Expert Systems with Applications 27 (2004) 481–492

Expert Systems with Applications

www.elsevier.com/locate/eswa

A development environment for intelligent applications on mobile devices

Lynne Hall[a,*], Adrian Gordon[b], Lynne Newall[c], Russell James[b]

[a]School of Computing and Technology, University of Sunderland, Sunderland, SR6 0DD, UK
[b]Mimosa Wireless Ltd., 13 Moorlands, Consett, Country Durham, DH8 OJP UK
[c]School of Informatics, University of Northumbria, Newcastle, NE1 8ST, UK

Abstract

Mobile computing devices are becoming increasingly prevalent in a huge range of physical guises, offering a considerable market opportunity for software. Mature Artificial Intelligence technologies offer the potential to create compelling software for the mobile platform. However, few intelligent applications have been developed for mobile devices. This lack of development is related firstly to mobile device limitations, such as memory and processing power, and secondly to the requirement for portability due to the diversity in the mobile device market. This paper presents a development environment for intelligent applications for mobile devices that successfully addresses both of these issues. Case studies of intelligent applications developed with this development environment are briefly described. Some conclusions are presented and directions for future research considered.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Mobile devices; Intelligent applications; Expert systems; Development environments

1. Introduction

The attraction of the mobile device market for software deployment is considerable, with penetration close to 80% for mobile phones in the UK (Peters, 2002) and expected to remain at this level (Kacker, 2002). Gartner reports that the mobile market is continuing to grow with expectations that between 450 and 460 million mobile phones will be sold globally in 2003 (Reuters, 2003). Handheld sales increased by 51% in Europe in 2002, with Palm, who lead the data-centric handheld segment experiencing a 45% growth Canalys (2003). Within the phone market, smart phones (e.g. Nokia 3650 and Sony Ericsson P800) continue to provide much of the growth showing the increasing popularity of integrated handheld devices (Canalys, 2001).

The prediction is that application development for mobile devices will explode (Cripps, 2001). Although this explosion may occur, currently there is considerable duplication within the application market, with the majority of applications for mobile devices tending to be scaled down versions of Office software, personal productivity tools, enhancement software (e.g. file management, ring tones) dictionaries and games. However, to ensure their success mobile devices, particularly handhelds need useful and compelling applications (Craig, 2002) or as Symbian's David Levin has prophesised, 'The PDA is dead.' (BBC, 2003)

Artificial Intelligence (AI) is a mature field with proven technology for activities such as data mining, information retrieval and natural language processing (Menzies, 2003). It has had increasing success in domains such as finance (Nedovic & Devedzic, 2002), games (Smith & Egenfeldt-Nielsen, 2003) and medical diagnostics (Willems, 1991). There are many potential applications for Intelligent Systems which would offer obviously improved usefulness by being supported on mobile devices. Examples might be:

- A system to help a horticulturist diagnose and propose treatments for plant disorders out in the field.
- A system that helps a Health and Safety officer conduct detailed compliance reviews on such things as fire risks whilst on site.
- A system that helps an athlete to plan, monitor and change an exercise regime while he or she is actually in the Gym.
- A system that negotiates on behalf of a user for a hotel room which has the right set of features and facilities at the right price.

Currently, although the technologies for implementing intelligent applications are well established, they have not

* Corresponding author. Tel.: +44-191-515-3249; fax: +44-191-515-2781.
E-mail address: lynne.hall@sunderland.ac.uk (L. Hall).

0957-4174/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2004.05.010

yet been adapted for delivery via mobile computing devices. Two main reasons can be suggested for this. Firstly, that mobile devices can in some cases be severely constrained in terms of processing power and memory size, both of which are required for successful deployment of many intelligent applications. Secondly, the mobile device market is highly fragmented with a large number of different hardware, software and operating system configurations existing in the devices currently on the market. It is likely that no one dominant product will emerge (Smiley, 2003), thus requiring the portability of applications across diverse hardware and software platforms.

The obvious approach to the development of intelligent systems for deployment on mobile devices might appear to be to adopt a 'thin client' strategy, whereby a mobile device connects wirelessly to a remote, already existing intelligent system. However, there are several reasons why this is not the best approach, including cost (most wireless carriers charge for connection by the byte or by the minute), reliability (there are still problems of network coverage and performance on devices (Evans & Baughan, 2000), constant connection impact on battery life (Canalys, 2001) and security. Here, we present an approach taken by Mimosa Wireless Ltd. to mobile intelligent application development that seeks to address the problems of the constrained nature and diversity of mobile devices.

Conventionally, intelligent systems are considered to require very powerful computing environments, with powerful processors and a lot of memory. However, a lot of this power is in fact only required in the process of developing intelligent systems. Intelligent systems are typically developed using some kind of prototyping methodology, where tools are used which enable systems to be constructed and tested rapidly. It is actually the development tools which require the power. Once a system has been constructed, tested, and verified, it can actually be executed with far fewer computing resources. In this paper we discuss the approach taken to provide a powerful set of tools for the construction of an intelligent system. Once the system has been constructed and tested, an executable version of the system can be derived, and ported to a range of mobile devices. The development environment essentially consists of an interpreter for programs, written in a simple programming language developed to enable intelligent system construction, and a compiler which can generate executable versions of the program targeted at a range of different devices.

The intelligent applications developed are based on a simple paradigm: production systems using forward chaining. Not only has this proven to be a very powerful programming approach for a range of different classes of intelligent systems, but additionally it has a number of characteristics that lend themselves to portability. The implementation of this programming approach is based around a very simple programming language for constructing intelligent systems, where systems are expressed as a set of facts and inference rules.

The approach described in this paper has been used to construct a number of intelligent systems, of different types, and targeted at a range of different mobile devices. These include an expert system for diagnosing and treating high levels of blood cholesterol and an intelligent agent which aims to negotiate for an appropriate hotel room on behalf of a user.

Section 2 discusses earlier approaches to intelligent application development environments for mobile devices. Section 3 discusses our development environment and describes the programming language that was developed. Section 4 briefly describes a number of intelligent applications that have been developed using this environment. Section 5 describes some ongoing development projects using the approach described in this paper and Section 6 presents some conclusions.

2. Approaches to intelligent applications development for mobile devices

This section outlines a number of different approaches to intelligent application development for mobile devices. Initially, a number of platform specific intelligent applications are illustrated. We then discuss development environments for intelligent applications for mobile devices, identifying the potential and problems of those environments.

