telecommunications plc (BT) and discuss how TeamWorker has been applied to support it.

3.1 Telecommunications Service Provision and Maintenance

BT operates a large mobile workforce to maintain its network and provide telecommunications services to residential and business customers across the UK. Every day thousands of engineers perform tens of thousands of tasks. The process begins when either a customer or an engineer requests the provision of a new service, or repair of an existing service. Once a new job is created (either manually by a call centre operator or automatically) it is placed within a job pool. This job is then assigned to a particular engineer by a centralized workflow management system, based on criteria such as the skills required to complete the job, and the location of the job. In some cases a job may require two or more engineers, and may also involve multiple locations such as a telephone exchange and the customer's premises.

Engineers receive their jobs by connecting to the centralized workflow management system using a laptop, at the start of the working day either via a GSM or PSTN connection. Once a job is completed, the engineer closes the job with the appropriate information such as the cause of the fault, equipment used, hours worked, etc. If new jobs arrive during the day they are kept in an engineers queue until they are manually retrieved,

otherwise a phone call is made to inform the engineer that s/he has new work (depending on the urgency of the new job).

As engineers infrequently use their laptops due to network (high access cost, slow network access, etc), and physical factors (size of laptop prevents use during job execution), a number of issues arise. Firstly, engineers often have an out of date view of their assigned work, and do not have visibility as to what other engineers within their area are working on. Therefore if the engineer's schedule has been altered s/he will only be aware of the change when s/he connects to the centralized workflow system, or by a phone call. Secondly, if an engineer knows that they cannot complete an assigned job, his/her only recourse is to initiate several mobile phone calls to other engineers (which increases costs). There is no support within the current system to transparently trade jobs amongst engineers. Finally, the inability of engineers to update their progress in real time limits the visibility management have, and therefore hampers their ability to respond to changes such as informing customers as to the progress of their requests (i.e. when an engineer will arrive), and assigning new urgent jobs to the appropriate engineers.

3.2 Application of the TeamWorker System

The TeamWorker system was designed to improve many of the working practices of BT field engineers. Improvement falls into three areas:

1. Team working/coordination: Currently, in principal engineers are assigned to teams, however due to the limitations of the current system (as discussed in the previous section), they typically work alone. The TeamWorker system addresses this issue by providing a set of coordination services such as job trading, job assistance, and organizing meetings (work or social) that enable engineers to interact with each other in real time.
2. Work visibility: As the TeamWorker system supports the ability to communicate over a GPRS network, engineers can benefit from the always-on aspect. Therefore, engineers not only have the ability to access the latest job information, including any changes to their schedules (which can be automatically pushed out to them), but also be able to update their progress in real time. In addition, management has greater visibility as to what is occurring in the field, and can respond to changes more quickly.
3. Work flexibility: As the TeamWorker system has been developed for use within mobile computing devices, engineers can take the device with them regardless of whether they are at a customer's premise, or up a pole. This not only enables engineers to access the system at any time and any place, but also increases their reachability when other engineers or management need to get in touch with them.

4 Technical Overview of the TeamWorker System

Within this section we firstly provide a high level architectural overview of the TeamWorker system, followed by a brief technical overview in section 4.1. Finally in section 4.2 the internal structure of the personal agent is discussed.

4.1 Architectural Overview of TeamWorker

The TeamWorker system is split into two parts: 1) a front-end part that executes within a mobile computing device (one for each engineer), and 2) a backend part that executes within a server, or set of servers (for all engineers). Both parts communicate securely over a network such as GPRS, via a VPN connection. Each front-end part contains a single agent, while the backend part contains a number of agents that may be distributed on a set of servers (depending upon the deployment options). Each agent provides a specific type of service. When an engineer wishes to access a service, a request is sent from her/his front-end agent to the corresponding backend agent which fulfils the service request by accessing resources within the Corporate Intranet such as a database, or a legacy system. Figure 1 provides the architectural overview of the TeamWorker system.

