AGENT MOBILITY IN FIPA-OS

MILLA MÄKELÄINEN

1999 – 2000

NOTTINGHAM TRENT UNIVERSITY
DEPARTMENT OF COMPUTING

Project Report in part fulfillment of the requirements for the degree of Bachelor of Science with Honors in Computing Systems (Artificial Intelligence)

ABSTRACT

Aim of the project is to investigate different agent mobility methods, choosing an implementation method and developing a mobile agent prototype using the FIPA standard. The implementation is based on FIPA-OS open source FIPA agent platform developed by Nortel Networks.

The project investigates three methods of agent mobility: using available solutions IBM Aglets and ObjectSpace Voyager, and using method that exports only agents state to remote location. All the options are studied and their suitability for the project is evaluated and based on this study best method is chosen for implementation.

Agent state moving is selected for the implementation and a prototype is implemented using the rapid application development as the design method. Report to all the stages of the implementation is given, including screenshots of the final GUIs as well as state transition diagrams for the implemented agents. Agents Java source code is given as an appendix. A test plan for testing the agents is devised.

The results of the tests are shown with screenshots. The discussion about the results also includes a debate about the usefulness of mobility compared to agent messaging and the limitations of the implementation are described.

Discussion about the future work justifies the chosen implementation method, and lists some future work that could be done on the subject.

ACKNOWLEDGEMENTS

Many people have contributed to this project, and without their help, I couldn't have done it. In no order of importance:

Many thanks to my fellow student Steven Robertshaw, who very kindly borrowed me his laptop when my own computer broke down.

I'm very grateful to Nortel Networks' FIPA-OS development team for helping and supporting me throughout the project. I'd especially like to thank Robert Hadingham and Philip Buckle for their encouragement, Simon Martin for his enthusiasm and Alan Treadway for his help with providing me some bug fixes and technical help I couldn't have done myself.

Of course, I'd like to thank my supervisor Dr. Taha Osman for steering me into right direction and helping me with the major task of producing this report.

CONTENTS

LIST OF FIGURES AND TABLES

1 INTRODUCTION

> *"An agent is a piece of software capable of acting intelligently on behalf of a user or users in order to accomplish a task."[1]*

1.1 HISTORICAL BACKGROUND

Agents and agent mobility goes hand in hand with distributed computing and the Internet.

1.1.1 INTERNET

The global Internets predecessor was the Advance Research Projects Agency Network (ARPANET) of the US department of defence, started in early sixties. Although ARPANET was a military project, it was formed with emphasis towards research. The objectives of the project were from the beginning to study networked computers and increase computer research through resource sharing – very much the principles the Internet was later founded on.

In the sixties, there was no general standard on how the computers systems worked. It was common that computer labs each used different systems (some of them unique!), and one of the research issues was to come up with a protocol that the computers could use to communicate. In 1969 first Interface Message Processor (IMP – the standard developed by ARPA) was installed at the University of California Los Angeles, making it the first node in

the network. By the end of the year, there were four nodes in the network[1]. In the seventies, numerous computers were added to the network.

ARPANET and other early networks were, however, purpose built and dedicated to closed communities like universities. There was no pressure for the individual networks to be compatible – in addition commercial companies were pursuing their interests as well. US National Science Foundation (NSF) saw the problems arousing from this, and started its own project in 1985 called NSFNET (project had existed in fractions before), they announced that their intent was to serve the entire academic community. For example, NSF would give universities grants for getting a connection to the network, but only if they agreed to let all qualified users in the campus to use the network. They also made the critical decision in 1985 to make TCP/IP mandatory protocol in the NSFNET.

The real growth of the Internet begun in 1989[2] – while it had took 20 years for the number of hosts in NSFNET to reach 100 000, this figure now tripled in one year alone. The same year the first commercial Internet provider started – appropriately named "The World" – and the next year first commercial services were provided in the Internet by Clari-Net, soon followed by big companies like Compuserve and MCI.

In 1989-90 the most used services in the Internet where email, FTP and Telnet, graphical World Wide Web was just a theory in minds of few CERN scientist. One of them was Tim Berners-Lee – often titled as the father of the WWW, now the director of W3C – who developed concept of hypertext (to be later called HTML) that was demonstrated first time in 1990. Next year Berners-Lee introduced first GUI browser (also an editor) called "Nexus", and in 1993 CERN announced that it was not going to patent the idea of WWW

[1] The other nodes were University of California Santa Barbara, Stanford Research Institute and University of Utah.

opening the way to success for the idea. The affect of this revelation can be seen in number of HTTP Web servers around the world (in 40 countries) rises from 50 in the start of the year to over 600 in the end of it. The rise of numbers of Web servers contributes also to the publication of Mosaic[3] that by the end of the year was available for both Unix and Windows.

Media got permanently interested in the Internet, and the phenomena kept on growing in exponential rate – in 1994 more commercial .com sites are registered than universities .edu sites. Many Internet shops open on the same year, and the first genuine Internet company, Netscape – evolved from Mosaic – is launched, and next year while Netscape went public, commercial companies such as Yahoo! and Lycos were born. In 1995 the amount of traffic in HTTP first time succeeded the amount of traffic in FTP. Microsoft launched a competitor for Netscape called Internet Explorer, and although Netscape is still on the lead with its new Netscape Navigator with frames, animated GIFs and Javacript there is for the first time some serious competition.

In 1995 the Open Source movement starts with Apache Group who introduce a high performance web server called Apache that quickly became the most used server in the Internet. The same year Sun Microsystems announced new programming language called Java, and a browser based on it called HotJava. The browser never took on, but the language did, and next year Sun released first Java development kit (JDK), and Netscape integrated Java support into Navigator.

On the other hand, development on the last few years has been business as usual – World Wide Web Consortium (W3C) releases new versions of HTML, and developed

---

[2] Ironically, the same year ARPANET was shut down.

XML and RDF. On the other hand, governments are now restricting the Internet trying to control the information available. In the US, the controversial Communications Decency Act (aimed at cleaning up the Internet) became a law and was later declared against the constitution. Similar acts have been seen around the world, as governments want to govern the amount of information available to its citizens.

1.1.2 JAVA

Sun was developing in early nineties a prototype hand-held device called *7[4], a consumer-computer convergence device that was seen to be a portable home controller (picture from [15]). A version of UNIX (running in less that 1 MB) was ported to the device. *7 had a flash memory and an embedded file system was created that would work on it. The device was envisaged to be able to dynamically take new application libraries – new software modules could be loaded on to *7 on-the-fly without having to install programs (true plug-and-play). This would allow *7 to control all the devices at home; user would only plug in required module for each device.

Originally, C++ was to be used in the project, but significant difficulties aroused because the language couldn't deliver the requirements for the project. Portability and security were among the problems arousing from the language.

