A multi-agent system to support location-based group decision making in mobile teams

H Lee, M A Buckland and J W Shepherdson

This paper describes an agent-based approach for developing a location-based asynchronous group decision-support system for mobile teams. The approach maximises the use of reusable service components (GSCmas — generic service component for multi-agent systems) as the main interaction mechanism between agents to allow flexible support of a new group-decision process. The paper describes the architecture of a GSCmas and provides details of how the GSCmas is integrated within a decision support system. Finally a system (mPower) based on the proposed approach is introduced and applied to a location-based group decision problem.

1. Introduction

One of the main issues for the development of a group decision-support system (GDSS) for mobile teams is the mobility of the team members which makes it difficult to identify their locations and co-ordinate their input to collective decisions. Therefore, a key challenge in developing a GDSS for mobile teams is to continuously track the location of each team member and to perform appropriate actions where necessary to maintain the consistency of the decision-making process. Furthermore, mobile workers lack the information required for prompt and accurate decision making when they are not able to access the various information sources within their corporate intranet. Even when each team member is equipped with a mobile device and wireless connection, searching for the necessary information is inconvenient because of connection instability, lower processing capability, inconvenient user interfaces (e.g. virtual keyboard), etc.

In this paper, an agent-based approach to developing location-based asynchronous group decision support systems is proposed, mPower (a system based on that approach) is introduced, and the way in which the above challenges can be resolved is discussed.

The mPower application adopts a multi-agent system (MAS) approach. The MAS is considered a key technology to support distributed teams. Intelligent, autonomous agents can collaborate to provide opportunities to reduce communications costs, combat information overload, and improve response times [1, 2]. The mPower system contains agents that interact to support distributed group decision-making processes. Each user of the system is supported by an agent that acts as their personal assistant, tracking their current position and using that information to automate part of the group decision-making process. The personal agent behaves independently of the user, according to a predefined decision-support policy, negotiating with other agents to exchange information as and when necessary. The autonomous characteristics of multi-agent systems are essential to facilitate asynchronous GDSS, as occasionally users cannot contribute in a timely manner to all the decision-making processes in which they are involved.

GDSSs must offer flexible support for group decision-making processes as each process has different information needs. The proposed approach to providing such flexibility is to use predefined service building blocks (referred to as generic service components) which use FIPA [3] standard interaction protocols to facilitate inter-component communication. The generic service component for multi-agent systems (GSCmas) is an aggregation of role-based [4] software components, with each GSCmas supporting a different group decision-making process. Therefore when a user needs to participate in a novel group decision-making process, the user's personal agent can support that process via the appropriate GSCmas.

This paper is structured as follows. The next section reviews related literature. The third section details the architectural features of GSCmas that particularly lend themselves to use in asynchronous group decision-support systems. In the fourth section, there is a description of the use of mPower by a mobile team in a telecommunications company in the UK. The paper concludes with a discussion and summary.

2. Literature review

Multi-agent systems are used as a core technology in various applications, from information retrieval [1] to business process automation [2]. Many of the multi-agent system platforms are based on Java and can be run on heavyweight devices using Java 2 standard edition (J2SE) [5, 6]. However, the lightweight extensible agent platform (LEAP) [7] is an exception, as it enables the key components of a multi-agent system to run on a wide range of devices, from PDAs and Smart Phones using either Java 2 Micro Edition (J2ME) or Personal Java, to desktops and servers running J2SE [8]. This enables the benefits of agent technology (e.g. autonomous decision making based upon contextual information) to be applied to mobile applications including location-based services.

LEAP, used as the basis for mPower, is a multi-agent platform that both complies with recognised standards for agent systems, i.e. FIPA, and can run on lightweight devices. In addition, LEAP provides useful functionality for mobile communications, such as the mediator concept and the JICP protocol [9].

Group decision-support systems fall into the category of CSCW (computer supported co-operative work). CSCW '... is about groups of users — how to design systems to support their work as a group and how to understand the effect of technology on their work patterns' [10]. GDSS can be classified as asynchronous or synchronous, according to the way in which collaboration takes place. A synchronous GDSS typically provides an electronic meeting place where participants can collaborate in real time, complemented by some computerised facilities that provide relevant information to participants during the course of the decision-making process. An asynchronous GDSS generally provides an electronic discussion space where participants can read or write items relating to a particular issue at different times, and therefore do not necessarily participate concurrently. A summary of the various types of GDSS can be found in Khoshafian and Buckiewicz [10].