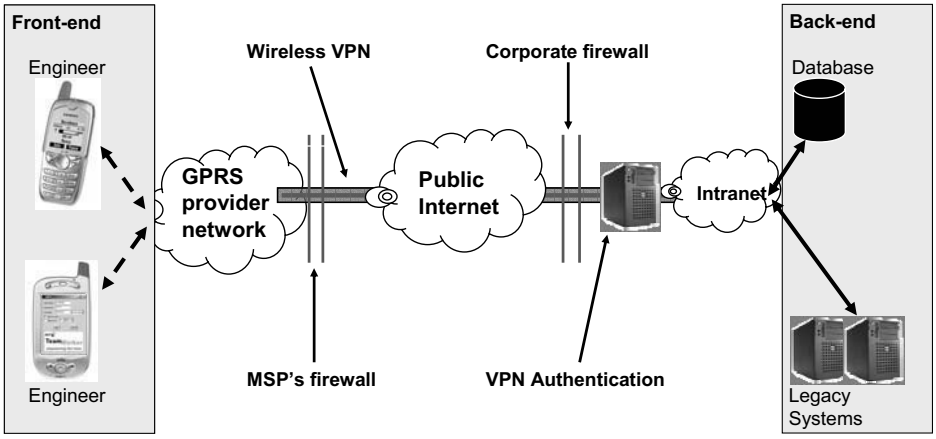

Fig. 1. Architectural overview of the TeamWorker system.

Within the front-end part of TeamWorker, JADE-LEAP runs in split-container mode, which reduces both the level of computing resources required and the time taken to register the agent with the agent management system (AMS) of the main container. Using this runtime configuration mode, a single TCP socket connection is established from the front-end to the back-end, and maintained for the duration of the application. All requests and responses are transmitted over this TCP socket. Both push and pull communication modes are supported. If a network disconnection occurs (i.e. no network coverage), messages are buffered at both-ends, and re-connection occurs automatically. Both the buffering and re-connection timings are configurable. Finally, only a single instance of the front-end is permitted, to not only conserve computing resources, but to also reduce the level of confusion for users as to which instance of the application they were using when they return from using other external applications. The backend part is comprised of approximately six agents, each of which provides a set service. Those services are:

1. User authorization: Before an engineer is able to access any other services s/he must firstly be authenticated. Once authenticated, an engineer is assigned a role that determines which services s/he can access.
2. Job management: This is the most frequently used service, as it enables engineers to retrieve their jobs, and also update job progress.
3. Job trading: This service enables an engineer to trade her/his jobs within the team. There are two types of job trades: 1) mini, and 2) maxi.
4. Notification: As new jobs enter the job pool during the day, this service ensures that they can be notified to the appropriate engineer. There are two types of notifications: 1) new job, and 2) urgent job.
5. Subscription: Once an engineer is authorized s/he is automatically subscribed to a number of services such as notification, based on her/his assigned user role.
6. Team list: This service keeps track of the status of each engineer, with regards to who is currently logged into the system. This presence information can be used in a variety of ways such as when an engineer wishes to request assistance during the execution of a job.

4.2 Technical Overview of TeamWorker

The TeamWorker system has been developed using the Java programming language. The front-end part has been implemented using the Foundation profile [9] of the Java 2 micro edition (J2ME) platform, while the backend part has been implemented using the Java 2 standard edition (J2SE) platform version 1.4.1. Both parts use version 3.2 of the JADE-LEAP platform. The front-end part has been tested on a number of mobile computing devices that operate on the Microsoft Pocket PC 2002, and 2003 OS. This includes consumer devices such as the Compaq IPAQ, the O2 XDA I and II, and the Panasonic P2 (a non-consumer device). The underlying Java virtual machine (VM) used to execute the front-end part is the IBM J9 version 2.1 VM. A number of optimizations as discussed in [8] have been applied to the front-end part of the TeamWorker system in order to improve performance. Table 1 provides a breakdown summary of the total size of both the front-end and backend part of the TeamWorker system, including the size of the mPower and JADE-LEAP runtime libraries.

A modular approach has been employed to both simplify the maintainance and installation of new features such as C-COM services to the front-end part of TeamWorker. For example, each C-COM service is deployed independently within its own individual jar file. Using this approach, if a C-COM service needs to be updated, only the jar file needs to be replaced, and not the entire front-end part of TeamWorker. There is no fixed limit on the number of C-COM services or user interface screens that can be used within TeamWorker, instead it is dependent on the available computing resources.