---

[3] Mosaic was developed by Marc Andreessen and Eric Bina.

[4] Read as "StarSeven" – *"…named after an 'answer your phone from any extension' feature of the phone system..."* [15] in *7 development teams office.

James Gosling – the originator of Java technology – started working on a language called "Oak" that would enable their team to write the necessary software for the *7. The new language had to offer following features (from [14]):

- *Networked - to communicate data and programs with other devices*
- *Secure - to prevent unauthorised access to sensitive or valuable information*
- *Reliable - so consumer devices will not fail or need to be re-booted sporadically*
- *Platform independent - for binary compatibility between a wide variety of devices*
- *Multithreaded - to easily allows more than one activity to occur simultaneously*
- *Dynamic - to allow loading and unloading of various software without a local hard disk*
- *Small code size - necessary for small-memory, low-cost consumer devices*
- *Simple and familiar - make it easy for programmers by keeping it similar to C/C++*

Oak wasn't supposed to be developed into a full language in the first place, it was supposed to solve problems that C++ had caused to the development of *7. In the end, *7 didn't kick off, but Oak did – with the emerging WWW.

Originally, the browsers only job was to fetch documents from the server and display them to the user – therefore putting strain to the server. In addition, many of the problems that disturbed *7s progress were hindering the progress in the WWW: binary compatibility, operating system compatibility, reliability, and security.

To see whether the problems could be solved Sun developed experimental browser first called WebRunner (and later HotJava). WebRunner was the first browser that allowed applets to be downloaded from the server to be executed in the browser. With

WebRunner, Oak was renamed to Java once introduced in the net in 1995 its popularity has never stopped growing.

Curiously, in the demonstration for *7 (later known as Javas) mascot called Duke was waving its hands – and Duke was actually called an agent, that would perform tasks for the user (picture from [16]). It is no accident, that Java even now is considered to be the most suitable language to program software agents – the features it offers are closely connected to distributed computing, the environment where agents live.

1.1.3 AGENTS

The evolution of agents can be divided into two strands: to the study that has been done before nineties and studies after that.

In the seventies, Carl Hewitt proposed a concept of *"…self contained, interactive and concurrently executing object…"* [17] that he called "actor". This object had an internal state and could communicate with other similar objects. This work was the base for studies done with multi agent systems that was mainly concerned with macro issues. The aim of the studies was to analyse and specify systems containing multiple collaborative agents. This approach gives emphasis to society of agents over individual agents. Later issues researched where theoretical issues like architectural and language problems.

Although there is inevitably some overlap with the issues researched in the nineties, new type of research has clearly emerged. Previously only research was done on macro level, but now new research has been done on broader range of agent types (or classes). This can also be credited to the fact that more and larger companies have started getting interested in agents – also the term 'agent' is being used more and more broadly than

before.

Nowadays almost every piece of software has some 'intelligence' in it – be it mail filtering, adaptive interfaces or help with writing a letter – and especially if the intelligence can be seen as an entity, it's usually called an agent. It has been predicted, that in few years time most of the consumer products (not just software) will have some kind of embedded agents in them.

1.2 AGENT MOBILITY

Mobile agents are software entities that may be dispatched from a client computer and transported to a remote server computer for execution. In a way agent mobility may be viewed as an extension to remote script despatching (like applets). The extension is the extension of the life cycle: applet's life cycle starts when you download it to you computer and ends when you move on to a different page. Mobile agent could have been running already when it's called and might not terminate when you don't need it anymore but migrate back to it's home platform. FIPA (Foundation for Intelligent Physical Agents) specifications "Agent Management Support for Mobility" [2], and the additions in "Nomadic Application Support" [3] specify the framework for supporting software agent mobility using the FIPA platform.

1.2.1 IMPLEMENTATION

Although the concept of a FIPA mobile agent itself is not very new, the formal specification is. FIPA 97 does not comment on it, FIPA 98 describes the basic framework and FIPA 99 adds new concepts. There has been lots of research in the area, but the mobility specification is still very much under development. This projects purpose is to

concentrate on the theoretical side of mobile agents and implement a prototype of a (FIPA compliant) mobile agent. This work will include validating FIPA agent management and solve the issue by testing out new concepts introduced in the draft specification "Nomadic Application Support".

1.3 FIPA

Multi-Agent Systems (MAS) consists of many agents that can combine their abilities to solve problems. Due to the collaborative nature of MAS, agent standards play an important role for commercialisation of agent technology. FIPA (Foundation for Intelligent Physical Agents), a non-profit organisation for producing standards for open agent interfaces, has produced several specifications tackling different aspects of MAS. These specifications don't try to dictate internal architectures of agents or how they should be implemented, but they specify the interfaces necessary to support interoperability between different MAS. Symbols, terms and definitions are represented in appendix "Appendix A: Symbols, terms and definitions in FIPA". FIPA identifies four areas for standardisation:

- **Agent Communication Interface**

  This describes the communication between agents and it supports all interactions between two agents. FIPA has specified an Agent Communication Language (ACL) to support the interface (ACL is based on Knowledge Querying and Communication Language KQML). ACL has five levels of formal semantics:

  1. *Protocol* – defines the structure of the agent dialogue, like fipa-request-protocol.
  2. *Communicative Act (CA)* – defines the type of communication currently

performed, like "request" when an agents is requesting a service from another agent.

3. *Messaging* – defines meta-information of the message, like identity of the sender and receiver.

4. *Content Language* – defines the language (i.e. the grammar) of the content message, like XML.

5. *Ontology* – defines the meaning of terms and concepts used in content expression like meaning of the XML tags.

- **Agent Management**

This describes facilities necessary to support the creation of agents, communication between agents, as well as security and mobility. **FIPA 97 defines the platform to be an infrastructure in which agents can be deployed and where FIPA agents can enter, advertise their services, locate other agents and communicate with agents of other platforms.** It consists of three agents – often called the platform agents – ACC[5], AMS and default DF:

- *Directory Facilitator (DF)* – DF is an agent that provides "yellow pages" services to other agents, where agents can register their services and request information of other agents.

- *Agent Management System (AMS)* – AMS is an agent that provides an agent name service, an index of all the agents currently registered in the platform. AMS makes sure that all the agents have unique names, Agent Global Identifiers

(GUIDs). AMS has supervisory power on the platform and it can create, delete and de-register agents and it oversees the migration of the agents to and from other platforms.

- *Agent Communication Channel (ACC)* – ACC is an agent that routes messages between agent platforms and it must minimally support Internet Inter-Orb Protocol (IIOP).

- **Agent/Software Integration Interface**

  This interface supports the interaction between agents and non-agent software. A concept called "wrapper agent" is introduced where an agent wraps itself on to a piece of software/hardware and acts as an agent representative of that piece.

