Far and A WAY: Context Sensitive Service Delivery through Mobile Lightweight PDA hosted Agents.

P.T. O'Hare **G.M.P. O'Hare** **T.D. Lowen**

Department of Computer Science
University College Dublin
Republic of Ireland
Peter.OHare@ucd.ie, Gregory.OHare@ucd.ie, Terry.Lowen@ucd.ie

Abstract

This paper introduces the Where Are You (WAY) system, a simple, yet effective application for assisting mobile users in the performance of a variety of routine tasks. The WAY system supports the mobile citizen in the location, tracking and rendezvousing with a variety of moving entities. The WAY system seeks to provide such support by deploying a rich collection of appropriate technologies including mobile agent based technologies, Geographic Information Systems and context aware mobile computing. System functionality is delivered through a collection of mobile lightweight intentional agents, which take cognizance of the memory and processing restriction of Personal Digital Assistants.

1. Introduction

This paper introduces the Where Are You (*WAY*) system, a simple, yet effective application for assisting mobile users in the performance of a variety of routine tasks. The WAY system supports the mobile citizen in the location, tracking and rendezvousing with a variety of moving entities. In particular one of the most common and costly usage of mobile devices is the synchronisation and rendezvous of people resulting in abundant inane narrative.

The WAY system seeks to provide an alternate solution to the problem by deploying a rich collection of technologies including mobile agent based technologies, Geographic Information Systems (GIS) and context aware mobile computing. Sections 2 and 3 review context aware service provision and mobile agent systems respectively. Subsequently we describe the design and realisation of the WAY system and consider our results before offering some conclusions.

2. Context Aware Mobile Computing

The concept of *context-aware computing* is concerned with systems which can detect elements of the user's environment. Contextual elements the system may detect include spatial, environmental, social, temporal and even social information. As a result of this newly acquired information the system is expected to react, providing information or services now applicable to the current context. The challenge for such systems lies in the complexity of capturing, representing and processing contextual data. Invariably user contexts are unique to the individual user and consequently this necessitates complex and dedicated tracking and profiling capabilities in order to derive the necessary leverage from such systems.

Pioneering context-sensitive systems in the early 1990's include ParcTab and Olivetti's ActiveBadge. The Cyberguide and HIPS project saw context sensitive applications used in the arena of tourists guides. Cyberguide (Abowd et al. 1997) provided a guided tour of Atalanta coupled with the ability to supply information on amenities in the users location. The HIPS tour-guide (O'Grady, O'Rafferty, O'Hare 1999) dynamically delivers multi-media presentations based on the user's location and profiles.

Several recent systems have deployed Multi-Agent Systems such as Impulse, ComMotion and Ad-Me. The Impulse (Youll et al. 2000) project provides personalized location-based information through the use of agent communication. A User Agent residing on a hand-held device creates a user profile and builds queries for the Wherehoo server and Provider Agents. The results of the queries are displayed to the user by the User Agents in the form of URLs. ComMotion (Marmasse 2000) uses a location-learning agent to observe the locations frequently visited by the user via a GPS receiver. It uses both a speech and graphical user interface, which assist in providing location, based information, displaying maps and controlling administrative functions. The Ad-me project (Hristova, O'Hare 2001) is a mobile tourist guide that proactively delivers advertisements to users based upon perceived individual user needs together with their location. It adopts a Multi-Agent System (MAS) design philosophy and strives for maximum content diffusion across HTML,WML, HDML and iMode formats.