4.3 Internal Overview of Personal Agent

The core of the TeamWorker system is the personal agent which is responsible for assisting an engineer during her/his working day by not only providing access to useful

services, but also by monitoring the engineer's progress and taking autonomous action. To achieve this the personal agent is comprised of three specific roles:

1. User manager: The user manager is responsible for managing a user's preferences by monitoring her/his interaction with the user interface. Through observing a user's interaction behavior over a period of time, the user manager can build an interaction profile, and is better able to tailor the application's functionality to meet the needs of the user. For example, if the user manager observes that the user seldom views the routing information for a job, it may decide to only download this information on demand and not when the job details are first downloaded. All interaction with the user interface pass through the user manager.
2. Coordination manager: The coordination manager is responsible for fulfilling a service request by selecting a goal plan that meets the requirements of the requested service from a list of available goal plans. Each goal plan contains details of the tasks involved and their execution sequence. Typically a task will execute one or more C-COMs, or access a resource from the Resource Manager or interact with the user during its execution. An exception handler is provided within the coordination manager that will process any failures that occur during the goal-achieving phase of a goal plan.
3. Resource manager: The resource manager is responsible for managing all the resources required to support the execution of the personal agent, and any application-specific components such as user jobs, C-COMs, and third party programs.

Front-end Component name	**Physical size**
TeamWorker • Personal agent • User Interface • C-COM (services)	 84 kb 209 kb 118 kb
mPower	138 kb
JADE-LEAP	309 kb
Total	**858 kb**
Back-end Component name	
TeamWorker	241 kb
MPower	138 kb
JADE-LEAP	1593 kb
Total	**1972 kb**

Table 1. Runtime size of TeamWorker system.

In order to increase the autonomy of a personal agent, a clear separation between itself and the underlying user interface was created. This provides a number of development and runtime benefits, such as: 1) the user interface can be developed independently from the internal workings of the personal agent by a non-agent developer; 2) the personal agent is capable of operating on a diverse set of mobile computing devices; and 3) during its execution the personal agent could move to another mobile computing device if a failure occurs (e.g. a battery runs flat). As both the personal agent and the user interface

run within separate threads, an asynchronous event-based mechanism is used for communication. There are four standard events that can be generated. These are:

1. Goal event: This is generated when a goal plan needs to be created in order to fulfill a service. For example when an engineer wishes to retrieve work assigned to her/him.
2. User interface event: Whenever an action on the user interface needs to be performed this type of event is generated. For example, when the results from a service call are available they are sent back to the user interface to be displayed to the user.
3. Resource event: This is generated whenever a resource needs to be accessed or modified. For example, when the coordination manager needs to execute a C-COM to complete a goal plan.
4. System event: This represents a system level event that typically is generated externally from the personal agent. For example, when the network has been disconnected.

All events are captured by the personal agent, which acts as a sensor that collates, records, and forwards the events to the appropriate destination, as shown in figure 2.

5. Support for Mobile Business Process

Within the remainder of this paper we discuss step by step how the TeamWorker system supports the lifecycle of an engineer's job from its inception to its completion.

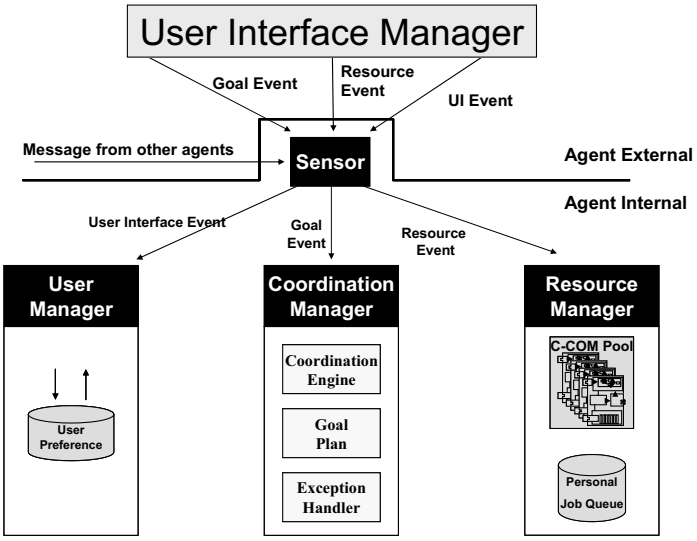

Fig. 2. Internal structure of the TeamWorker system personal agent.