- **Agent/Human Interaction Interface**

This interface describes how agents can interact with human users providing agents with user models.

1.4 CONTENT OF THE PROJECT

The project consists of six chapters:

1. **Introduction** – this chapter: an introduction to the project.
2. **Limitations**: results of literary survey of agent mobility methods. Due to the newness of the subject, literary survey consisted of information gathered from the Internet. Research introduces three different methods to implement agent mobility: Aglets, Voyager and moving agent state.

[5] FIPA99 specification actually changes the previous specifications and states that the ACC doesn't have to be an agent anymore. In FIPA-OS (version 1.03) however the ACC is still an agent – this is not relevant for this project, since ACC is

3. **New ideas**: focusing the attention to the chosen implementation method, the agent state moving. Proposed timetable for the project.

4. **Software development**: details of the implementation and design methodology (Rapid Application Development). Each state of the implementation phase is explained, the final implementation with more detail with state transition diagrams and screenshots of the GUIs. A detailed test plan is shown.

5. **Results and discussion**: results of the testing, the implementation, and its limitations. Discussion of whether agent mobility is a good idea.

6. **Conclusions and future work**: appropriateness of the chosen mobility method and future work that could be done on the subject.

---

not included in the scenario.

2 LIMITATIONS

There are numerous mobile agent platforms around. When researching these platforms the main purpose was to find platform that could be extended to support the FIPA standard and used with FIPA-OS.

2.1 FIPA MOBILITY

FIPA is interested in two types of mobility: mobility in devices (handled as part of Agent Management specification of FIPA98) and mobility in software such as mobile agents. Mobility support specification specifies *"…the minimum requirements and technologies to allow agents to take advantage of mobility."* and it *"…provides a wrapping mechanism for existing mobile agent systems to promote interoperability."* [2]. Therefore the specification doesn't instruct the actual use of mobility features but mandates how agent platforms can support mobility. There isn't an obligatory technology for use, or a definition of how mobile agents operate or how they are implemented.

FIPA recognises the many ways of agent mobility (code and state mobility, agent migration, cloning…), so a core set of actions are defined that allow flexible forms of mobility protocols to be supported. Mobility support specification defines two types of mobility protocols; **Simple Mobility Protocols** and **Full Mobility Protocols**. Different protocols are meant to be used in different situations, therefore not making the other better than the other. If FIPA platform supports mobility, it's expected that either or both protocols will be implemented. When agent sends the first *move* action to remote platform, it specifies what protocol it wants to use.

2.1.1 SIMPLE MOBILITY PROTOCOLS

Simple mobility protocol is a high level protocol where an agent uses a single action that causes the agent to be moved to destination agent platform. Agent delegates the operation to both of the platforms therefore moving the complexity of the move to the platforms from the agent itself. This allows developers to use existing agent platforms via FIPA ACL (Agent Communication Language) enhancement.

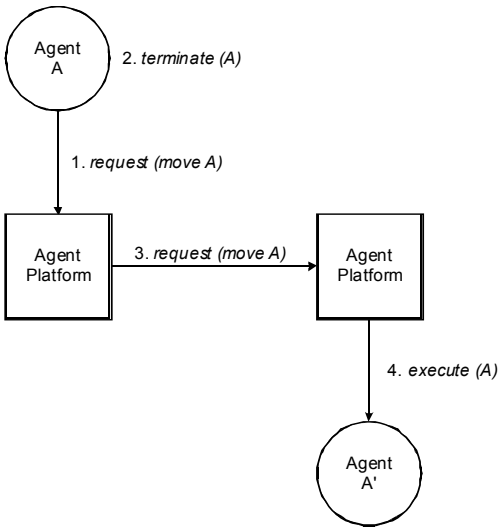


Figure 2.1: Simple Mobility Protocol [2]

2.1.2 FULL MOBILITY PROTOCOLS

Full mobility protocol is low level protocol where agent takes the control of the move without delegating any duties to the platforms involved – only possible contact is information of the move. *"…an agent first moves its agent code (and possibly state) to a destination AP and eventually transfers its identity and authority once it is assured that the new agent has been created successfully."* [2] The move is not considered complete before the code and the identity of the agent have successfully been transferred. In this way agent platform complexity is reduced while increasing the capabilities of the application agent. This protocol

enhances the security of the agent when most of the messaging in simple mobility protocol becomes unnecessary.

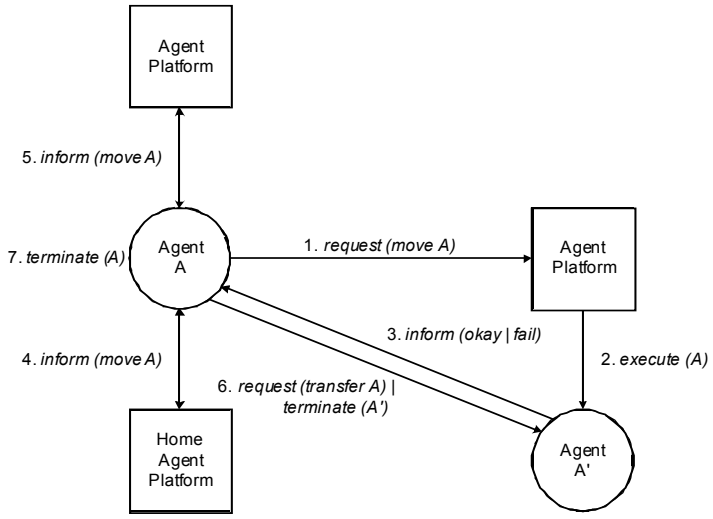


Figure 2.2 : Full Mobility Protocol [2]

There are four FIPA actions that relate to agent mobility, and they are presented in Table 2.1. Two first ones, move and transfer are actions for the mobile agent, two latter ones are actions for AMS and are used to manage mobile agents.