3. Existing Mobile Agent Systems

(Gray et al. 2000) define a mobile agent as "*an executing program that can migrate, at times of its own choosing,*

from machine to machine in a heterogeneous network". Numerous competing mobile agent systems exist, exemplars are those of Telescript Agents, D'Agents, Ara, Tacoma and Grasshopper. We will consider each briefly.
Telescript (White 1994) developed at General Magic, Inc., constitutes the first commercial system specifically designed to support the development of mobile agents. Telescript agents travel between places with the *go* instruction, which captures the agent's code, data and thread state. On its new platform, the agent resumes execution from the statement immediately after the go instruction. The Ara (Peine and Stolpmann 1997) platform similarly offers support for portable and secure execution of mobile agents written in various interpreted languages on top of a common run-time core. Ara agents may migrate to another host at any point during their execution while preserving their internal state. D'Agents represents a more recent offering that supports agents written in Tcl, Java and Scheme. The ultimate goal of D'Agents is the support of applications that require the retrieval, organization and presentation of distributed information in arbitrary networks. The TACOMA (Trodheim And COrnell Mobile Agents system) unlike D'Agents and Ara does not provide automatic state capture facilities for migrating agents. When a Tacoma agent wants to migrate to another host, it packs up its code and any desired information into a *folder*. This folder is then sent to the new host, which then starts up the execution environment and calls a known entry-point in the agent's code to resume the execution of the agent. Grasshopper (Baumer et al. 1999) is the first mobile agent system compliant to the OMG MASIF (Mobile Agent System Interoperability Facility) interoperability standard.

4. The Way System

The WAY system is designed to be hosted on mobile Personal Digital Assistants (PDAS). Users would firstly subscribe to the WAY system by contacting a server in a wireless mode. Thereafter the subscriber would receive a default WAY agent mental state that would migrate to the PDA. In addition an address book is supplied which maps users to network locations (IPs). Step three then involves the issuing or acceptance of WAY requests from fellow users. Acceptance would enable the other user to know and track your whereabouts. To attain the same functionality, with regard to tracking them, then they need to accept your WAY request. Upon disconnection the agent mental state migrates to the server to be stored persistently.

The WAY system has been implemented with a combination of Smalltalk-80 and Java. Java ensures portability, and easy accessibility to Application Programming Interfaces such as Swing, Advanced Windowing Toolkit (AWT), the Java Foundation Classes (JFC), and JDBC (Java Data Base Connectivity).

Figure 1 presents the WAY schematic architecture.

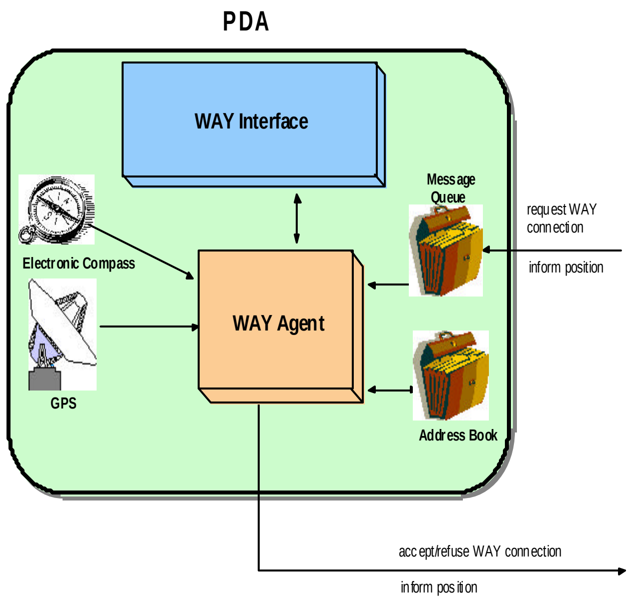
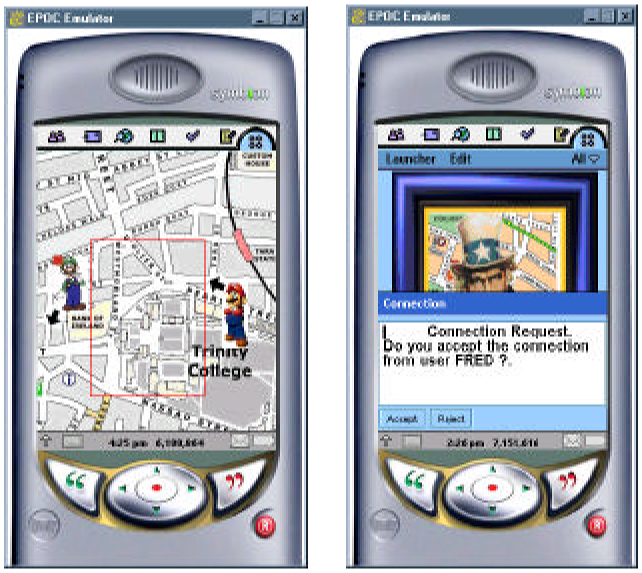