5.1 Job Creation

When a new job is created (usually as the result of a Customer request), it is placed within a job pool where it is scheduled overnight by a centralized workflow management system. At the start of the day, a job agent from the TeamWorker system retrieves these scheduled jobs, including a list of engineers who have been roistered to work for the day. The job details and engineer data are stored within a relational database, ready to be sent to an individual engineer as they log into the system. To compliment the job creation process, a future enhancement of the TeamWorker system will include the ability for engineers to create new jobs, e.g. on noticing damage to a network element. We will now discuss how jobs are delivered to engineers.

5.2 Job Delivery

Jobs are delivered to engineers using two well-known communication modes: 1) Push, and 2) Pull. In the push mode, jobs are dispatched transparently to an engineer from the backend part as they become available, while in the pull mode jobs are dispatched (if available) to the engineer when requested manually. When an engineer logs into the system at the start of the day (as shown on the left hand side of figure 3), a list of any jobs assigned to her/him are delivered once s/he are authenticated (push mode). In addition to this, if the engineer is assigned the team leader role, s/he will also receive the list of jobs assigned to the team. The right hand side of figure 3 shows the view an engineer has available to them to view the jobs assigned to them. This view is identical when viewing both personal jobs and team jobs.

Currently within the TeamWorker system there are two business rules that determine how and when jobs are delivered to engineers. Either jobs are delivered directly to engineers, or via the team leader. When jobs are delivered via the team leader, the team leader has the ability to check the job assignments, and perform job re-allocation at her/his discretion. This is useful if a team member is absent as typically the team leader will be notified and s/he is then able to immediately make the necessary job reallocations to ensure that jobs are completed on time. New business rules for job delivery may be added to the TeamWorker system at any time based on the working practice of the targeted organization.

An engineer is able to refresh her/his job queue manually, by selecting the refresh menu item from the queue menu. There are two types of refresh modes available:

1. Heavy refresh: This mode retrieves all the job details from the backend part regardless of the job list that is currently available locally. This type of refresh is performed automatically when an engineer logs into the system.
2. Light refresh: This mode only retrieves new job information from the backend, by comparing what is available locally and at the backend part.

5.3 Job Execution

A number of services are provided within the TeamWorker system that assist an engineer during the execution of her/his work. Within the following subsections we discuss

four such services beginning with how the TeamWorker system provides engineers with the required knowledge to complete their assigned jobs.

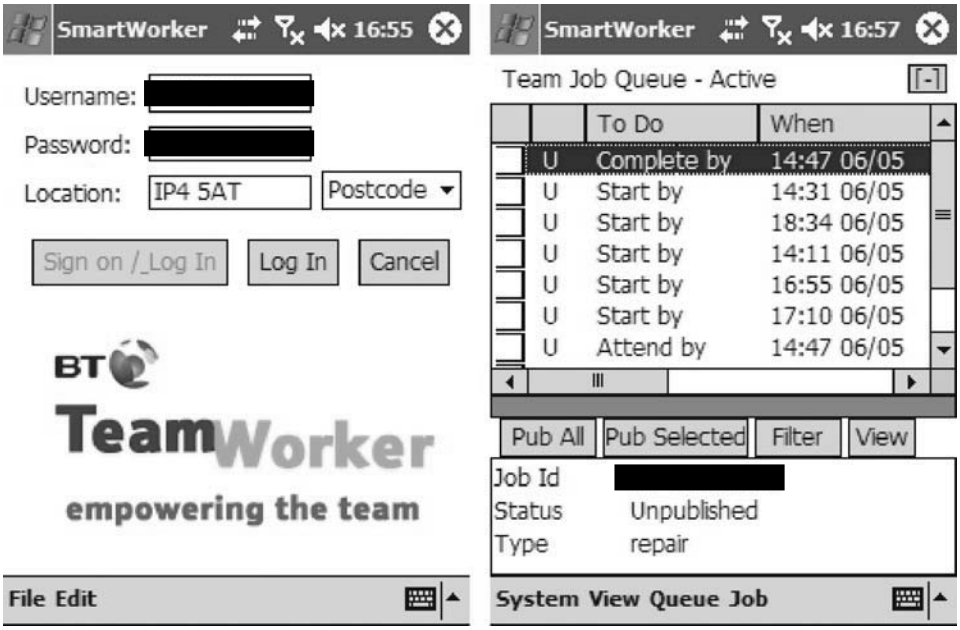

Fig. 3. Screenshots of the user authorization and job queue screen.