The mPower system is an asynchronous GDSS in that the mobile workers participate in a decision-making process at different times. However, it does not provide a shared place where the participants can read or write items. The views of the participants are passed to each other via messages exchanged by autonomous agents.

Current GDSSs are mostly based on wired environments where the participants are guaranteed to have a stable connection and access to various information sources with reasonable response times. However, this assumption does not hold for mobile workers, who frequently have difficulties getting information from multiple information sources because of wireless network instability. This paper shows that many of the challenges faced by designers of asynchronous group decision-support systems for mobile working environments can be addressed by the combination of multi-agent systems and GSCmas.

3. mPower — an asynchronous group decision-support system for mobile teams

3.1 *Overall architecture*

The main components of a decision-support system (DSS) are data, dialogue, and model (the DDM metaphor) [11]. The model component represents the basic decision model used in the system. The data component is used during the execution of the decision model, and the dialogue component is used to get input from, or show model execution results to, the user. The mPower system uses the DDM metaphor within a multi-agent framework.

A GDSS is different from a DSS in that a GDSS supports multiple users having multiple roles, whereas a DSS supports a single user with a single role. Within a GDSS the different roles have associated activities that are linked to form decision processes.

Figure 1 shows the internal architecture of an mPower personal agent that supports group decision-making processes. A personal agent consists of four main components:

- dialogue management module,
- decision model executor,
- data management module,
- GSCmas library.

The dialogue management module is used to provide a means to get information from, or show results to the user. The decision model executor uses the appropriate template to perform a given decision process — a decision process template specifies all the activities associated with the role that a personal agent is playing in a particular group decision-making process. During the execution of some of these activities, the decision model executor may need external input. This is obtained from the user via the dialogue management module or from other information agents. In the latter case, the data management module first uses the GSCmas library to find the generic service component type that can supply the required information, then looks up the address of an instance, and finally contacts the instance to request the information. All

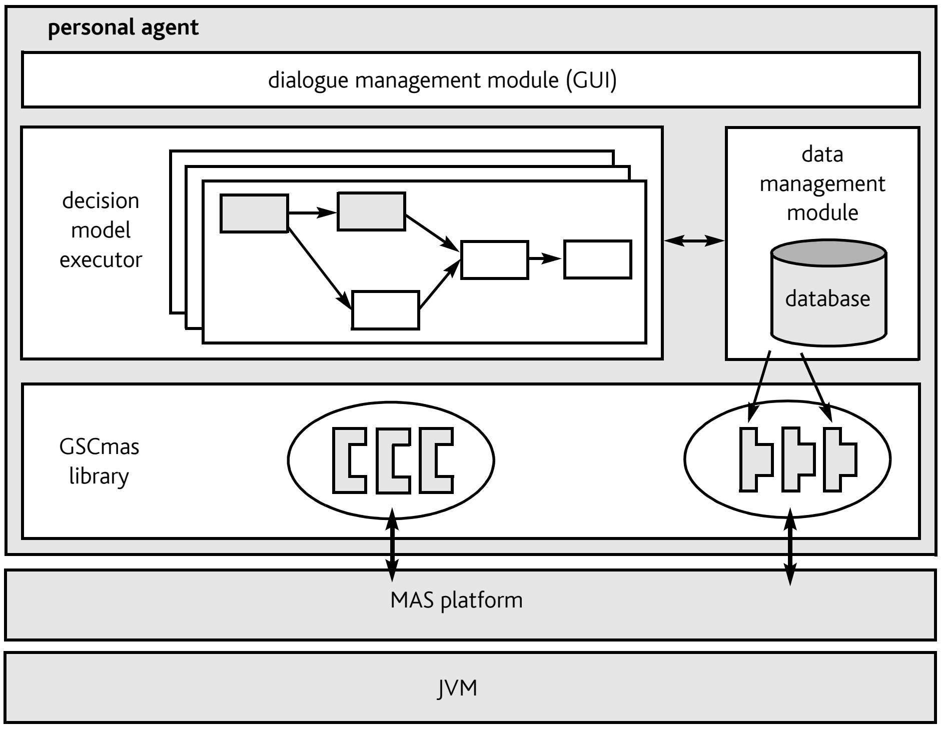


Fig 1 Internal architecture of an mPower personal agent.

information obtained from external sources is stored in a database within the data management module. The fourth component, the GSCmas library, is responsible for the registration and de-registration of all GSCmas modules associated with the platform.