Early approaches to intelligent applications development for mobile devices were proprietary solutions with development restricted to a limited set of devices. The main focus of development was on obtaining an acceptable performance level and this typically resulted in severely restricted (if any) portability. Several successful intelligent applications have been developed for specific platforms, including aids for visitors to museums (Oppermann & Specht, 1999) and art galleries (Brachtl, Slajs, & Slavik, 2000). Within medicine, where up to 80% of physicians use PDAs (Mozelak, Glassman, & Johnston, 2001), decision aids that have emerged are typically platform specific (Hameed, 2003) such as the National Cancer Institute's (NCI) Clinical Trial Decision Aid (Whiteis, McGovern, & Johnston, 2001), developed for the PalmOS.

Whilst intelligent applications have considerable potential within fields such as medicine, they will have only limited value if restricted to a single platform. Not only is there huge diversity in the mobile device market with no dominant platform (Canalys, 2003), but additionally, new devices, offering a diverse range of services, continue to emerge. Permitting portability across multiple platforms for intelligent applications has been attempted through the use integrated development environments that permit the development of intelligent applications. Here we briefly discuss a number of applications and environments that have been developed to permit the development of intelligent applications for mobile devices.

In (Albuquerque, Guedes, Filho, Robin, & Ramalho, 2002), an embedded inference engine for handheld devices developed in J2ME is presented. This provides a deduction mechanism that was created using an object-oriented approach, with objects representing facts, rule conditions implemented as a conjunction of object method calls and rule actions represented as object methods. However, when the intelligent application was deployed on mobile devices, only a lightweight rather than a direct mapping from the original desktop deduction could be achieved.

The objectives of lightweight and extensible agent platform (LEAP) (Bergenti & Poggi, 2001) are to develop a FIPA compliant platform for fixed and mobile devices, ensure that this platform can run in different operating systems and to enable the platform to reconfigure with respect to the capabilities of a target device. LEAP provides basic technology for running FIPA compliant agents (FIPA, 2003) in Java enabled devices, with sufficient resources and connected to a mobile or a fixed network. LEAP is a synthesis of earlier agent prototyping environments, created using the ZEUS development environment (Nwana, Ndumu, Lee, & Collis, 1999) and deployed using a revised JADE run time environment (Bellifemine, Poggi, & Rimassa, 1999). LEAP permits the development of agents in a mobile network using an open infrastructure.

KSACI (Albuquerque, Hubner, de Paula, Sichman, & Ramalho, 2001) is a tool that proposes a communication infrastructure among agents running in handheld devices. KSACI allows agents embedded in handheld devices to exchange information and knowledge with other embedded agents or with agents located in desktops.

A database perspective is taken by Loke (2002) to the storing and running of agents based on mobile devices. This approach uses a simple BDI-model (Beliefs–Desires–Intentions (BDI) (Rao and Georgeff, 1995)) of agents. BDI agents like other intelligent applications need computational capacities (storage, connectivity) that are not abundant in limited devices. The memory and processing power limitations of mobile devices are dealt with by reducing execution through caching of agents and parts of agents, and enabling queries to be answered locally (without connection), faster, and processed on the client side (minimizing processing on the server side). Some success has been noted with this approach, however, agent updates need special attention in relation to conflicts and consistency.

I–X technology for intelligent systems (Tate, 2001) has been used to create applications that provide intelligent planning information (Ala-Siuru, Belkin, Mayora-Ibarra, & Mizzaro, 2003) for mobile devices. Intelligent planning information is delivered to users of mobile devices that are participating in collaborative planning environments. To enable the application to operate effectively within the constraints of the mobile devices, a user requirements based model is used. This results in only those features which will be directly used by the user being available on the mobile device, providing only a subset of planning information, processes, mechanisms and features.

The central control apparatus of Agent Factory (O'Hare, Duffy, Collier, Rooney, & O'Donoghue, 1999) was re-engineered and recast in a lightweight Java implementation, Agent Factory Light (O'Hare, O'Hare, & Lowen, 2002) for intelligent agent development for mobile devices. This redevelopment was necessary because of the limited computational power and memory restrictions of current PDAs. The stripped down version of Agent Factory replicates much of the run time environment namely the Agent Virtual Machine but none of the development environment. The agents developed by Agent Factory Light are restricted in comparison to those developed for the desktop, with considerably less functionality offered on constrained devices, for example a very limited number of agents or reduced functionalities (O'Hare et al., 1999).

The majority of intelligent applications developed for both the mobile and static sectors are not identical nor even particularly similar across platforms. Rather in the mobile variant we are seeing reduced capabilities or sub-sets of the intelligent application expertise. In the approach that we have taken, which is discussed in the following section our intention is to maintain the PC-based functionality and capability of the intelligent application but on a mobile platform.

3. The development environment

The development environment for intelligent systems targeted at mobile devices consists of a programming language for expressing intelligent systems, and an integrated development environment (IDE). This IDE provides an interpreter for the programming language, tools for testing and debugging the systems written using this programming language, and a compiler for generating executable versions of a functioning intelligent system which can be targeted at a range of mobile devices.

3.1. Mimosa: a simple programming language for intelligent systems

Similar to agent factory (Collier, O'Hare, Lowen, & Rooney, 2003) and Jack Intelligent Agents (Busetta, Ronnquist, Hodgson, & Lucas, 1999) with their purpose built agent programming language, the development environment involves the use of a programming language for expressing intelligent systems.

The programming language, Mimosa, that forms the basis of the work described here is a very simple language that enables intelligent systems to be defined as production systems, i.e. they are expressed as sets of facts and inference rules. As such, the language borrows heavily from earlier production systems languages, such as the OPS class of languages (Brownston, Farrell, Kant, & Martin, 1985).

'Facts' in this language are structured data items that might be termed 'structures' in C or Pascal like languages. Particular structured data types are defined using the 'literalize' construct of the language (borrowed from OPS):

```
(literalize patient:string name
:string sex
:integer age)
```

This states that a patient will consist of 'name', 'sex' and 'age' data elements. Each of them is typed, name and age are strings, age is an integer. The other possible data type is 'atom' (e.g. the default). An 'atom' is just a unique symbolic value. Although 'real' datatypes exist in the language, they are not as yet implemented in the development environment (many current mobile devices do not support floating point numbers).

To actually create an instance of a predefined literal, the make construct is used:

```
(make patient ^name 'jim' ^sex 'male' ^age 55)
```

The caret symbols proceed one of the attribute names of a predefined literal, such as 'goal' or 'object'. In this application the literal 'goal' has attributes 'g-status', 'g-type', and 'g-obj', the literal 'object' has attributes 'o-name', 'o-at', and 'o-on'.