5.3.1 Job Details

Each job contains a set of information that provides an engineer with the knowledge that helps them to complete it. This information is broken down into a number of categories, such as customer information, routing information, and job summary. Within the TeamWorker system this information is presented to the engineer using tabbed folders, where each category is presented to the user in an individual tab as shown in figure 4. The lower right arrow buttons enable the engineer to scroll through the available information categories. The majority of the information presented is read only. However there are a few fields available that can be edited such as the status of a job, and when a job has been completed. An engineer is able to view the job details via the job queue screen as shown in the right hand side of figure 3.

In addition to viewing job information, an engineer is also able to enter notes for a job. This information can be specific to a job or to a location. For example, an engineer may add a note that a cable at the job location needs to be replaced soon. This information is useful for other engineers who may in future receive a job in the similar location. A future enhancement of the note taking function is to enable engineers the ability to add a digital photograph or small video clip instead of a plain text note.

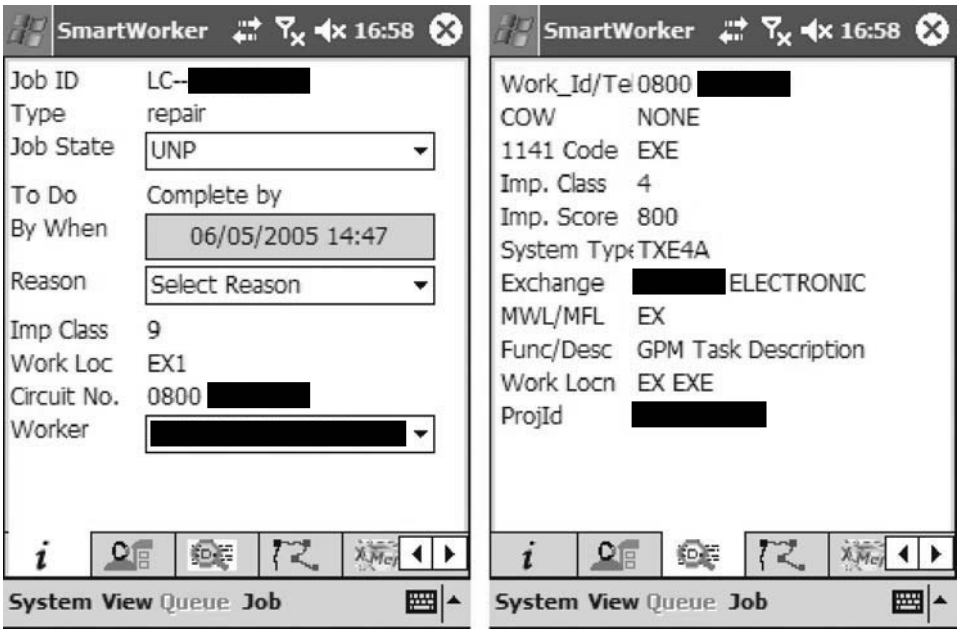

Fig. 4. Screenshot of the job details screen.

5.3.2 Job Trading

Job trading enables engineers to trade jobs amongst themselves. This is very useful when an engineer realizes that s/he cannot complete an assigned job, and rather than missing the appointment time, s/he can instead ask if a colleague is willing to complete the job. Two types of job trading services have been implemented, 1) mini trade, and 2) maxi trade. In a mini job trade an engineer offers a job to other engineers within their team who meet certain criteria. For example, an engineer who has the right skills to complete the job, or an engineer who is within close proximity of the job. New criteria may be added depending upon the working practice of the organization. An engineer can trade a job by selecting it from the job queue screen, and then choosing the mini trade menu item as shown on the left hand side of figure 5. Once an engineer has initiated a mini job trade, s/he can keep track of the progress of this trade by switching to the coordination view screen as shown on the right hand side of figure 5.