Action	Description	Performative
move	An agent issues a move request to transfer itself to a local/remote AMS. AMS may refuse to accept the move request due to lack of agent profile support or other local restrictions. When an agent applies to move to a local/remote AMS, a mobile agent description must be supplied containing values for all of the mandatory attributes of the mobile agent description.	request
transfer	An agent issues a transfer request to send its identity and authority to another agent on a destination AMS. Receiving agent may refuse to accept the transfer request for security reasons. When an agent applies to transfer its identity and authority to another agent on a destination AMS a mobile agent description must be supplied containing values for all of the mandatory attributes of the agent identity description.	request

Action	Description	Performative
move-state	An agent issues a move-state request to transfer its internal state (its instance data) to a local/remote AMS. AMS may refuse to accept the move-state request due to lack of agent profile support or other local restrictions. When an agent applies to move-state to a local/remote AMS, a mobile agent description must be supplied containing values for all of the mandatory attributes of the mobile agent description.	request
move-codebase	An agent issues a move-codebase request to transfer its Code Base (its class code) to a local/remote AMS. AMS may refuse to accept the move-codebase request due to lack of agent profile support or other local restrictions. When an agent applies to move-codebase to a local/remote AMS, a mobile agent description must be supplied containing values for all of the mandatory attributes of the mobile agent description.	request
load-codebase	Load a codebase (class of an object) into the specified AMS.	platform specific action: no specific performative
execute	Executes an agent on an AP.	platform specific action: no specific performative
terminate	Terminates the execution of an agent on an AP.	platform specific action: no specific performative

Table 2.1: Actions supporting mobility [2],[3]

Because agents can be transported to agent platforms, they can therefore make demands for a set of conditions to be met. These conditions could be related to certain operating system to be running or particular software to be present. These requirements are expressed as part of meta-information of the mobile agent called `:agent-profile`, `whole ontology is defined in` "Appendix B: Mobility Ontology".

2.1.3 AGENT LIFECYCLE

FIPA agent lifecycle defines that the agent can be in three different states in its lifetime (represented in the AMS): active, waiting and suspended. Mobility support specification defines one more state to support the mobility (transit), and two actions to enter and leave

the state (move and execute).

2.2 FIPA-OS

FIPA-OS is an open source implementation of a FIPA agent platform developed by Nortel Networks. FIPA-OS is implemented in Java, and uses Common Object Request Broker Architecture (CORBA). It has the potential to support mobile agents (as a FIPA platform), but mobility hasn't been implemented. The FIPA reference model shown in Figure 2.3 illustrates the core components of the FIPA-OS distribution: the platform and the message transport service. The agent reference model provides the normative framework within which FIPA agents exist and operate. Agents by default communicate with other platforms agent via the ACC, but FIPA doesn't stop agents communicating directly with each other.

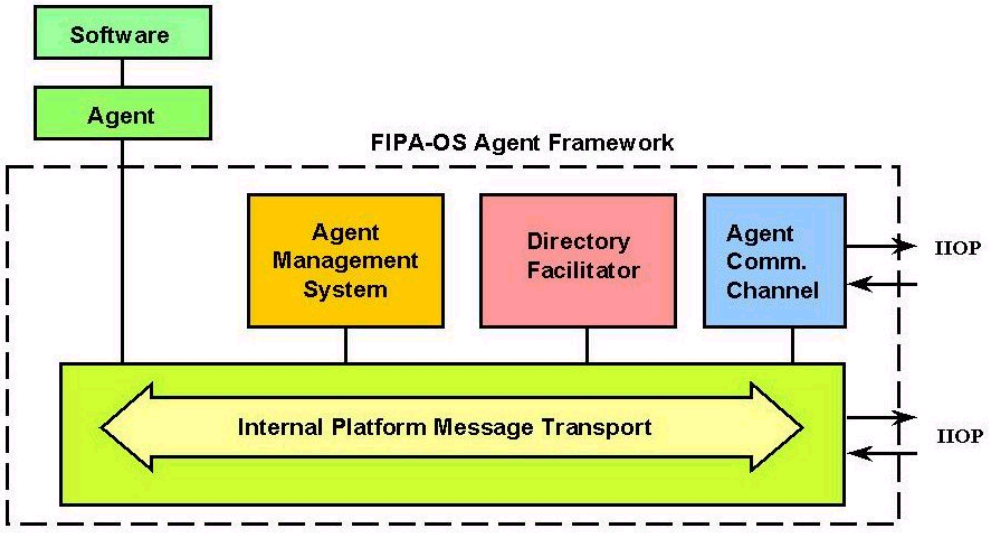


Figure 2.3 : FIPA reference model [4]

In FIPA-OS, agents are constructed on top of this framework – an agent shell that includes the message transport, but leaves rest of the implementation to the developer.

Both agent and transport interfaces are abstract, so both could be implemented independently, but there is no need for that in this project – ready implementations that come with FIPA-OS (transport alternatives are RMI, SunIDL and Voyager) are quite suitable.

FIPA calls series of messages in an order dictated by a protocol a **conversation**. Conversations between agents can get extremely complicated, and because of this implementing a system just dealing with incoming individual ACL messages will be difficult. In FIPA-OS there has been implemented a conversation manager, that deals with whole conversations instead of individual ACL messages. Conversations are stored in two lists: conversations-in-progress and completed-conversations, and each time an ACL message arrives (either from the agent, or from message transport channel), conversation manager (CM) inspects it. If the message starts a conversation, new conversation object is added into conversations-in-progress list, otherwise new message is added into an old conversation. If this message completes a conversation, this conversation object is moved into completed conversations. Operation is illustrated in Figure 2.4.

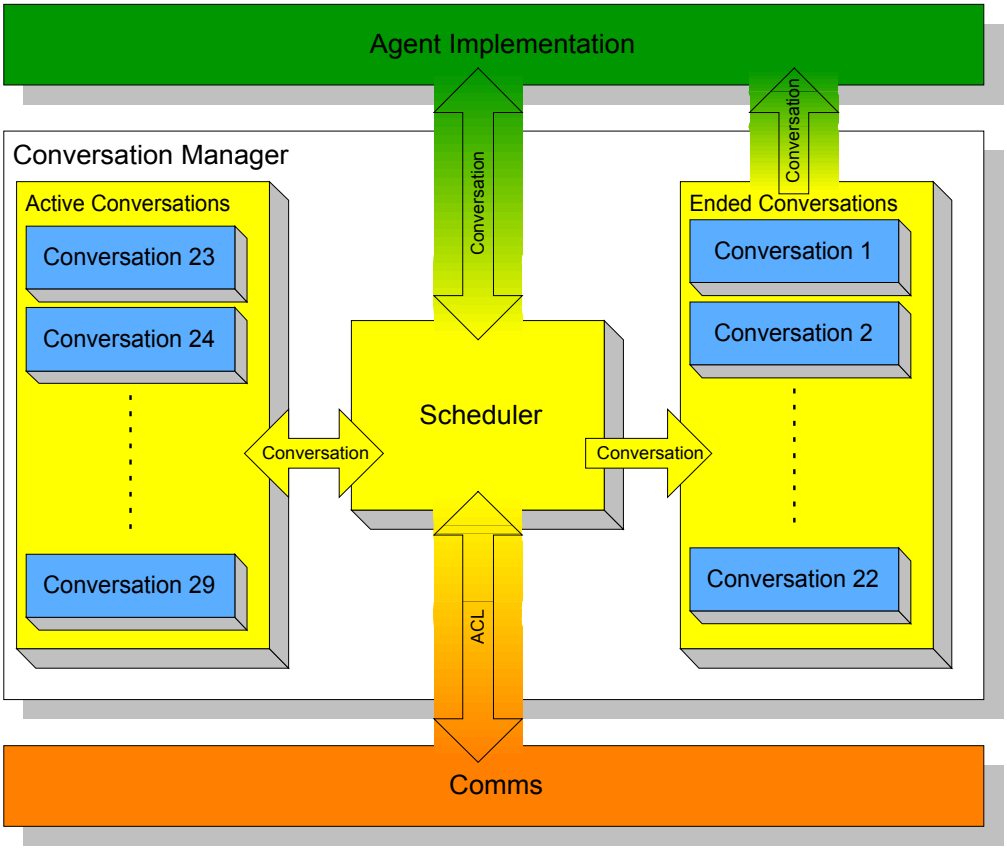


Figure 2.4: Conversation Manager [13]

Next we take a look at three different methods for implementing mobility, and evaluate their possible uses for implementing FIPA mobility with FIPA-OS.