3.2 *Group decision-making process definition*

A group decision-making process is modelled via extended Petri nets [12], using the constructs shown in Fig 2.

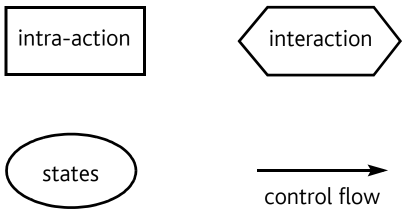


Fig 2 Modelling constructs for group decision-making processes.

A decision process definition is composed of nodes and links. Nodes represent activities and states of a decision process, and links represent control flows between two nodes. An activity is classified as an intra-action or an interaction. An intra-action represents an activity that can be performed by a personal agent using its own resources (e.g. evaluating a proposal based upon a policy). An interaction is an activity which requires communication with other agents, for example sending a message to, or receiving a message from, an information agent. An interaction may be specified with a GSCmas. States represent the current state of a decision process, and are used to determine the next activity that should be performed after an activity has been completed.

3.3 *GSCmas — a component-based information retrieval process for multi-agent systems*

A GSCmas is a set of role-based components which interact with each other via messages to achieve a common goal (i.e. support a given group decision-making process). It is an implementation of an interaction pattern [13] for a multi-agent system. While an interaction protocol defined by FIPA specifies the order of valid messages for an interaction, the interaction pattern in this paper specialises the interaction protocol by specifying actions that should be performed by participating agents in an interaction. Each sub-component of the GSCmas is responsible for the behaviour of a given role in the interaction and is distributed across one or more agents according to the application requirements. This means that an agent can participate in any predefined interaction by installing and using the necessary sub-component. The GSCmas can be reused in similar applications where the same interaction protocol and the same message content language are used, with minimal change to the existing software components. A GSCmas is non-divisible in that a sub-component cannot achieve the designated goal without interaction with other sub-components.

Figure 3 shows the overall structure of a GSCmas. It consists of an instantiation of each of the two generic role

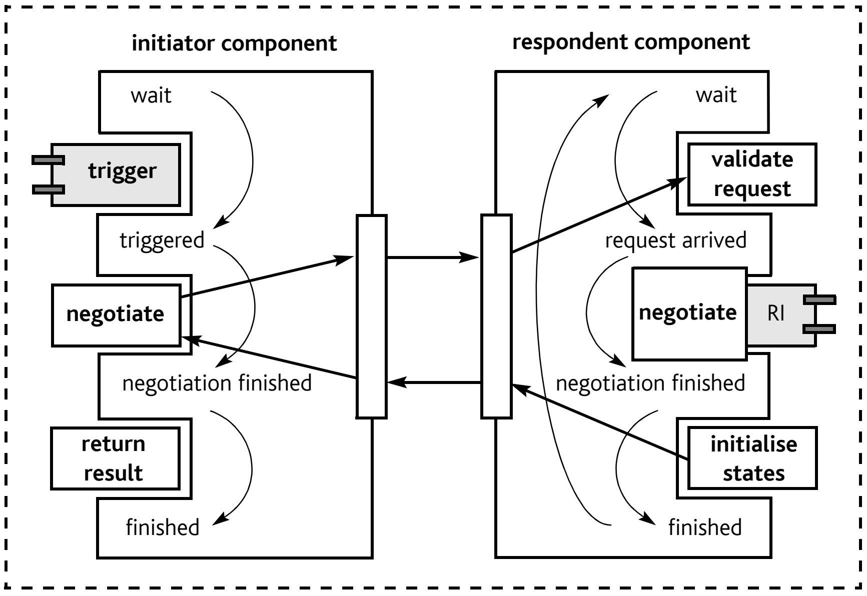


Fig 3 Structure of a GSCmas.

components — initiator and respondent. The initiator component is responsible for starting an interaction with other agents which have the relevant respondent component installed, and returning the service result (if any) to the requesting agent. The respondent component is responsible for handling a request message from the initiator component, and interacts with its host agent via a predefined interface to generate service results which it eventually returns to the initiator.