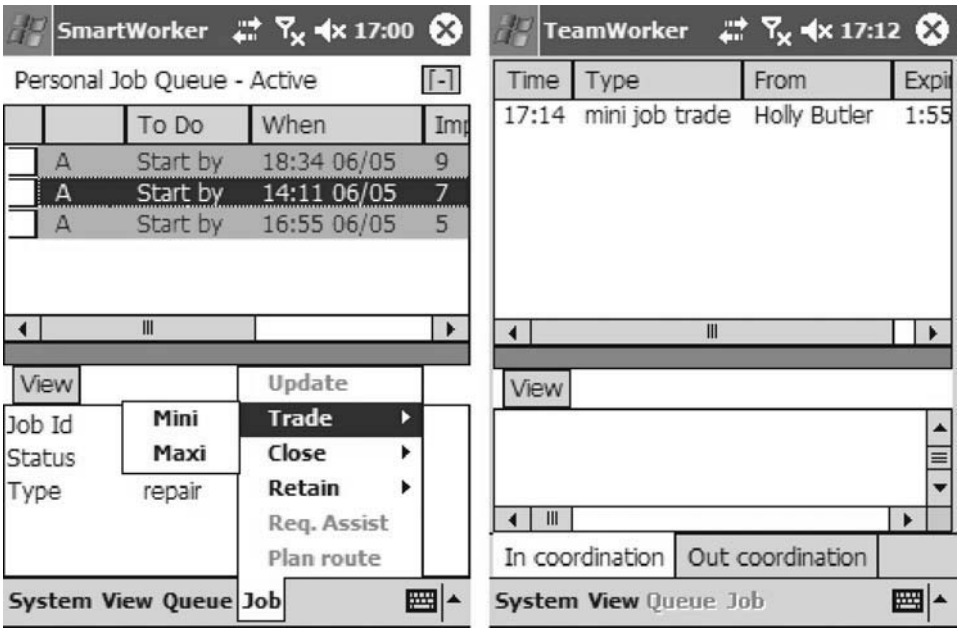

Fig. 5. Screenshot of a mini job trade.

An engineer who receives a mini job trade offer is able to view the offer within the same coordination view screen as shown on the right hand side of figure 5. From this screen an engineer is able to view the job details and decide whether or not s/he is willing to accept the job. To view the jobs, an engineer highlights the job proposal and selects the view button, then is presented with a dialog that shows all the job details as per figure 4, with the exception that an engineer cannot edit any fields. Once an engineer accepts a mini job trade proposal, all the other engineers who received the same one will receive a notification informing them that it has now been withdrawn.

A maxi job trade is an alternative job trading option for an engineer, as instead of sending an offer to several of their peers, a single offer is sent directly to their team leader, who will be asked to re-assign the job on their behalf. This can be useful when an engineer's mini job trade repeatedly fails. When a team leader receives a maxi job trade offer s/he can view the job details in the same way as with a mini job trade. However s/he will have the ability from within this view to reassign the job. As with both job trade types, there is a time limit on the offer. Finally additional business rules may be added to the job trading process that determines how it operates. For example, one such rule could be that jobs cannot be traded more than three times, or an engineer who already has four jobs is not able to accept or receive a job trade.

5.3.3 Request Assistance

When a job requires two or more engineers to complete, engineers are able to request assistance from their colleagues. This coordination service is similar to the job trading service in that engineers who require assistance send a request that contains such information as the job type and location, and when assistance is required to their peers. Once an engineer receives a job assistance request it appears within their coordination view screen (as shown in the right hand side of figure 5), where s/he can view the details and decide if s/he is willing to assist. As with the mini job trade service, once an engineer accepts this request, the job assistance request will be withdrawn from other engineers.

5.3.4 Job State Update

To provide greater visibility as to the progress of jobs within the field, engineers are able to update the progress of their assigned jobs. There are six available job states:

1. Unpublished: This is the initial job state of a job when it enters the TeamWorker system.
2. Published: Once a team leader publishes a job, its job state changes to published which enables an engineer to retrieve the job or for it to be pushed out automatically.
3. Activated: Once a team member receives a job its state is set to activate, to indicate it has been received.
4. Executed: This job state is used to indicate that an engineer is currently executing a job.
5. Negotiated: This job state is used to show that a job is currently under negotiation (that is either a mini or maxi job trade is occurring).
6. Closed: This job state is used to represent that a job has been completed.