2.3 IBM AGLETS

Aglets are IBM's version of mobile agents. They are *"…Java objects that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, dispatch to a remote host, and resume execution there. When the aglet moves, it brings along its program code as well as its state (data). A build-in security mechanism makes it safe to host untrusted aglets."* [5]

Aglets' roots are in applets and servlets[6]: when aglet moves, it brings with it everything it needs to execute – including the program code and possible runtime data. IBM has intended the aglets to be the industry standard in mobile agents, they should be secure, easy to program and be platform independent (meaning Java Aglet Application Programming Interface platform independent).

2.3.1 AGLET OBJECT MODEL

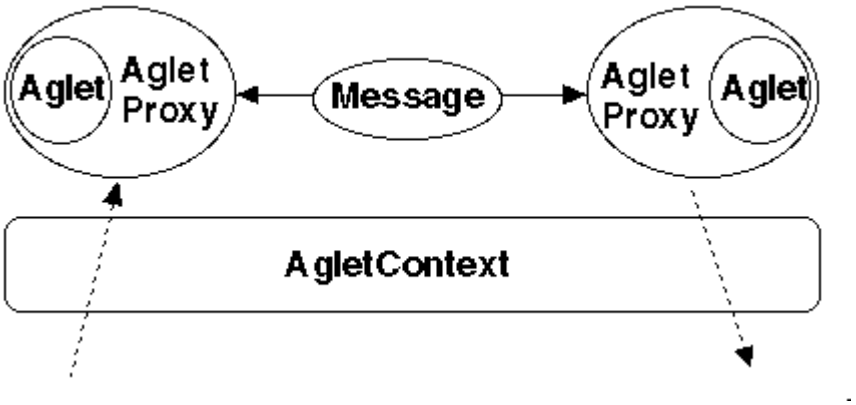


Figure 2.5: Aglet Object Model [6]

Aglet object model consists of abstract classes and interfaces that define necessary behaviour for mobile agents and messaging. These abstractions are Aglet, AgletIdentifier, AgletProxy, Itinerary, Message, AgletContext, FutureReply and MessageManager. Two most important concepts are Aglet and Aglet Context:

- *AgletContext* – this is where the aglets workspace where they live, it's a base platform that provides a uniform execution environment for the aglets. One computer can have multiple contexts, but a context can't be distributed across many computers. Aglet creation happens in the context – *"The new aglet is assigned an*

[6] *"Servlets are modules that extend request/response-oriented servers, such as Java-enabled web servers. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a company's order database. Servlets*

identifier [AgletIdentifier]*, inserted in the context, and initialised. The aglet starts executing as soon as it has successfully been initialised."* [5] Aglet can also be cloned – then a new copy (identical in every aspect except identifier and execution threads) is initialised in the same context. When agent moves it's dispatched from it's current context and it's inserted into the destination context where it will start execution (threads won't migrate). Moving the agent can be either done by pushing or pulling, depending on whether context wants to despatch or retract the aglet.

- *Aglet – "An aglet is a mobile Java object that visits aglet-enabled hosts in a computer network. It is autonomous, since it runs in its own thread of execution after arriving at a host, and reactive, because of its ability to respond to incoming messages."* [5]. Aglets can also have Proxies that act as a shield to both protect aglet from the outside world (by not allowing public access to it's methods), but it also provides a uniform look of the world to the aglet by hiding all location information. Aglets can be deactivated after creation (aglet will be removed from the context) and activated again (returning the aglet to the context). Aglets can send messages (Java objects) to each other – MessageManager provides concurrency control for incoming messages. Messages can be either synchronous or asynchronous and aglets can collaborate with messaging.

---

are to servers what applets are to browsers. Unlike applets, however, servlets have no graphical user interface." [18]

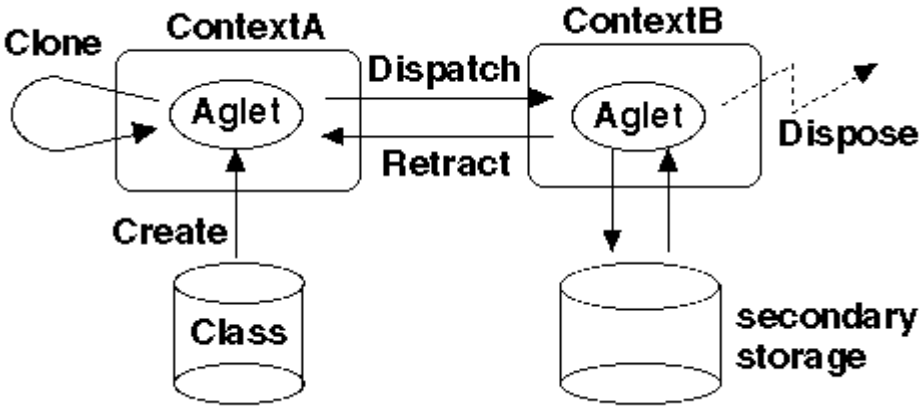


Figure 2.6: Aglet Life Cycle [6]

Because security is essential with agent mobility – accepting a hostile agent may lead to computer system being damaged – aglets try to give an answer this problem as well. IBM's security will be provided by the aglet host: aglets have been divided into trusted and untrusted aglets (host's decision). Aglet security manager checks every time aglets attempts to access the file system, network, or other aglets whether it's trusted or not.

2.3.2 AGLETS' USEFULNESS

It seems that despite all IBM's efforts, aglets didn't turn out to be the standard of mobile agents. New updates for aglet development kit seem to come far and between, the research is mainly done in Japan. Aglets use sockets to move between contexts, which limits it's capabilities to communicate with other agents. When considering that one of the main arguments for agents is the collaboration between each other, this restricts aglets to aglets and makes interoperability difficult – FIPA uses CORBA. Because aglets are treated very differently to "normal" Java objects, it's not possible to remotely create aglets – they have to be created locally and then dispatched. Stationary agents aglets can send messages,

but message handling is left to the aglet itself! There is no distributed naming service that would allow giving aglets aliases and therefore being universally able to send them messages after they have moved (when moved, the referenced aglet become stale).