Figure 1. The WAY Schematic Architecture

4.1 The WAY Graphical User Interface

The WAY graphical user interface was delivered using the AWT (Advanced Windowing Toolkit). We embraced the less is more ethos. We sought to deliver several key interface functionalities. Firstly the ability to represent in a clear concise and visible manner the users location and orientation. As indicated in Figure 2 the former is acquired through a PCMCIA GPS card while the latter is obtained from a special custom built electronic compass accessible via a serial port. To achieve this we adopted a map based interface whereby users location and orientation would be depicted by way of a unique icon with an arrow indicating their orientation (see figure 2) As there may be a number of WAY users connected at a given instance and due to the limited screen real estate the enough to be recognizable but small enough so as not to obscure adjacent map related content. Another key element of the interface was that of zooming. We needed to have the capability of zooming in on and out of the map. Zooming in to see in detail where a single icon was but also zooming out in order to identify were others were in relation to ones self. Zooming was achieved by the use of an adjustable red square, the square is moveable so can be placed over any area of the map. The adjustable red square covers a segment of the map, this segment is magnified in direct proportion to the size of the square i.e. the smaller the square the larger the magnification and vice versa. The zoom function is activated when the user double-clicks the user interface. Figure 2 displays a WAY zoom square and the receipt of a WAY connection request from user FRED respectively.



Connection Request.
Do you accept the connection from user FRED ?

Figure 2. The WAY User Interface

4.2 Realising WAY Agents

In delivering the WAY agents we utilised the Agent Factory (AF) system. The AF System is an environment developed in part by one of the authors which supports the fabrication of agent-based applications. AF provides an integrated environment for the development of agent based systems providing a methodological framework together with an accompanying set of software tools that support the various stages in the design, specification, implementation, debugging and visualisation of agent behaviour. Detailed descriptions of AF are provided elsewhere in the literature (O'Hare et al. 1999).

In actuality it was necessary for us to reengineer the central control apparatus of AF and recast this in a lightweight Java implementation. This incarnation of AF we refer to as *Agent Factory Light*. This was necessary because of the limited computational power and memory restrictions of current PDAs and because no Smalltalk implementations exist for the EPOC operating system.

4.2.1 The Agent Factory System

AF is developed using the VisualWorks Smalltalk-80 environment. The system itself is implemented as two distinct environments namely: the *run-time environment* and the *development environment*.

The run-time environment is responsible for the delivery of agent-based applications. It is structured into two fundamental components: The Agent Virtual Machine (AVM). This delivers the support for agent realisation and operation. The AF Registry System (AFRS). This delivers support for agent interoperability.

The operation of agents is modeled as an interpreted agent programming language that is executed upon an agent interpreter embedded within the AVM. The AF System framework identifies four modules that collectively deliver the agent deductive machinery:

A *mental state architecture*: This encapsulates the agent's current model of itself and its environment.
An *agent interpreter*: which manages rule activation, mental state update and behaviour realisation
An *actuator model*: A template that is used to build actuators.
A *perceptor model*: by which world changes or perceived.