The other essential component of programs written in this language is provided by a set of inference rules. These are expressed using the 'p' construct (for 'production rule'). The general syntax is as follows:

```
(p < production rule name >
 < pattern_1 >
...
 < pattern_n >
– >
 < action_1 >
...
 < action_n > )
```

Here is an example, taken from the famous 'monkeys and bananas' planning problem (the elements in square brackets represent variables):

```
(p mb1
   (goal ^g-status active ^g-type holds ^g-obj [w])
   (object ^o-name [w] ^o-at [l] ^o-on ceiling)
   – >
   (make goal ^g-status active ^g-type move ^g-obj
ladder ^g-to [l]))
```

To paraphrase this rule in English: 'If you have a currently active goal which is to hold some object w, and if that object w is currently found on the ceiling at some location l, establish a new active goal to move the ladder to the location l'.

Actions in this language are limited to make (create a new fact), modify (modify an existing fact), remove (remove an existing fact), and bind (establish a variable binding for later use).

However, the language is extensible, by use of what are termed 'external functions'. Here is an example, taken from the Cholesterol Expert System described in Section 4.1:

```
;input initial data about the patient. Comments are
preceded by semicolons
   (p getInitialData
   (goal ^arg1 getInitialData)
   – >
   (dialogBox "Data must be provided for LDL, HDL
   and overall
   cholesterol levels gathered within the last five
   years")
   …);details omitted for clarity
```

In the above production rule the dialogBox function is an external function which is used to present a message to the user before continuing. It is defined as an external function in the source code as follows:

```
(external dialogBox 31:integer 1:string)
```

'external' is the keyword indicating that what follows is the definition of an external function, 'dialogBox' is the name of the function, '31' is a unique numeric identifier (within a given application) for this function (used later by the compiler-see Section 3.3), 'integer' is the type of argument returned by the function, '1' is the number of arguments taken by the function, and 'string' is the type of that argument.

External functions can be used as actions in their own right, as shown above, or can be embedded within make, modify or bind constructs. They are typically used for interacting with the GUI of a system, accessing network resources, interacting with external systems, etc. The aim of using this external function mechanism is to isolate the 'intelligent' aspects of a system from the other elements, such as the GUI and the network environment, as an aide to portability. The core elements of the system (rules and facts) remain the same whatever the target device, only the external functions need to be re-implemented on a device by device basis. This is described further in Section 3.3.

3.2. The MADE integrated development environment

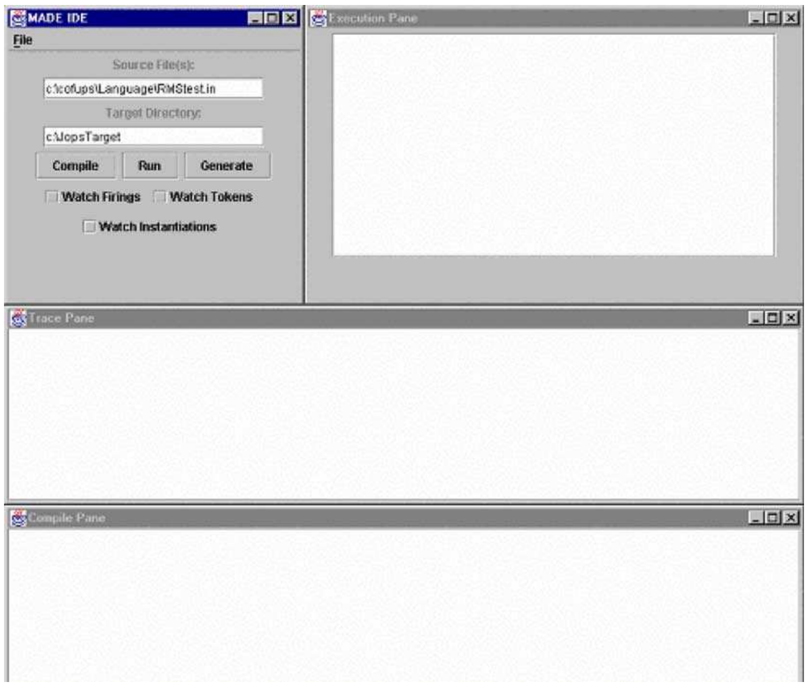
The IDE, known as MADE (Mimosa Application Development Environment) is used to construct prototype intelligent systems that are ultimately to be targeted at a range of mobile devices. MADE consists of a set of tools for interpreting the Mimosa programming language and a tool for generating executable versions of the working software that are to be ported to mobile devices. MADE exists in both console and GUI versions. Both of these are written in Java. Fig. 1 shows a screenshot of the GUI version.

In common with other intelligent application development environments, such as Jess (Friedman-Hill, 2001) the basis of our development environment is the highly efficient Rete Algorithm (Forgy, 1982). This algorithm improves the speed of forward-chained rule systems by limiting the effort required to recompute the conflict set after a rule is fired. This efficiency is achieved by remembering past test results across iterations of the rule loop and limiting testing only to new facts, reducing the computational complexity per iteration Such limitation of effort greatly speeds up performance and requires less processing power due to the reduction in recomputation. This reduction of requirement on processing power is of obvious relevance to mobile devices, however, the downside of this algorithm is that like many other AI technologies to improve performance it has high memory space requirements. Here, we discuss our approach to enable intelligent applications developed using the Rete algorithm to be deployed on constrained mobile devices.

The Rete algorithm is used to interpret programs written in the Mimosa programming language. With the Rete algorithm, the entire set of inference rules is turned into a network of interconnected nodes, with different rules sharing parts of the network. Interpreting the rule base then becomes a problem of interpreting this network of nodes. The first stage in running a program is therefore to compile the Rete network for the rules that make up the program. The 'compile' button shown in the top left corner of the IDE performs this task.

Output from this compile process will appear in the Compile Pane of the IDE-this output will include such things as syntax errors, for example. Once a program has been processed in this way, it can be executed, using the 'Run' button shown on the IDE. Rules will be matched and executed using the Rete algorithm on the network created by the compilation process. Various options for tracing the execution of the program can be set (as indicated by the checkboxes labelled 'watch...' in the IDE). It is possible to see which rules are fired, it is possible to watch which tokens (facts) are added to the knowledge base, and it is possible to watch which rules are instantiated (i.e. matched) as execution progresses (rules may be instantiated, but never fired, due to the interpreter's conflict resolution strategy). Output from tracing execution appears in the Trace Pane.

A prerequisite for being able to execute a program using MADE is that all of the external functions declared within the source code of the program have been implemented within the IDE. A new application may require a new set of external functions which manage the GUI of the target application, use network resources, connect with external systems, data storage devices, etc. As well as writing the code for the underlying intelligent system in the language described in Section 3.1, code must also be written, in Java,



Source File(s): c:\cofups\Language\RMStest.in

Target Directory: c:\JopsTarget

Fig. 1. A screenshot of the GUI version of MADE.

for all of the external functions to be used within the application. Thus, MADE is gradually extended over time with libraries of external functions. Some of these libraries may be used across a range of applications, but many will be used only for one specific application (this will be typically the case with the GUI, for example).