Aglets are not well suited for implementing FIPA agents: the whole idea of aglets is to be mobile, whereas FIPA agents don't need to be. To be able to use FIPA-OS, the whole platform would have to be changed; all the agents would need to be aglets.

2.4 OBJECTSPACE VOYAGER

Voyager is a family of products for distributed computing that includes a free of charge Object Request Broker (ORB). The family includes Voyager ORB, Voyager ORB Professional, Voyager Security, Voyager Transactions and Voyager Application Server. All the other products except the standard ORB are commercial, so only that product was investigated.

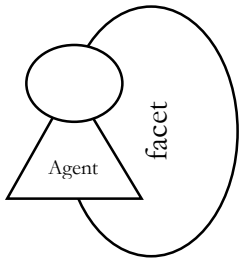
"Voyager ORB includes a universal naming service, universal directory, activation framework, task management framework, resource policy management, publish-subscribe, and mobile agent technology." [7] Voyager ORB supports simultaneously all the major ORB standards: CORBA, RMI and DCOM – this enables interoperability between different kinds of applications. This also doesn't force the developer to decide permanently on what standard to use, since it can be changed without changing the software, or even (to a degree) the code. ORB allows programs to communicate remotely (Java, for example, didn't use to have a remote messaging at all before RMI) with objects called **proxies** – with Voyager ORB proxies can be generated dynamically. Universal communications make it possible for the programs to be both server and a client by supporting concurrent messaging using CORBA, RMI,

COM and Voyager's own VRMP (Voyager Remote Method Protocol). With universal messaging different types of messages (synchronous, one-way, future) can be sent to an object independently of its location of object model. Universal directory service allows a single directory to be used by all the clients, and therefore allowing same objects to be accessed by clients supporting different ORB standard.

To enable use of Voyager ORB in the existing code is simple: Java classes don't have to be modified to enable remote access. Voyager also supports dynamic class loading from multiple locations so that it doesn't (need to) import the code when migrating.

2.4.1 MOBILE AGENTS WITH VOYAGER

"Voyager supports dynamic aggregation, which allows you to attach new code and data to an object at runtime." [8] This allows developer to attach code to (maybe third party) components and extending the functionality on run-time. These secondary objects are also called **facets** and it's possible to attach more than one facet into a primary object. Facets don't have to be related to the primary object in any way, although it's not possible to attach facets into facets.



By attaching a facet into an agent, it's possible to make an agent mobile without modifying the existing code. When moving an object, several things will happen:

1. If object is processing concurrent messages, they will be allowed to complete while receiving of new messages is suspended.
2. The object is copied to the new location using Java serialisation.

3. New address of the object is cashed in the old location and the old object is destroyed.
4. All the suspended messages are sent to the new location.
5. When new messages come to the old location, it's rebound to the new address and sent on.

"The rules for garbage collection are not affected by mobility. A moved object is reclaimed when there are no more local or remote references to it. The new addresses cached at the old location are not treated as references by the garbage collection system." [8]

2.4.2 VOYAGER'S USEFULNESS

Voyager is designed to work with third party software, so in theory it could be ideal "add-on" to FIPA-OS[7]. However, incorporating Voyager isn't straightforward, since *"It is unsafe to move an object when local Java references point to it from outside the context of Voyager or when the object has one or more threads not associated with a remote message."* [8]. This will make the task more challenging: agents in FIPA-OS are very much rooted in the communication channel making it very difficult to severe all the connections.

As it will turn out in chapter 4, Voyager is not suitable for implementing mobility with FIPA-OS, because of the message transport system. When agent is started up, it is bound to message transport system, so that it's ready to push and pull messages to and from the message queue, and new conversation management system is created to manage the conversations. All these references outside the context of Voyager make it impossible to move the agent once it's created. Only possible way to move the agent would be to shut it

[7] In fact, Voyager is already in use in FIPA-OS as a message transport option, although this is not associated with mobility.

down first, and that would mean loosing it's instance data and therefore nullify the whole idea of moving the code-base.

2.5 ALTERNATIVE APRROACH: MOVING AGENT STATE

Aglets and Voyager support actual code mobility – it means that they actually move the whole instance of the agent while it's running. However, this is not the only way to implement agent mobility. The other possibility is to import just the **agent state**, the instance data, all the information that the agent holds at a particular time. In situations where the hosts are extremely small (like mobile phones), moving the execution code is not even feasible. If moving agents state, the migrating agent would just pass the state to the remote platform and shut down. The remote platform would then take this data and instantiate a new agent with this data. The resulting agent would practically be identical to agent that's code would have been migrated.

Moving the agent state would require platform to be running in the destination host – but the same requirements apply to the aglets and Voyager systems as well. The difference is in the volume of the traffic – only one string message would be required to move the state whereas moving the code-base requires a lot more bandwidth. This can also mean that the migration is faster, since sending (string) messages is so much faster than moving big pieces of byte code.

FIPAs nomadic application support specification [3] suggests an approach for agent state transfer. By using *move-state* action, agent could ask the AMS to move its state to another platform. If the code-base of the agent exists in the destination platform, the originating platform can send the state, and the destination platform will then put this state

into the code-base and activate the agent.

This can be extended even further – the destination platform may have ways of searching the code-base either from a local storage (possible even a storage agent?), or it could even search the network. This could be done using different protocols like HTTP, FTP… Figure 2.7 shows the agent migration protocol with code uploading: it works otherwise as before, but the destination platform loads up the code base from (in this case a local) database before accepting the request.

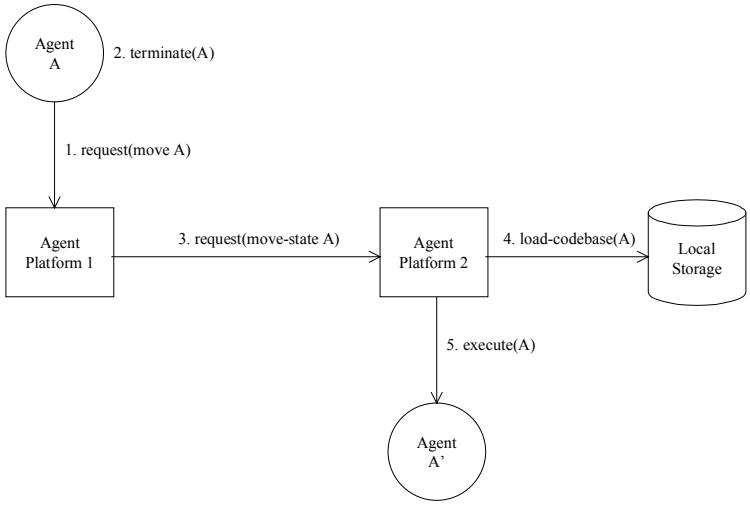


Figure 2.7: Agent Migration with Uploading Code-base Protocol [3]

This approach requires minimal changes to be made to the existing platform and agents. Agents' functionality will remain the same; only addition is the gathering of the instance data, passing it on to the AMS and adding a new constructor so that new agents can be started with ready instance data. AMS will have to be extended to dynamically load up required classes, but this can be viewed as part of AMS' responsibilities anyway.