An engineer can change the status of a job by firstly viewing the details of the job and then selecting the appropriate job state from a combo box as shown on the left hand side of figure 6. Once the progress of a job has been changed, a team leader is able to view these changes by refreshing the team job queue. A sample screenshot of this view is shown on the right hand side of figure 6.

5.4 Job Closure

Once an engineer has completed a job s/he is able to close the job and capture closure information such as the time taken to complete the job, the cause of the fault, materials used etc. In addition, an engineer is also able to capture customer feedback. Information such as if they were satisfied with the level of service received, consent for an engineer to dig, and the cost of non-contractual work can be captured directly by an engineer once a job is completed.

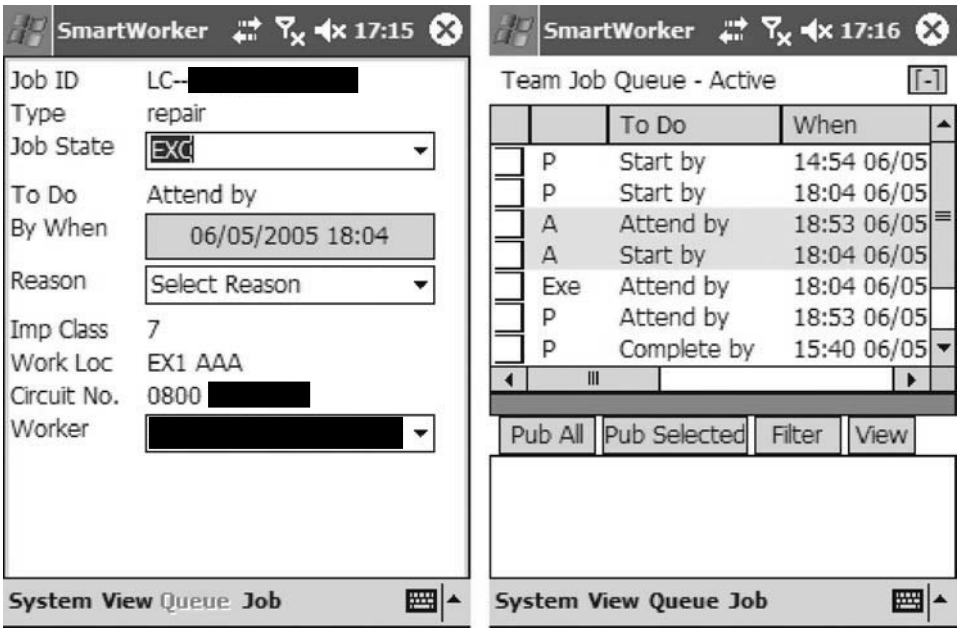

Fig. 6. Screenshot of job state update.

5.5 Job Injection

Within the course of the day new jobs may arrive in the job pool, which need to be delivered to engineers. The TeamWorker system supports two forms of job injection:

1. Normal: In normal job injection mode, jobs have already been assigned to engineers and only need to be delivered. As with the job delivery (discussed in section 5.2), there are two business rules that determine how jobs are delivered to engineers: 1) directly to engineers, or 2) via the team leader. The left hand side of figure 7 shows job injection when the business rule is set to 'via the team leader', and the right hand side of figure 7 shows direct job injection to team members.
2. Urgent: In urgent job injection mode, jobs have not yet been assigned to an engineer rather they have been assigned to a team. In this case a mini job proposal is generated automatically by the system and sent to members of the team that meet certain criteria. If no one accepts this urgent job after the expiry time, its priority will be raised, resulting in a maxi trade request being generated.