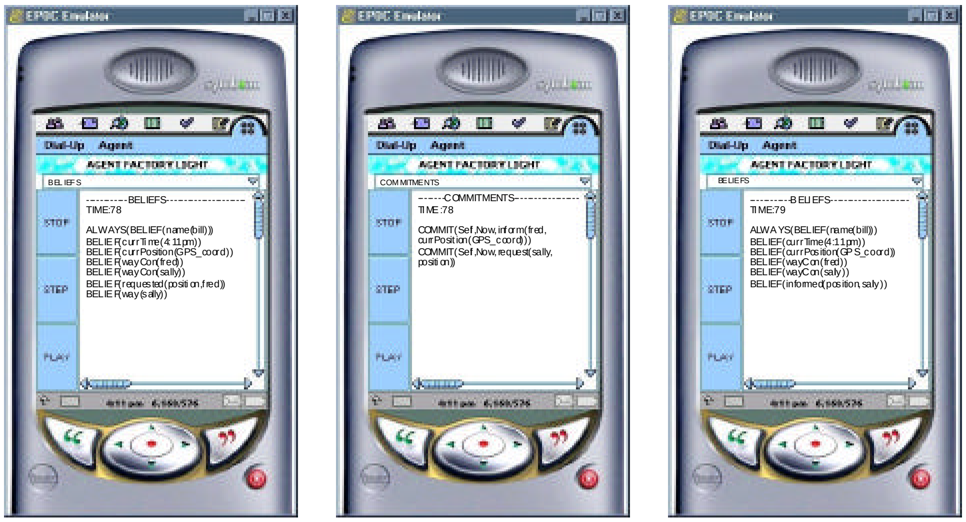
The AF Registry System (AFRS) is a global registry of agents, resources and agent platforms. The main responsibilities of this registry are to support agent name resolution, and to deliver a security model. Secondary responsibilities include ability and resource advertisement.

For all intents and purposes the *Agent Factory Light* system is merely a stripped down version of AF replicating much of the run time environment namely the Agent Virtual Machine and none of the development environment nor the AFRS. Thus agent mental states that are able to be handled within AF are similarly able to be handled by AF Light. By implication all agent development would need to be undertaken at the server side.

4.2.2 Agent Structure and the Agent Interpreter

The agent structure consists of a *mental state*, *commitment rules*, *perceptors* and *actuators*. In this sense the WAY agents constitute strong agents adhering to the broad Belief Desire Intention (BDI) class of agents. The mental state contains the agent's current model of itself and its environment. This knowledge is stored as Beliefs. Two types of belief are permitted: *current* and *temporal*. Current beliefs refer to beliefs about the current (time point) state of the environment. Temporal beliefs refer to beliefs that have some temporal quality (e.g. persistence). Commitment Rules represent the behaviour of an agent. They define the situations in which the agent should adopt a given commitment. Commitments represent the actions that an agent has chosen to realise as a result of the interpretation of its beliefs with its commitment rules. They are the results of the agent's decision-making process. Perceptors are the functional units that an agent uses to build a model of its environment. This is achieved through the implementation of a perception algorithm that, when executed, generates the appropriate perceptions and inserts them into the agent's beliefs. Actuators are the functional units that an agent uses to effect its environment. To achieve this, each actuator implements some algorithm that causes the desired effect. They are triggered by the actions in commitments that the agent has adopted. Figure 3 illustrates the update of the mental state of a WAY agent. Figure 3 (i) shows the agent's beliefs at time point 78. It has a temporal belief that it always believes its name to be 'bill'. It has current beliefs about the current time and its GPS coordinates WAY connections with other users are represented by the belief BELIEF(wayCon(USER)). In this case the agent has connections with 'fred' and 'sally'
(variables start with uppercase letters, constants with lowercase).

BELIEF(requested(position,X)) & BELIEF(wayCon(X)) => COMMIT(Self,Now,inform(X,currPosition(GPS_coord)))
BELIEF(way(X)) & BELIEF(wayCon(X)) => COMMIT(Self,Now,request(X,position))



```
---------BELIEFS-----------------
TIME:78

ALWAYS(BELIEF(name(bill)))
BELIEF(currTime(4:11pm))
BELIEF(currPosition(GPS_coord))
BELIEF(wayCon(fred))
BELIEF(wayCon(sally))
BELIEF(requested(position,fred))
BELIEF(way(sally))
```

```
------COMMITMENTS--------------
TIME:78

COMMIT(Self,Now,inform(fred,
currPosition(GPS_coord)))
COMMIT(Self,Now,request(sally,
position))
```

```
---------BELIEFS-----------------
TIME:79

ALWAYS(BELIEF(name(bill)))
BELIEF(currTime(4:11pm))
BELIEF(currPosition(GPS_coord))
BELIEF(wayCon(fred))
BELIEF(wayCon(saly))
BELIEF(informed(position,saly))
```