FIPA suggests, that both moving the code-base and state could be implemented in the

platform. This way, the agent would start by asking the move of the state and if the code wasn't available, it could ask code mobility. In the end, the agent would be moved – in one way or another. From the agent point of view, the method is irrelevant.

3 NEW IDEAS

This chapter discusses changing location in context of FIPA. While FIPA considers agents to be mobile only when they change platform, this project investigates mobility inside the platform and how FIPA mobility protocols could be applied to it. A new concept of Mobility Management System is introduced, a suggested delegate agent to Agent Management System. Further, a change to content language from SL to XML in mobile agent management is established. Last, a detailed plan and timetable for the project is devised. Symbols and terms used are introduced in Appendix A: Symbols, terms and definitions in FIPA.

3.1 CHANGING LOCATION

FIPA defines an agent to be mobile only if the agent moves between two different platforms. This essentially means that the agent will give its ownership from its current platform to the destination platform. However, FIPA platforms can be distributed across different computers, even over the Internet, and therefore potentially any networked computer can belong to any platform.

How about a situation where agent just wants to relocate without changing a platform? There are many reasons why an agent might want to do this; here are just few examples:

- Resources. Agent might lack resources in its current location, and elsewhere in the platform are more locations where the needed resources are available. Resources could be memory, disk space, processor power, etc. There is no reason why an agent should give its ownership to another platform because of this. Even more, it might be crucial for the agent to be in its current location: for example think about

the Directory Facilitator, it can't leave its home platform.

- Task. Agents task might involve moving from place to place doing its task, it might be installing software on the network or maintaining the system, like scanning it for viruses. Tasks like these might be very important for the platform (or the network AP was maintaining), and stopping them might be fatal. For instance, the AMS (usually owner of the agent) has the authority to destroy an agent from its platform handing over that power to a remote platform will not allow that.

Because of this it is suggested, that there should be two kinds of mobility: mobility inside a platform and mobility between two platforms. Since FIPA defines *location* to be associated with the AP, for inside platform mobility to be introduced we need a concept of *place*.

Place could best be described as an address: a *hostname* or *IP address* with *port number*. This would introduce dependence to network software to be present – but on the other hand, if computer's not in the network, what's the point of mobility?

Does the concept of place need to be added to FIPA? FIPA only defines an agent platform that provides an infrastructure in which agents can be deployed. *"FIPA is concerned only with how communication is carried out between agents who are native to the AP and agents outside the AP, or agents who dynamically register with an AP."* [9] Isn't mobility part of APs internal functionality, nothing to do with anything else? It could be argued that mobility inside the platform could be viewed as part of the physical infrastructure and consequently part of the platform.

3.2 MOBILITY BETWEEN TWO PLATFORMS

Mobility between two platforms is the traditional kind of mobility that FIPA offers. In this scenario agent asks remote platforms permission to move there and if agent's request is accepted, the move is executed (either by the platform or by the agent itself).

3.3 MOBILITY INSIDE THE PLATFORM

Mobility inside the platform is new kind of mobility suggested in this report that's meant to extend existing FIPA mobility. Extensions to the existing ontology (in Appendix B: Mobility Ontology) are minimal, since only difference is the fact that the agent/AMS won't contact other platform. The protocols will also remain the similar:

- Simple Mobility Protocols: agent will ask to be moved, and AMS will move it to a different location. Difference is that the AMS won't ask remote AP to move the agent: it will do it by itself – compare **Error! Reference source not found.** and Figure 2.1.

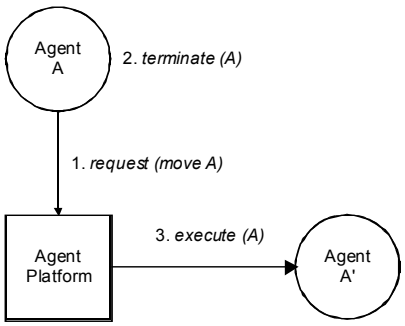


Figure 2.1: Simple Mobility Protocols in Inside Platform Mobility

- Full Mobility Protocols: agent will ask the permission to be moved, and when permission is given agent will move itself to the destination platform. The protocol is virtually the same since destination and home platforms are the same, compare **Error! Reference source not found.** and Figure 2.2

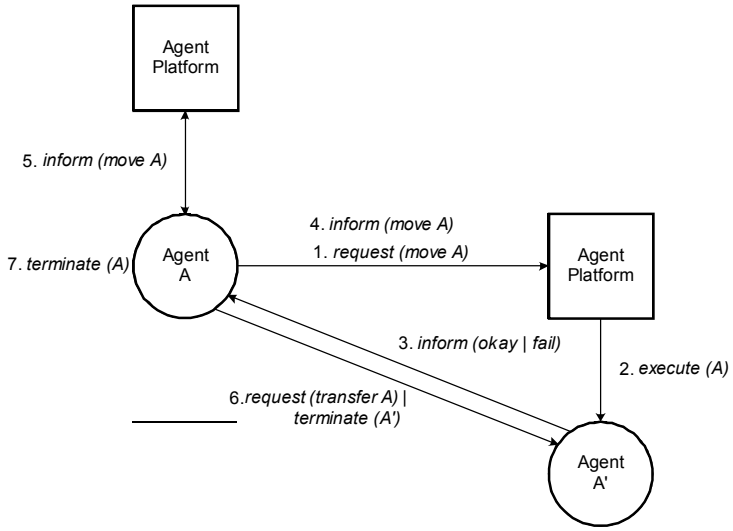


Figure 2.2 : Full Mobility Protocols in Inside Platform Mobility

FIPA states, that the implementation of either of these protocols will provide mobility to the platform, so it's not necessary to implement both of them. Implementations could potentially be very different, so they don't need to correspond to each other in any way. In this project the mobility inside the platform is implemented, but as later is discussed the extensions to implement mobility between two platforms will be minor.