The initiator component provides a pseudo method-call interface (shown as 'trigger' in Fig 3). The requesting agent calls the trigger method, and the initiator component blocks (or freezes) the caller until the service result is prepared and returned. This is to align the synchronous method call of the requesting agent with the internal asynchronous message passing between the initiator and respondent components. The initiator component hides all the interaction details (such as preparing request messages, interpreting response messages, and the interaction protocol used for the collaboration with other agents, etc) from the caller. Customisation of a respondent component for a particular service is performed by providing a different implementation of its interface specification — thus different sources of the same information can be made available to initiator components.

Aggregation produces more complex service components which use service results from two or more GSCmas in a certain order. Figure 4 shows a GSCmas which uses the service results from two generic service components (GSCmas$_2$ and GSCmas$_3$). The links in this GSCmas 'chain' interact with each other via the trigger methods provided by each initiator component.

4. Illustrative example

4.1 *Target application*

The mPower application has undergone trials with a team of mobile workers in a telecommunications company in the UK. The target process was a 'survey process for telecommunications service provision' where the main actors are survey officers.

The function of a survey officer is to survey a customer's site to gather the information needed for the provision of telecommunications equipment. 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. Survey officers work within geographical areas delineated by groups of telephone exchanges, known as a 'patch', which vary in physical size. Survey officers each have ownership of a queue of work for the patch for which they are responsible. Jobs are allocated to the survey officer's queue by a system that is responsible for breaking down a service provision into a number of co-ordinated tasks. New surveys 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 service provision demand at the time. Factors that influence this include marketing of new communications services and patch population.

One of the main decision problems for survey officers is dynamic job allocation to team members. Survey officers choose a number of jobs in the morning from their job queue, depending on the importance and geographical adjacency of the jobs. Problems can occur when urgent jobs arrive during the working day. Normally, any job (urgent or otherwise) would be allocated to the survey