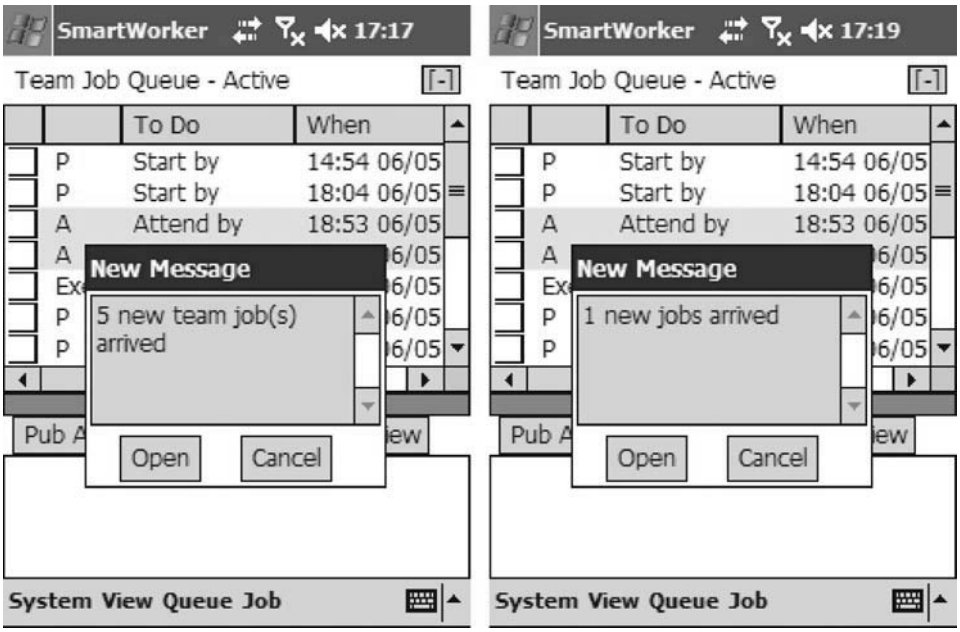

Fig. 7. Job injection screenshot.

Conclusion

This paper provides details about an agent based computer supported cooperative system known as TeamWorker, which supports mobile business processes. Agent technology plays a key part in the TeamWorker system, as each mobile worker is assigned a personal agent, which negates some of the problems experienced within the mobile computing environment (in particular usability, device adaptability, communications efficiency and computing resource requirements). A technical overview of the TeamWorker system was presented, and a detailed description as to how the system supports a real life telecommunications mobile business process was given. An internal trial of the system has been conducted in order to assess the technical qualities and potential business benefits of agent technology in general and the TeamWorker application in particular.

References

1. Berger, M., Rusitschka, S., Schlichte, M., Toropov, D., and Watzke, M. (2003). Porting Agents to Small Mobile Devices - The Development of the Lightweight Extensible Agent Platform. EXP in search of innovation special issue on JADE, vol. 3, no. 3, pp. 32-41.

2. Forman, G. and Zahorjan, J. (1994). The Challenges of Mobile Computing. IEEE Computer, Vol. 27, No. 4, pp. 38-47, April.
3. Huhns, M.N. and Singh, M.P.: Workflow Agents. IEEE Internet Computing, 2 (4), (1998) 94-96.
4. Jennings, N. R., Norman, T. J., Faratin, P., O'Brien P. and Odgers, B.: Autonomous agents for business process management. Int. Journal of Applied AI 14(2), (2000) 145-189.
5. Khoshafian, S. and Buckiewicz, M.: 'Introduction to Groupware, Workflow, and Workgroup Computing', John Wiley & Sons, Inc (1995).
6. Lee, H., Mihailescu, P. and Shepherdson, J.W.: mPower: A Component-based Framework for the Development of Multi-Agent Systems, BT Technology Journal, 23 (3), Oct. 2003..
7. Lee, H., Mihailescu, P. and Shepherdson, J.W. (2003). Conversational Component-based Open Multi-agent Architecture for Flexible Information Trade. Lecture Notes in Artificial Intelligence (LNAI) 2782, 109-116.
8. Mihailescu, P., Lee, H. and Shepherdson, J.W. (2004). Hold the sources: A Gander at J2ME Optimisation techniques, Proceedings of the 1st International Workshop on Ubiquitous Computing, pp 73 – 82, April.
9. Sun. (2002). JSR 46 J2ME Foundation Profile. http://jcp.org/en/jsr/detail?id=46.

Author Affiliations

Intelligent Systems Research and Innovation Centre
British Telecommunications plc
B62 MLB1/pp12, Adastral Park
Martlesham Heath, Ipswich
Suffolk, IP5 3RE, UK
Ha.lee@bt.com
patrik.2.mihailescu@bt.com
john.shepherdson@bt.com

Information about Software

Software is available on the Internet as

( ) prototype version
( ) full fledged software (freeware), version no.:
( ) full fledged software (for money), version no.:
( ) demo/trial version
(X) not (yet) available

Contact person for question about the software:

Name: John Shepherdson
Email: john.shepherdson@bt.com