(i) Belief Set at Time 78 (ii) Commitments at Time 78 (iii) Belief Set at Time 79

Figure 3. WAY Agent Mental State

The first rule states: if the agent believes its position was requested by another agent and it believes that it has a WAY connection with that agent then it commits itself, *now*, to inform that agent of its position. Figure 3 (ii) illustrates that the agent has used this commitment rule and has now committed itself to informing Fred of its position. The second rule states: if the agent believes it wants the position of another agent and it believes it has a WAY connection with this agent then it commits itself, now, to requesting the agent for its position. Again figure 3 (ii) illustrates that the agent has used the commitment rule and has now committed itself to requesting Sally for her position. Figure 3 (iii) displays the beliefs at the next time step and the last belief shows that Sally has informed the agent of her position. We thus see that the mental state of the agents are dynamic with beliefs and commitments being subject to change. The commitment rules however remain fixed and have admissibility functions that are expressed in terms of mental state and message conditions.

4.3 Agent Mobility

Agent mobility is of paramount importance within the WAY system enabling agent migration and discovery of system resources and potential load balancing. Mobile agents move by transmitting their code from one computer platform to another in an action often termed as migration. To ensure that migration can occur, four main criteria must be dealt with:

1. There must exist an appropriate environment at the destination to receive the agent;
2. The agent must be able to transmit itself correctly;
3. This environment must be able to reconstruct the received agent;
4. The environment must be able to resume execution of the agent.

AF supports migration from one AF system to another through cloning (Collier et al. 2000). When an agent wants to migrate it informs the destination that it wishes to do so. The destination creates an agent of the appropriate agent design. The mental state of the agent is only then copied and transmitted to the required destination. Upon receipt it is incorporated into the new agent. The old agent is then disposed of and the new agent begins execution. In the WAY system it is assumed that each user has an agent interpreter pre-installed on their device which manages the execution of WAY agents. When a user subscribes for the first time a default agent class is instantiated and uniquely named on the AF Server. A copy of the mental state of this agent is then sent to the users device along with a list of other users. The AF Light system on the device then uses this mental state to create and execute the WAY agent. Periodically AF Light sends back updates of its mental state to the clone on the server. This is a safeguard in case anything causes the agent on the device to terminate prematurely. Upon resumption a roll back recovery enables the clone on the server to migrate back to the destination PDA device.

Mobile agent systems provide two classes of migration:

Strong Migration where the agent's object state, code and control state is captured. Upon migration this allows the agent to resume execution on the new machine from the exact point that it left off.

Weak Migration where only the agent's object state and code is captured. Upon migration the system calls a known entry-point in the code to restart the agent on the new machine.

For the end user, a mobile agent system that uses strong migration can be seen to be more advantageous in that the agent can migrate at any time point. Using weak migration the agent can only migrate at set points in its interpreter's cycle. However weak migration has an advantage in that there is less to be transported when an agent is on the move it travels lightly as it were.

The Java Virtual Machine (JVM) as it stands does not support the capture of thread states. Therefore, most commercial Java based systems use weak migration. Java Virtual Machines have been modified in order to provide thread state capture (Bouchenak 1999), but in general the market dictates that mobile agent systems run on unmodified Java Virtual Machines. Recent work has been conducted on trying to capture thread states without modifying the JVM (Truyen et al 2000).

Agent Factory supports weak migration (like TACOMA and Grasshopper, unlike Telescript, D'Agent and Ara which offer strong migration) in that only the mental state of the agent migrates. This makes the task of migrating an agent from a Smalltalk environment (AF) to a Java environment (AF Light) and vice versa, a much easier lightweight task. The same mental state can run on both the AF and the AF Light interpreters without modification. While AF Light is not currently compliant with the FIPA and OMG MASIF standards for agent mobility, its modular structure allows for the incorporation of such standards in the future.