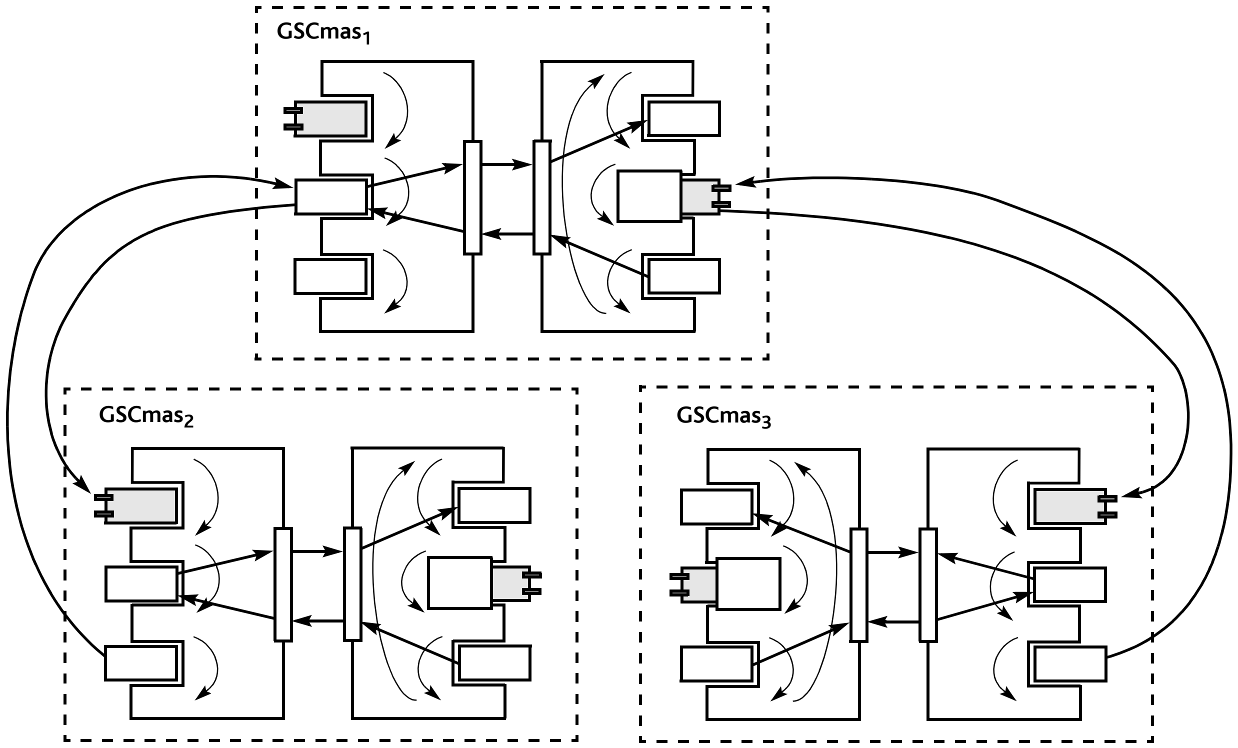


Fig 4 Chained service component interaction.

officer who is responsible for the patch covering the location of that job. However, the survey officer may already have other urgent jobs in hand and so could not complete the new, urgent job in time. In which case, the central controller (normally the team manager) telephones other survey officers to check whether they can organise their workload to take the new jobs. The decision is mainly based on the distance between their current location and the location where the new job should be performed. Ideally, the current traffic situation on the route between a candidate survey officer and the new job should be considered as well. The priority of the job the survey officer is currently performing is also an important criterion.

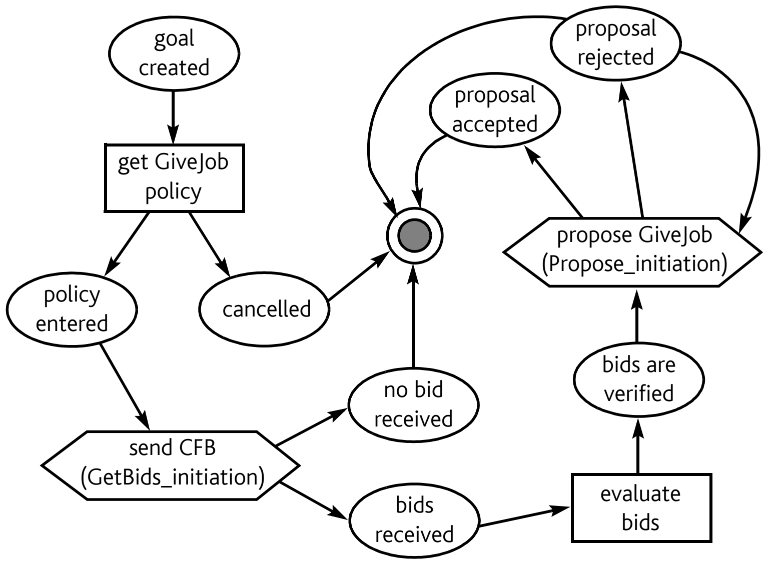
All of this process is currently manual, relies on contacting the survey officers via their mobile telephones, and can be quite time consuming.

4.2 Group decision-making process modelling

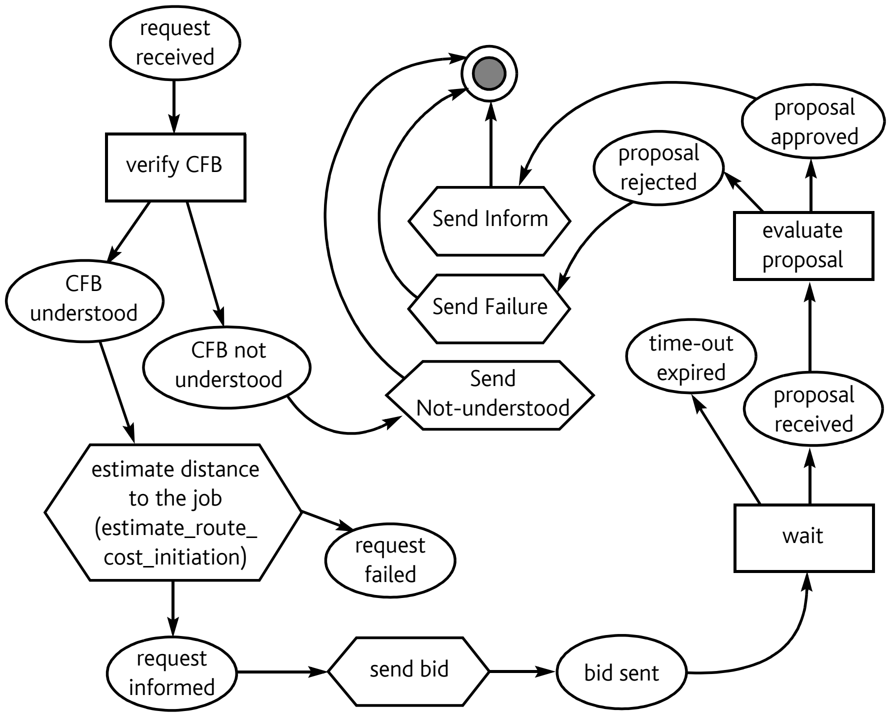
The process definition for the above decision problem is shown in Fig 5. There are two roles involved — job giver and job receiver. The job giver is an initiator role which starts the decision process and represents the interests of a worker who wants to re-allocate some of their jobs to other survey officers. The job receiver is a respondent role which reacts to requests from the job giver.