Once all external functions have been implemented, the application can be executed within MADE. Output from the execution of the application will appear in the Execution Pane of the IDE. This is where diagnostic output will appear, for example. Any interface components of the application will also appear here. Within MADE, interface components will use the underlying Java interface widgets. At this stage, the focus for the GUI of the intelligent application will be on functionality rather that usability-the interface will of course be very different on any actual target mobile devices.

3.3. Generating target applications

Once a working intelligent system has been constructed using MADE, target versions of the working software can be generated for a range of mobile devices. This is achieved by using the ‘Generate’ button on the MADE interface of Fig. 1. Choosing ‘Generate’ will essentially produce versions of the intelligent application’s knowledge base expressed in a number of dialects of various programming languages, targeted at different mobile device categories. Each of the derived knowledge bases are then combined with an Execution Environment (termed a Mimosa Execution Environment, or MEE), which is itself targeted at a particular device category. Each combination can then be tested using an emulator, before being transferred to the hardware in question. This scheme is illustrated in Fig. 2.

The most important aspect of the knowledge base that is generated by MADE is the Rete network that represents the rule base of the application. In all cases, this Rete network is linearised, and each node is represented by a 16 bit byte, regardless of the target device. This has a significant impact on the memory requirements for this algorithm, reducing these to a level that is attainable on a mobile device. Each of the implementations of the MEE is essentially an interepreter for a Rete network which is represented by a block of these 16 bit bytes.

At present, different MEE versions exist which are written in various dialects of Java and C++. There are Java Versions written to conform to Personal Java and J2ME (Java 2 Micro Edition) standards, and C++ versions targeted at PalmOs and Symbian OS. This selection actually covers a significant portion of the current mobile device market, from laptops running Microsoft Windows, to small cellphones running proprietary Operating Systems. It should be a relatively straightforward task to extend this to cover Qualcomm’s BREW platform, or Microsoft’s WinCE for cellphones, for example.

Although the core elements of the Rete interpreter are implemented for all of the platforms described above, in each case, any external functions that were used within MADE to develop an intelligent application must be re-implemented in each of the target applications for each mobile platform. This is usually just a case of a function for function re-write of the external functions implemented within MADE.

4. Applications

The MADE and MEE have been used to create a number of intelligent applications. Here, two of these are discussed, firstly an expert system that diagnoses and proposes treatments for high levels of blood cholesterol, and secondly, an intelligent agent that negotiates for hotel rooms. Both of these applications have been extensively tested on a variety of devices and device emulators and have

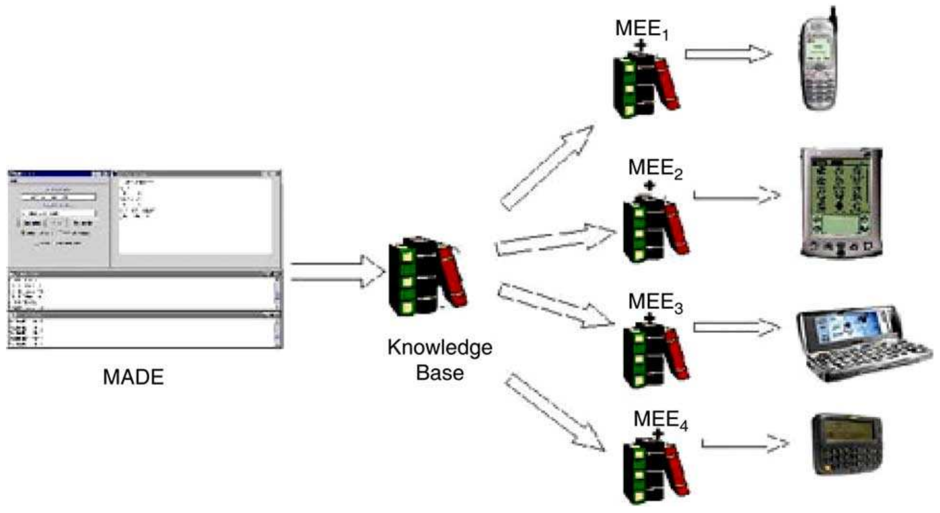
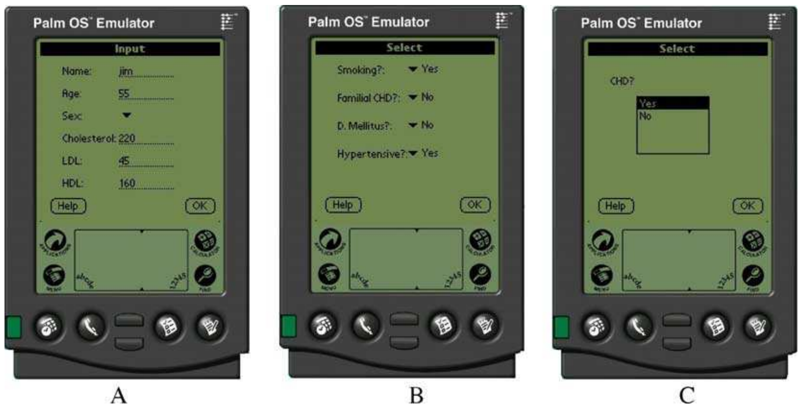


Fig. 2. Generating intelligent applications targeted at different device categories.



Input
Name: jim
Age: 55
Sex:
Cholesterol: 220
LDL: 45
HDL: 160

Select
Smoking?: Yes
Familial CHD?: No
D. Mellitus?: No
Hypertensive?: Yes

Select
CHD?
Yes
No

Fig. 3. The blood cholesterol expert system running on a PalmOS emulator.

acceptable performance in terms of robustness, reliability and speed.

4.1. Coronary heart disease expert system

The first case study illustrating the utility of the technologies described above involves an application that is a typical expert system. The expert system illustrated here captures the expertise of medical specialists who are expert in the diagnosis and treatment of high blood cholesterol levels. The expertise embodied in this system is based on that to be found in the US National Cholesterol Education Program 'Second Report of the Expert Panel on Detection, Evaluation, and Treatment of High Blood Cholesterol in Adults'(National Cholesterol Education Program, 2002).

The Blood Cholesterol Expert System was developed using the MADE system. Once the Blood Cholesterol expert system was developed on the MADE system, its knowledge base was extracted, and, combined with a suitable Execution Environment, used to generate versions of the application to run on a variety of different mobile devices. Figs. 3 and 4 illustrate the system running on an emulator for Palm OS (the system also runs on the Palm hardware itself).