5 Conclusions

Within this paper we have introduced the WAY system which supports the mobile citizen in the location, tracking and rendezvous with a variety of moving entities. It accepts WAY connections from a fellow users and in a bi-directional sense posts location and orientation updates. These are subsequently depicted on a map based interface. Additional functionality allows users to rendezvous with other entities like taxis buses and so forth.

The WAY system embraces an agent-oriented design supporting weak migration of strong BDI agents across a wireless network. Laboratory prototypes have been tested using an Ericsson R380. In addition user trials have been conducted using a Compaq Ipaq 3660 equipped with a dual PCMCIA sleeve which is used to accommodate a PCMCIA GPS and a Nokia card phone 2.0. The former provides the localization data and the later the wireless communication infrastructure. Development on this device utilized the Jeode EVM. At present detailed field trials are underway and the results thus far are favourable.

Acknowledgements

This work was funded in part through the UCD-Ericsson Mobile Computing Initiative. We gratefully acknowledge this support.

References

Abowd, G.D et al. 1997. Cyberguide: A mobile context-aware tour guide. Wireless Network 3, 421-433.

Baumer, C., Breugst, S., Choy, S., Magedanz, T. 1999. Grasshopper – A Universal Agent Platform Based on OMG MASIF and FIPA Standards. IKV++.

Bouchenak, S. 1999. Pickling thread states in the Java system, in Proc. of the third European Research Seminar on Advances in Distributed systems (ERSADS'99).

Collier, R.W., Rooney, C.F.B., O'Donoghue, R.P.S., O'Hare, G.M.P. 2000. Mobile BDI Agents, 11th Irish Conf. on Artificial Intelligence & Cognitive Science, University College Galway.

Gray, R., Kotz, D., Cybenko, G., Rus, D. 2000. Mobile agents: Motivations and state-of-the-art systems, In Jeffrey M. Bradshaw, editor, Handbook of Agent Technology, AAAI/MIT Press.

Hristova, N., O'Hare, G.M.P. 2001. "Ad-me: A Context-Sensitive Advertising System, in Proc of the 3rd Int'l Conf. on Information Integration and Web-based Applications & Services. (II-WAS), Pub. By Austrian Computer Society, Linz Austria.

Marmasse, N. September 25-27, 2000. Location-aware information delivery with comMotion. Proc. of the 2nd Int'l Symposium on Handheld and Ubiquitous Computing (HUC), Bristol, UK.

O'Hare, G.M.P., Duffy, B.R., Collier, R.W, Rooney,C.F.B., O'Donoghue, R.P.S. 1999. Agent Factory: Towards Social Robots, Proc. 1st Int'l Workshop of Central and Eastern Europe on Multi -Agent Systems (CEEMAS'99), St.Petersburg, Russia.

O'Grady, M.J., O'Rafferty, R.P., O'Hare. G.M.P. December 1999. A Tourist-Centric Mechanism for Interacting with the Environment. 1st International Workshop on Managing Interactions in Smart Environments MANSE'99, Dublin.

Peine, H., and Stolpmann, T. 1997. The Architecture of the Ara Platform for Mobile Agents, in Kurt Rothermel, Radu Popescu-Zeletin (Eds.): Proc. of the Int'l Workshop on Mobile Agents MA'97 (Berlin, Germany), April 7-8th. LNCI No. 1219, Springer Verlag.

Truyen, E., Robben, B, Vanhaute, B., Coninx, T., Wouter, J., Verbaeten, P. 2000. Portable Support for Transport Thread Migration in Java, submitted to AM2000.9.

White, J.E. 1994. Telescript Technology: The Foundation of the Electronic Marketplace, General Magic Inc.

Youll, J., Morris, J., Krikorian, R., Maes, P. June 3 - June 7, 2000. "Impulse: Location-based Agent Assistance" Software Demos, Proc. of the Fourth Int'l Conf. on Autonomous Agents (Agents 2000),Barcelona,Spain.