Figure 5 (a) shows the decision process of the job giver role. The whole decision process is activated when a 'GiveJob' goal is created. The first action is to set a decision policy for the process, in this case, the maximum distance of a team member from the job location and a time-out value for a colleague to respond to a proposal. This information is requested from the user via a graphical user interface (GUI) on the mobile device. If a policy is given by a user, the 'GetBids_initiation' component is used to perform a 'send call for bids (CFB)' interaction. After sending the CFB, the initiator component waits until the time out has expired for the request or until all responses have been received. The received bids are evaluated and used as inputs to the next intra-action — 'evaluate bids'. This sorts the candidates according to their distance from the job (as contained in their bids), and the sorted candidate list is passed to the interaction 'propose GiveJob'. This interaction sends a propose message which contains an offer to delegate the job to each candidate in turn, until someone accepts the proposal or all team members refuse.

Figure 5(b) shows the decision process for the job receiver role. The process is activated when a request message arrives. The respondent component uses another two service components ('get current position' and 'estimate route cost') to calculate the distance between the job and the current location of the responding agent. Then a bid based on the distance is prepared and sent back to the initiator agent. If the agent receives a 'propose' message, the user associated with the respondent agent is presented with a GUI component showing proposal details for the GiveJob. If the proposal is accepted, the whole process is terminated. Otherwise, the proposal is passed on to the survey officer next closest to the job location.



(a) Decision process definition for job giver (initiator) role.



(b) Decision process definition for job receiver (respondent) role.

Fig 5 Goal plan for dynamic job re-allocation process.

In the decision process, most of the decision-making activities are performed by personal agents, which minimises the intervention of survey officers. The job giver simply specifies the job to be given and the decision policy to use. The job receiver is saved the bother of creating and sending a bid for job re-allocation. The personal agent autonomously creates this bid by using a GSCmas to get the distance from the current location of the survey officer to the job location. However, the critical decision whether to accept the proposal from the job giver is made by the survey officer receiving the proposal, ensuring that the computer system plays a decision support, rather than command and control, role within the team. This provides a degree of empowerment for each user — a key factor in maintaining effectiveness and increasing motivation [14].

4.3 Implementation

The mPower application was developed using Personal Java and was put through a trial by a team of survey officers in the UK. Each survey officer was equipped with a 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 telephone. A GPS PCMCIA card was used to provide the location data. Secure access to the corporate intranet was via a GPRS virtual private network.

Figure 6 shows two screenshots of the mPower trial application. The left-hand image shows the screen presented to the job giver. Survey officers can list all their assigned jobs on the PDA and start a new decision process for re-allocation of work just by clicking the 'give to' button. When requested, the decision policy used by the personal agent (via the 'GiveJob options' dialogue box) must be specified. At the end of the process, the survey officer is notified whether the job has been accepted by another survey officer or rejected by every team member. The right-hand image in Fig 6 is seen by a job receiver. It shows who proposed the job and gives details, including the distance from the job receiver's current location to the job location.