In order to diagnose and propose treatments for high levels of blood cholesterol, the system first needs to gather some initial information about a patient (Fig. 3A), such as name, age and sex, and their most recent blood cholesterol levels, as measured in a blood test (there are three such measures, Overall, HDL-High Density Lipoprotein, and LDL-Low Density Lipoprotein levels). The system then needs to gather information about the patient's lifestyle and health history, such as whether they smoke, or have a family history of coronary heart disease (Fig. 3B). Based on the answers to these questions, the system may ask for other pieces of information about the patient, such as whether they are already receiving treatment for Coronary Heart Disease (Fig. 3C).

The system is able to provide help if an explanation of the questions being asked is required (Fig. 4A). Finally, the system provides a diagnosis of the patient's Blood Cholesterol levels, and proposes a treatment (Fig. 4B). Based on the information provided about a patient, the system can produce a variety of different diagnoses, and a range of different treatments (including general lifestyle advice, dietary or drug based therapies).

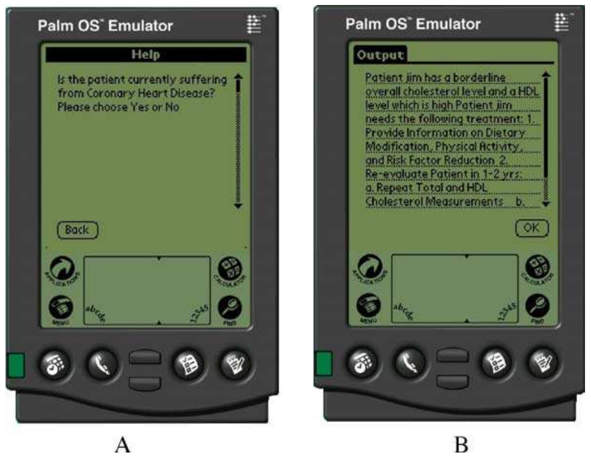
A well as running on Palm OS based devices, the system also runs on Symbian OS devices, such as the Nokia 9210i Communicator (Fig. 5), and mobile phones that are able to run J2ME (Fig. 6).

The Blood Cholesterol expert system application illustrates that it is possible, using MADE, to develop a complex expert system application that can be deployed on a variety of mobile devices of different types.

4.2. An intelligent agent hotel negotiator

Intelligent agents can be constructed and executed using exactly the same software, MADE and MEE, as described in Section 4.2. Intelligent agents have been used for many applications, but one of the tasks for which they are particularly well suited is negotiation in some kind of marketplace (Lomuscio, Wooldridge, & Jennings, 2001).

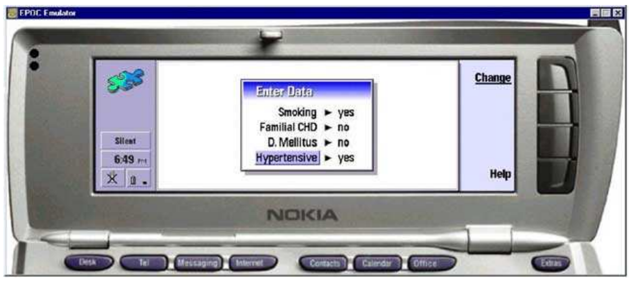
As an example of this approach to constructing Intelligent Software Agents, Mimosa Wireless Ltd. has



Help
Is the patient currently suffering from Coronary Heart Disease? Please choose Yes or No

Output
Patient jim has a borderline overall cholesterol level and a HDL level which is high Patient jim needs the following treatment: 1. Provide Information on Dietary Modification, Physical Activity, and Risk Factor Reduction 2. Re-evaluate Patient in 1-2 yrs: a. Repeat Total and HDL Cholesterol Measurements b.

Fig. 4. A sample help screen and output screen from the blood cholesterol expert system.

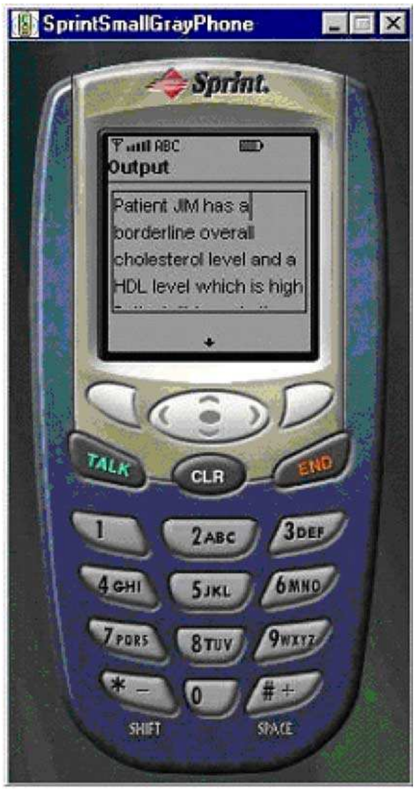


Enter Data
Smoking ► yes
Familial CHD ► no
D.Mellitus ► no
Hypertensive ► yes

Fig. 5. The blood cholesterol expert system running on ay Nokia 9210i communicator emulator.

created a system which is intended to enter into a negotiation process whose goal is to locate a hotel room which provides the best fit to the end user's requirements. These requirements are expressed in terms of a combination of price and features.

The scenario within which this demonstrator application operates assumes that a traveller has arrived at a particular location without having a place to stay, perhaps due to a late change of itinerary. Under normal circumstances, this would require the traveller either to contact a number of hotels him or herself, or use (and pay for) some kind of booking agency. The Intelligent Hotel Room Negotiator, running on a mobile device which has a network connection (e.g. a Smartphone such as the Nokia 7650), itself contacts the Intelligent Agents of a number of hotels, and negotiates directly with them to find a suitable room.



Output
Patient JIM has a borderline overall cholesterol level and a HDL level which is high

Fig. 6. The blood cholesterol expert system running on a J2ME based emulator.

The Negotiator system has a number of strategies for getting the best hotel room for the user, in terms of price and facilities. If there are a number of hotels willing to offer rooms, the Negotiator will try to get them to reduce the price they are willing to accept. If several rooms are offered at an appropriate price, the Negotiator will attempt to improve the facilities on offer for that price, e.g. breakfast included in the price, or free use of the hotel's business centre. The Negotiator and the Hotel Agents use an 'Iterated Contract Protocol' (Smith and Davis, 1978). The Negotiator agent makes successive 'calls for proposals', (which embody requirements such as price, and room and hotel facilities). The Hotel Agents respond either with offers of rooms, or else with an explanation for why the offer is being rejected (the price requested is too low, a requested facility is unavailable, etc.). Fig. 7 shows the general architecture of the system.

As can be seen in Fig. 7, the hotel room negotiation agent runs on a mobile device such as Smartphone. The Negotiator Agent connects via a wireless network to a central server, whose job is to forward messages between the Negotiator Agent(s) and the Hotel Agents. The Hotel Agents are themselves intelligent software agents operating on behalf of various hotels. Each Hotel Agent has access to a database of available rooms and their facilities. Hotel Agents have their own customisable negotiation strategies; for example, they are more willing to compromise on the price of a hotel room booking for tomorrow than they would be for a booking for next week. The central server forwards room requests to any interested Hotel Agents, and forwards their responses (either offers or explanations for the inability to meet the request) to the Negotiation Agent.

Over a series of iterations of this process, the Negotiation Agent should be able to find a room for the user that has the best combination of price and features currently available.

Figs. 8 and 9 show some screenshots from the current implementation of the Negotiation Agent. At the start of the process (Fig. 8A) the user must specify some details of the room that they are seeking, and their preferred type of hotel. The user must specify the type of room they require ('single', 'double', 'twin', etc.) and a target (maximum) price they are willing to pay per night, as well as the dates of their required stay. In addition, the user must specify the maximum time (in seconds) that they want the Agent to continue negotiating. The constraints of the user specification are relaxed over time, with the Agent willing to accept less favourable rooms closer to the deadline. At the end of this time, they will be presented with details of the best room that the Agent has been able to locate during this time period.

Having specified these details, the user must then choose a 'profile' which is used to try to evaluate each hotel room that is offered (Fig. 8B). The profile contains user-defined weightings of the features and facilities of a hotel and a hotel room, and a function for evaluating the price of the room relative to the user's target price. Essentially, the profile

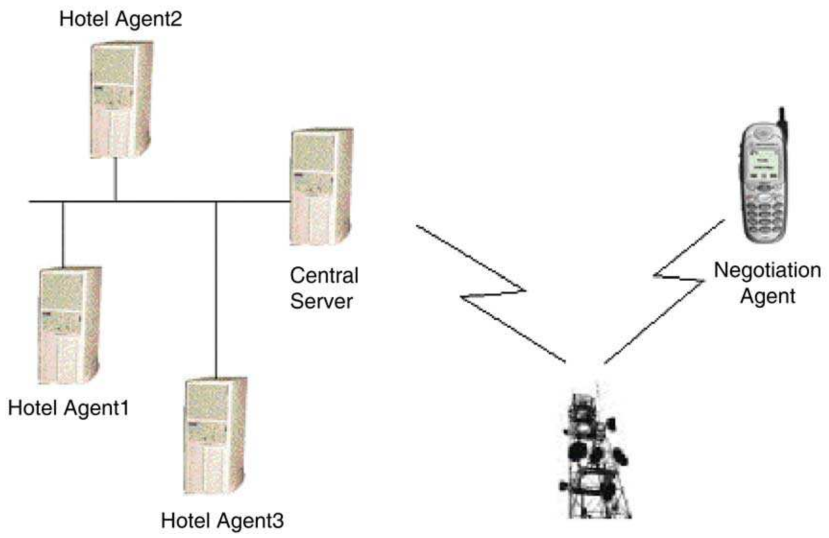
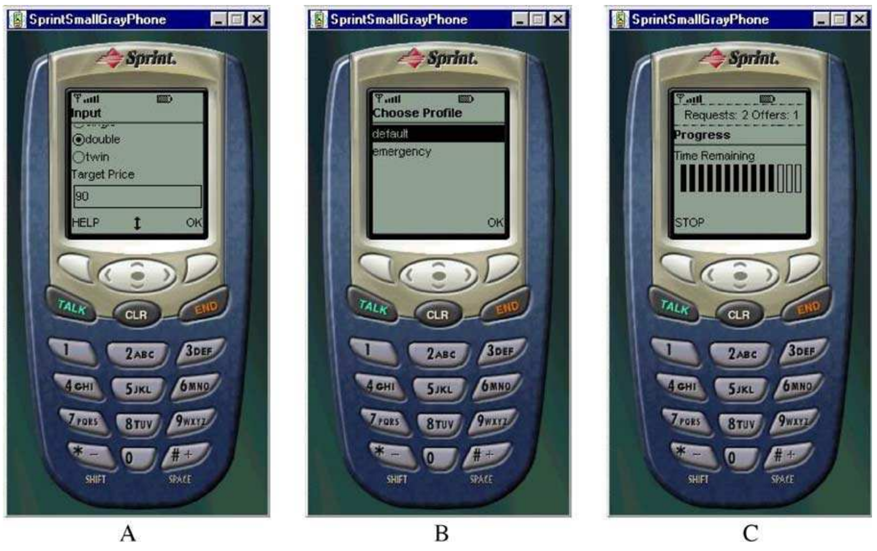


Fig. 7. Architecture of the hotel room negotiation agent system.

provides a means of applying a utility function to evaluate each room offered to the agent, in order for it to be able to evaluate the 'best' room on offer. As shown in Fig. 8B, there are currently two predefined profiles, a 'default' profile, which aims to locate a room which rates facilities and price equally, and an 'emergency' profile which does not rate either price or facilities highly ('Get me a room, any room, fast!'). It would be an easy matter for the user to specify additional profiles (A 'leisure' or 'business' profile, for example), weighting each room/hotel feature appropriately, though this functionality is not currently implemented.

Once the user has specified both the required room details and the profile to use, the Negotiator Agent connects to the central server, and starts to negotiate with the Hotel Agents. As it does so, the user is provided with feedback about the progress of the negotiation (Fig. 8C). A 'ticker' across the top of the screen shows how many room requests the Agent has sent, and how many offers of rooms the Hotel Agents have made. A progress bar shows how much time remains for the Agent to complete negotiations. The user can stop the process at any point, if they are willing to accept the best of the room offers that has already been made.

At the end of the negotiation process, the user is presented with brief details of the best of the room offers that have been received (Fig. 9A). This just details the type of room, the price, and the number of stars for the hotel that the room is in. The user can simply book one of these rooms at this stage (this functionality has not been implemented in the demonstrator application). Otherwise, selecting one of the room offers will show further details of the room offer, features of the room and the hotel, for example (Fig. 9B). Again the user can book the room, or, by choosing one of the items from the menu presented, can download further information about the hotel in question, including an image of the hotel, and a text description of it (Fig. 9C).


Input
[✓ double]
twin
Target Price
90

Choose Profile
[✓ default]
emergency

Requests: 2 Offers: 1
Progress
Time Remaining

Fig. 8. Screenshots form the hotel room negotiator agent application.