The field trial proved that agent technology works for distributed teams using state-of-the-art mobile devices, GPS and GPRS. The mPower application was used by a team of survey officers to support their day-to-day work. Example comments from the users in the trial provide a deeper insight into the effectiveness of mPower for mobile workforces. One survey officer talked of the usefulness of the GiveJob service:

> He liked '... the ability of being able to send a job and for [mPower] 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.'

Another survey officer highlighted an interesting human/computer interaction (HCI) [15] issue:

> '... 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.'

This confirms the need for further research into how to transfer the normal social pressures present in 'real' contact (e.g. a person wanting to improve their conversational partner's perception of themselves) to computer-mediated contact.

5. Discussion

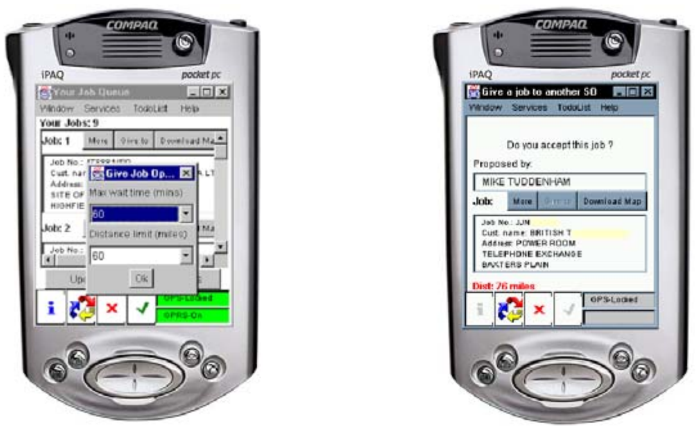
The approach in this paper provides some useful advantages when developing a GDSS for mobile teams.

Firstly, from a DSS perspective, multi-agent technology enhances the autonomy of a GDSS. Mobile workers spend the majority of their working day in the field, and as a result are frequently not able to access the computer-based information they need to do their jobs effectively. Autonomous agents can automatically execute some decision-making activities according to the workers' preferences, which minimises the interruption of a group decision process and facilitates rapid group decision making.

Secondly, from a software engineering perspective, the use of reusable GSCmas modules and group decision-making process modelling gives the GDSS a great deal of flexibility. The developer can define a new decision-making process, using existing GSCmas modules, which is then interpreted and executed by a user's personal agent.

6. Conclusions

In this paper, mPower has been proposed to support asynchronous location-based group decision making in mobile teams. mPower is based on collaborating multi-agents which automate a group decision-making process



Your Jobs: 9
Job: 1
Job No.:
Cust. nar
Address:
SITE OF
HIGHFIE
Job: 2
Job No.:
Max wait time (mins): 60
Distance limit (miles): 60

Do you accept this job ?
Proposed by: MIKE TUDDENHAM
Job:
Job No.: JJN
Cust. name: BRITISH T
Address: POWER ROOM
TELEPHONE EXCHANGE
BAXTERS PLAIN
Dist: 76 miles

Fig 6 Using mPower to support the dynamic job re-allocation process.

by creating a group decision session, negotiating with other agents, and collecting information needed for the decision-making on behalf of their users. Particularly, the personal agent utilises a group of predefined generic service components. If at design time the personal agent needs various types of information, the relevant service components are installed. A group decision-making process can be modelled by specifying the reusable service components in each phase of the decision process. Lastly, the trial of mPower by a team of survey officers demonstrates that this technology is applicable in real world scenarios where a personal agent negotiates on behalf of the user.

A number of benefits arise from using this system:

- increased response time of job negotiations by not relying on user input when dealing with computer-readable contextual information such as current location,
- reducing the work required for a survey officer to complete their task, for example, shortening the time spent arranging jobs over the telephone,
- reduced communications costs.

Typically, a manual equivalent of the GiveJob service would involve three one-minute peak-rate calls over a GSM network — for a large corporation this would cost roughly £0.15 [16]. The data needed for the automated GiveJob service with three participants (including login and job download overhead) was roughly 6 kb, costing £0.007 — a twentieth of the 'manual' method. With a very conservative estimate that at least one GiveJob process happens once a week, we can estimate that, for a 25 000-strong workforce, an enterprise would save at least £60 000 per year by using the GiveJob service.

Acknowledgements

The authors would like to thank the following people for the contribution to the work described in this paper: Rosie Butterworth and Ros Douglas for the evaluation of the field trial; Dennis Noble and Michael Tuddenham for their invaluable support with the trial; Dave Gooch and Jagdip Bath for their solutions to network security issues; and Barry Farrow for his excellent support for back-end systems. This work was part-funded by the European Commission Fifth Framework, project no IST-1999-10211.

References

1 Klusch M: 'Intelligent information agents: agent-based information discovery and management on the Internet', Springer (1998).

2 Maes P: 'Agents that reduce work and information overload.', Communications of the ACM, 37, No 7, pp 31—40, ACM Press (1994).

3 FIPA Web site — http://www.fipa.org/

4 Role modelling — http://ootips.org/role-object.html

5 Bellifemine F, Poggi A and Rimassa G: 'JADE — a FIPA-compliant agent framework', Proceedings of PAAM'99, London, pp 97—108 (April 1999).

6 Collis J, Ndumu D, Nwana G and Lee L: 'The Zeus agent building toolkit', BT Technol J, 16, No 3, pp 60—68 (July 1998).

7 Next generation mobile workforce management project details — http://ibsr.info.bt.co.uk/ [BT people only]

8 Giovanni A, Bergenti F, Poggi A and Rimassa G: 'Enabling FIPA agents on small devices', Lecture Notes in Artificial Intelligence, 2182, p 0248, Springer-Verlag (2001).

9 Caire G, Lhuillier N and Rimassa G: 'A communication protocol for agents on handheld devices', Workshop on: 'Ubiquitous Agents on Embedded, Wearable, and Mobile Devices', held in conjunction with the 2002 Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy (2002).

10 Khoshafian S and Buckiewicz M: 'Introduction to Groupware, Workflow, and Workgroup Computing', John Wiley & Sons, Inc (1995).

11 Turban E: 'Decision Support and Expert Systems: Management Support Systems', New York, Macmillan (1990).

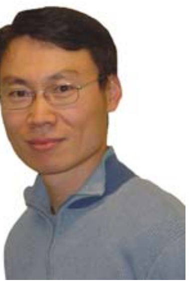
12 Peterson J L: 'Petri Net Theory and the Modelling of Systems', Prentice-Hall Inc (1981).

13 The Interaction Design Patterns page — http://www.pliant.org/personal/Tom_Erickson/InteractionPatterns.html

14 Mehandjiev N and Odgers B O: 'SAMBA — agent-supported visual interactive control for distributed team building and empowerment', BT Technol J, 17, No 4, pp 72—77 (October 1999).

15 Dix A, Finlay J, Abowd G and Beale R: 'Human-Computer Interaction', Prentice Hall (1993).

16 O2 Products — http://www.o2.co.uk/business/productsservices/



Habin Lee joined BT in 2001, and is a Senior Research Engineer with the Intelligent Business Systems Research group in BTexact Technologies. He had previously worked as an Assistant Professor in Paichai University and for the foundation of Korean Software Component Consortium. He gained a PhD in Management Information Systems and an MEng in Management Science at KAIST (Korea Advanced Institute of Science and Technology). He was a team leader of the project that developed the GIGA EXCELLENCE AWARD winning K-WFMS (knowledge-based workflow management System) at KAIST. He also holds a BSc in Management from Hankuk Aviation University. His main research interests include the application of multi-agent technology to eBusiness, design of multi-agent architectures, knowledge management in business processes, and simulation of consumer markets in the presence of network externality.

Mark Buckland joined BTexact in 2000 and is a Senior Research Engineer with the Intelligent Business Systems Research Group.

Before working with BT, he received an MSc in Intelligent Systems from Sussex University and a BEng in Software Engineering from UMIST. 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 animat perception and control architectures that pay more attention to agent/environment interaction.

John Shepherdson is head of the Intelligent Business Systems Research Group within BTexact Technologies, and leads projects on semantic integration and advanced pervasive applications. He represents BT at FIPA (the Foundation for Intelligent Physical Agents, an international standards organisation) and chaired its Gateways Technical Committee (which developed standards for interworking between software agents in wired and wireless networks) for 2 years. He obtained an honours degree in Electronics and Communications Engineering at the University of North London in 1987 and joined BT the same year. He went on to gain a masters degree in Artificial Intelligence from Kingston University in 1993. He is a Member of the IEE.