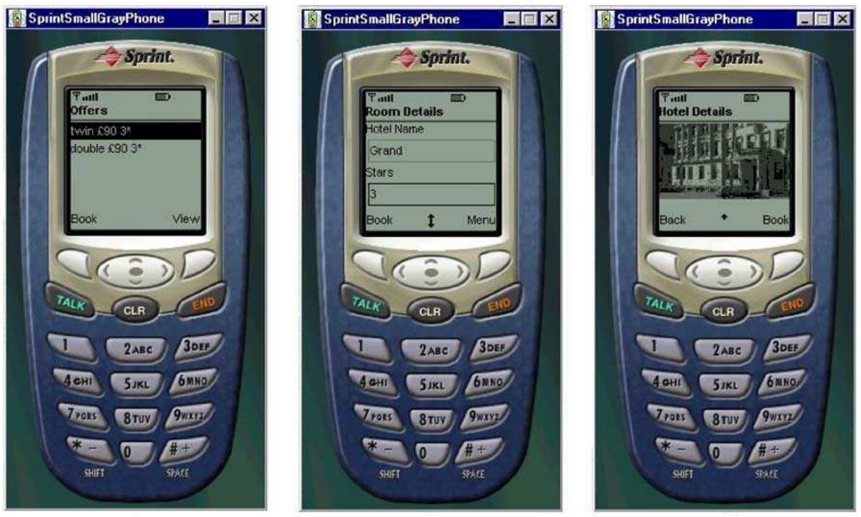


Fig. 9. Further screenshots form the hotel room negotiator agent application.

The above process has been demonstrated using real devices (the Nokia 7650 and Sharp GX10 smartphones) connecting wirelessly to a system which plays the role of the central server/hotel agent configuration shown in Fig. 7. The Hotel Agents (which simulate how real Hotel Agents might act) were created using the same technology as the Negotiation Agent (i.e. MADE), but run on a workstation using the Personal Java version of the MEE. Each connects to its own database of hotel rooms. The central server is implemented using Java Servlets, running under an Apache Tomcat server.

Although location information is not currently used in the demonstration application, it could be added fairly easily. The user could specify whether they would prefer the hotel to be located where they are currently (e.g. the airport), or where they would like to be tomorrow (e.g. the city centre). Location could then be added to the utility function of the system to further evaluate room offers.

5. Discussion and future work

Although some intelligent applications have already been developed for mobile devices, these are typically either limited by platform dependence (Whiteis, McGovern, & Johnston, 2001) or are scaled down versions that offer restricted functionality (O'Hare et al., 2002). The approach that we have described here permits the development and deployment of the same intelligent application (possibly with a slightly modified interface) across a diverse range of devices including PCs, laptops, tablets, PDAs, smart and mobile phones.

The Mimosa programming language enables the development of facts and inference rules. This permits the creation of forward chaining production systems, with expertise such as cholesterol levels as in the expert system or negotiation behaviour in the intelligent agent. Through use of the Rete algorithm, the intelligent application is able to perform efficiently within the restrictions imposed by the mobile devices' processing power.

The intelligent applications are created within the Mimosa Application Development Environment. This environment provides a set of powerful tools that are used to construct the intelligent application. MADE runs on a standard PC and offers the developer a comfortable, appropriate environment for creation, debugging and testing of intelligent applications. Through tailoring the external functions of the application for the relevant device the intelligent application is deliberately kept lean, to ensure that a high level of performance is achieved. However, aside from this tailoring at the user interface level, the same intelligent application created with MADE can be targeted at a range of devices.

This generation of a target device executable is achieved through adding the knowledge base, created in MADE, to the Mimosa Execution Environment (MEE). Each combination of knowledge base and MEE is then tested on the PC with an emulator before being transferred to the mobile device. The intention with our approach is to create 'write once, run anywhere' software for mobile devices. The fragmentation of the mobile device market has required the development of a range of MEE versions in different dialects of Java and C++. This enables the generation of intelligent applications for a significant sector of the mobile device market.

The two sample applications that were discussed in this paper show the considerable potential of intelligent applications for mobile devices. They offer compelling software that supports mobile users with useful tools, illustrating potential in two different intelligent application areas, expert systems and intelligent agents. The examples reveal the potential of intelligent applications in different contexts, for leisure and for work. Our current aim is to develop additional prototype applications that further explore the potential of intelligent applications for mobile devices. A number of applications are currently under development including a diabetes monitor (Newall et al., 2004) and an intelligent application that tutors users in poker (Hall et al., 2004).

The intelligent software agent approach could have numerous other applications in the travel and tourism arena: negotiating for long distance taxi fares, or hire cars, for example. In the work domain, intelligent software agents are being used in an increasing number of other types of application, such as information filtering and diary management applications. Many of these applications would have obvious benefits to users of mobile computing devices. Intelligent applications offer considerable potential for the growing number of mobile workers (Ellsworth, 2002) who are often in situations access to relevant information and expertise can be restricted (Pollock, 2003).

Until recently, the focus of MADE development was on feasibility and functionality, with limited attention paid to the aesthetics of the user interface or to the market potential of the applications. Currently our aim is to develop applications that will achieve some market penetration. Following a similar approach to that for the coronary heart disease expert system, we are in the process of developing a personal intelligent application that gives advice to diabetics about their insulin requirements. The intelligence for this application is being provided by specialists in diabetes and an early prototype is under development.

Additional intelligent applications aimed at both the corporate and consumer sectors are currently under development. Whilst many sectors could benefit from such applications, the increasing number of knowledge industry workers, many of whom are mobile or home-workers, offers a substantial market for intelligent applications. For the consumer sector, there are exploitation possibilities for health, recreational and lifestyle activities.

6. Conclusions

The mobile device market continues to grow, with ever increasing numbers of consumers with diverse backgrounds, professions and expectations. This market offers considerable potential for software producers, however, hitherto few compelling software applications have been developed.

Artificial Intelligence offers a range of proven techniques and technologies that have been used for the development of intelligent applications for the workstation market. Intelligent applications provide compelling software that can readily be seen to have relevance for supporting mobile workers and adding value to the mobile lifestyle. However, there are currently very few intelligent applications for mobile devices.

This lack of intelligent applications partially relates to the constraints imposed on software by the mobile device, with the view that the limitations of memory and processing power would make mobile devices unable to support powerful AI technology. The difficulties of producing intelligent applications for the mobile device market is exacerbated by the diversity of the mobile device market, requiring that any intelligent application would be portable.

The Mimosa approach permits the development of portable intelligent applications for mobile devices and has been used successfully for two diverse applications, a Coronary Heart Disease Expert System and an Intelligent Agent Hotel Negotiator. These case studies reveal the potential of intelligent applications to provide compelling software to the mobile device user.

As AI enters maturity it offers nascent technology such as mobile devices the possibility of shippable applications rather than research lab demonstrators. Through overcoming the constraints of mobile devices, the Mimosa approach enables the ubiquitous deployment of intelligent applications.

References

Ala-Siuru, P., Belkin, N. J., Mayora-Ibarra, O., & Mizzaro, S. (2003). Preface. In *AI moves to IA: Workshop on Artificial Intelligence, Information Access, and Mobile Computing*, Acapulco, Mexico.

Albuquerque, R., Guedes, P., Figueira Filho, C., Robin, J., & Ramalho, G. (2002). Embedding J2ME-based inference engine in handheld devices: The KEOPS case study. *Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices, Italy*.

Albuquerque, R. L., Hubner, J. F. de Paula, G. E., Sichman, J. S., & Ramalho, G. (2001). KSACI: A handheld device infrastructure for agents communication. *ATAL 2001 Conference, Seattle, USA*.

BBC. Mobiles to replace handheld PCs. *BBC News On-line*, 2003.

Bellifemine, F., Poggi A., & Rimassa G. (1999). JADE—A FIPA-compliant agent framework. *Fourth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents (PAAM), London, UK*.

Bergenti, F., & Poggi, A. (2001). LEAP: A FIPA platform for handheld and mobile devices. *ATAL 2001 Conference, Seattle, USA*.

Brachtl, M., Slajs, J., & Slavik, P. (2000). PDA based navigation system for a 3D environment. In *Third International Workshop on Intelligent Interactive Assistance and Mobile Multimedia Computing (IMC'2000)*, Rostock-Warnemünde, Germany.

Brownston, L., Farrell, R., Kant, E., & Martin, N. (1985). *Programming expert systems in ops5: An introduction to rule-based programming*. Reading, MA: Addison-Wesley.

Busetta, P., Ronnquist, R., Hodgson, A., & Lucas, A. (1999). *JACK intelligent agents—components for intelligent agents in Java*. AgentLink Newsletter, 1.

Canalys, (2003). Canalys research release 2003/71. *Canalys*.

Canalys, (2001). Mobile analysis: A choice of one or two devices. *Canalys*.

Collier, R. W., O'Hare, G. M. P. Lowen, T. D., & Rooney, C. F. B. (2003). Beyond prototyping in the factory of agents. *Third International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, Prague, Czech Republic* Springer.

Craig, C. (2002). Compelling apps key to mobile devices' success. *InfoWorld*.

Cripps, T. (2001). Applications on the move. *Computer Business Review On-line*.

Ellsworth, L. (2002). Business on the move: empowering your mobile workers. *Ebiz*.

Evans, B. G., & Baughan, B., Visions of 4G. (2000). *IEE Electronics and Communication Engineering Journal*, 12(6). 293–303.

FIPA. *FIPA specifications*, 2003.

Forgy, C.L. (1982). *Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence*, 19, 17–37.

Friedman-Hill, E. J. (2001). *The Java expert system shell: Version 5.2*. Livermore, CA: Sandia National Laboratories.

Hall, L., Gordon, A., James, R., & Newall, L. (2004). A lightweight rule -based AI engine for mobile games. *Advances in Computer Entertainment*, Singapore, ACM Press.

Hameed, K., (2003). The application of mobile computing and technology to health care services. *Telematics and Informatics*, 20(2), 99–106.

Kacker, A. (2002). New value drivers for the mobile business. *Analysys Resources: Industry Comment*.

Loke, S. W. (2002). Supporting intelligent BDI agents on resource -limited mobile devices-issues and challenges from a mobile database perspective. *Workshop on Ubiquitous Agents on Embedded, Wearable and Mobile Devices at the First International Joint Conference on Autonomous Agents and Multiagent Systems, Italy. (AAMAS 2002)*.

Lomuscio, A., Wooldridge, M., & Jennings, N. (2001). A classification scheme for negotiation in electronic commerce. In F. Dignum, & C. Sierra (Eds.), *Agent-mediated electronic commerce: A European perspective* (pp. 19–33). Springer.

Menzies, T., (2003). 21st Century AI: Proud, Not Smug. *IEEE Intelligent Systems*. 18–25.

Mozelak, B., Glassman, B., & Johnston, R. (2001). Personal digital assistant uses and preferences among clinical oncologists: a needs assessment. *14th IEEE Symposium on Computer-Based Medical Systems* (pp. 529–534). Bethesda, MD: National Library of Medicine.

Nedovic, L., & Devedzic, V. (2002). Expert systems in finance—a cross-section of the field. *Expert Systems With Applications*, 23(1), 49–66.

National Cholesterol Education Program (2002). *Second Report of the Expert Panel on Detection, Evaluation, and Treatment of High Blood Cholesterol in Adults.* Bethsheda, MD, National Heart, Lung and Blood Institute.

Newall, L., Hall, L., Paley, G., James, R., & Gordon, A. (2004). Mobile mentoring for diabetes. *Fifth International Conference on ICT in Education, Samos, Greece.*

Nwana, H., Ndumu, D., Lee, L., & Collis, J. (1999). ZEUS: A tool-kit for building distributed multi-agent systems. *Applied Artificial Intelligence Journal*, 13(1). 129–186.

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. First International Workshop of Central and Eastern Europe on Multi-Agent Systems (CEEMAS'99), St Petersburg, Russia*.

O'Hare, G. M. P., O'Hare, P. T., & Lowen, T. D. Far and A WAY: Context sensitive service delivery through mobile lightweight PDA hosted agents. *15th International Florida Artificial Intelligence (FLAIRS) Conference (Pensacola, Florida).* 2002. AAAI PressM.

Oppermann, R., & Specht, M. Adaptive mobile museum guide for information and learning on demand. *Eighth International Conference on Human–Computer Interaction, Munich.* 1999

Peters, S. (2002). Technology and adoption: Mobile services. *The Teleconomy Newsletter*.

Pollock, W. K. The evolution of mobile/field service management software packages: Why you need one, how to select one and what it needs to do for you. *Strategies for growth, Westtown, PA*. 2003

Rao, A. S., & Georgeff, M. P. BDI agents: From theory to practice. *First International Conference on Multiagent Systems*. 1995.

Reuters. Mobile market heads for new high. *ZDNet UK*, 2003.

Smiley, K. (2003). IT Trends : Mobile and wireless devices. *Giga Information Group*, 2002.

Smith, R. G., & Davis, R. (1978). Distributed problem solving: The contract net approach. *Second National Conference of the Canadian Society for Computational Studies of Intelligence*.

Smith, J. H., & Egenfeldt-Nielsen S. (2003). The Six Myths of Computer Gaming. *Game Research.*

Tate, A. I-X and ⟨I-N-C-A⟩: an architecture and related ontology for mixed-initiative synthesis tasks.

Whiteis, D., et al. (2001). NCIcon. Patent Pending.

Willems, J. L., Abreu-Lima, C., Arnaud, P., van Bemmel, J. H., Brohet, C., Degani, R., Denis, B., Gehring, J., Graham, I., & van Herpen, G. (1991). The diagnostic performance of computer programs for the interpretation of electrocardiograms. *New England Journal of Medicine*, *325*(25), 17–1767.