3.4 MOBILITY MANAGEMENT SYSTEM

The agent that supervises mobility (and everything else) in FIPA is Agent Management System (AMS). However, AMS is large and complicated agent, and constantly under construction – extending it would be bigger task than how much there is time in the scope of the project. Therefore, the agent in place of the AMS that is to be implemented is Mobility Management System (MMS).

MMS has the AMS mobility extensions defined in mobility support specification [2],

but none of the capabilities of AMS itself. The functionality of MMS will be following:

- Starting up agents: since MMS is responsible for starting up agents, it will need the facility to dynamically load them up.
- *move-state*[8] action: when MMS receives a valid move request, it will move the requesting agent state to the destination location and shut down the old one. MMS will put the agent into *transit* state and therefore all communication with the agent is put on hold.
- *execute* action: when MMS has completed the move, it will put the agent into *active* state and re-establish all communication. This will be implemented as a "request" message from the MMS.

MMS can also be viewed AMS' delegate agent – then AMS can be unchanged, but it can have delegate MMS' who will handle mobility. If there are no MMS' on the platform, mobility is not supported. This provides flexible approach to providing mobility on the platform: small platforms or scenarios that don't require mobility will still have AMS, but without MMS and bigger platforms and scenarios needing mobility can have (possibly multiple) MMS'.

3.5 PREVIOUS WORK

There is currently no other published work on mobile FIPA agents. Although there is no doubt, that many FIPA members have implemented mobility on their platforms, most members are commercial companies and therefore can't easily publish their results.

[8] Specified in the nomadic application support specification [3] – used in the demonstrator instead of vague *move* action in mobility support specification [2].

The previous work that this project relies is therefore only the FIPA-OS FIPA platform. FIPA is open source code, so it's easy to modify existing code when needed and existing code gives a good base for new code to be developed for the platform.

3.6 XML AS CONTENT LANGUAGE

Agents exchange information using ACL messages, and the most important part in the message is the content. FIPA mobile agent management ontology is described in Appendix B: Mobility Ontology. This ontology however is coded in SL[9], and to be able to parse messages in SL requires writing a parser. However writing a parser like this would take far too much time compared to time available to this project.

Solution is to use Extensible Markup Language (XML) as a content language. XML (like HTML) is a subset of SGML and is designed to be used in the Internet. W3C endorses the current specification (1.0 [10]) and it has already become a standard that many companies are implementing parsers for it. The specification *"…describes a class of data objects called XML documents and partially describes the behaviour of computer programs which process them."* [10].

Since the specification describes the behaviour of parsers as well, it has been possible to develop a **Simple API for XML (SAX)**. SAX has been developed by W3Cs XML-DEV mailing list and it's an event driven mechanism for parsing XML – practically all XML parsers extend it. This allows developers to write handlers that react to XML events, and then use any SAX extending parser to parse it.

[9] Semantic Language

3.7 DEVELOPMENT TIMETABLE

The project development can be divided into three separate parts: research, coding and write-up. Due to familiarity of the subject, coding and write-up will take more time than research, since most of the fundamental research (like researching the FIPA standard) was done during the placement. Table 2.1 shows the approximated times to be used in this project for each part.

Task	1999						2000		
	July	August	September	October	November	December	January	February	March
Research	3	3	10	10	4		10	10	
Write-up			5	5	5		5	30	20
Code			10	10	10	10	10	20	10
Total	Research: 50 + write-up: 70 + code: 80 = 200 hours								

Table 2.1: Project Timetable

More detailed division of the tasks into subtasks is in Table 2.2.

Task	Subtask	Hours
Research	What kind of methods and packages are available to support Agent Mobility? The research includes finding out history of mobile agents, what kind of mobile agents are out there, and what of these might be useful in this project. Criteria used to pick a suitable method include implementation language (Java), available documentation and frequency of the updates for the product. Main research channel is the internet.	15
	Concentrated research on subjects that seem to be most relevant. From previous subtask, few methods/products are chosen and more research is done on them. Research includes investigating relevant parts of FIPA specifications as well as reading independent reviews of the chosen product. Main research channel is the internet.	25
	How these methods could be used in this project. Maybe biggest part of the preceding subtask is to find out which one of the methods would be chosen for the implementation. The main selection criteria will be how the new technology can be integrated into existing, i.e. FIPA-OS.	10
	Chapter 1: Introduction	5
	Chapter 2: Limitations	12
	Chapter 3: New Ideas	15
	Chapter 4: Software Development	8
	Chapter 5: Results	7

Task	Subtask	Hours
	Chapter 6: Conclusions	9
	Layouts	9
	Final typing	5
Coding	Design. Includes communication patterns, messages sent and the model of the platform and of the agents. Main development method will be Rapid Application Development (RAD) so design will be changing many times before the finish.	20
	MMS. Biggest work will include parsing the messages and dynamic class loading.	20
	Search Agent.	15
	Testing. As well as ongoing testing – testing each part while developing the code – planning of the overall final testing.	25
	total	200

Table 2.2: Breakdown of Tasks

To avoid time running out in the end, coding has been distributed evenly for the time available, this allows more time to be left to write-up in the later stage. Also the implementation method (rapid application development) guarantees that there will be some functional code in the end.

4 SOFTWARE DEVELOPMENT

This chapter explains the software development process in the project. The development method used in the process is Rapid Application Development, so each of the stages of the development is explained. Final version of the implementation is described with more detail including screenshots and state transition diagrams of the agents.

The software used in the development process, and the 3rd party components of the implementation and their versions are listed, and coding standards used are described. The implemented scenario is outlined before the explanation of the code development.

4.1 DEVELOPMENT METHODOLOGY: RAPID APPLICATION DEVELOPMENT

Rapid Application Methodology (RAD) is a methodology for compressing analysis, design and test phases into a series of short iterative development cycles. Traditional sequential development cycle follows very formal steps completing every step at a time and producing formal assessments at finishing each step – the final product is delivered in the end. RAD iteration produces functional version of the final system at the end of each cycle giving the developer opportunities to reconsider and improve the design.

RAD was chosen as the design method, because it was seen as the most useful methodology for developing a working prototype in a short time. Other design methods produce prototypes after many hours of analysis and design, and when producing something as volatile as this project there is very little time for mistakes.

FIPA defines ACL (Agent Communication Language) that specifies communication between agents and has an associated formal semantics. Semantic consists of five levels,

but only two of them are central in this context: protocol ("social rules" for structuring the dialogue) and communicative act (defines the type of communication performed).

"Agent interaction protocols define a sequence of messages which represent a complete conversation or dialogue between agents." [1]. FIPA defines several protocols, but this assignment will only use one of them: fipa-request protocol. FIPA defines also numerous communicative acts (CA), but fipa-request protocol uses only six of them. These CAs are: request, not-understood, refuse, agree, failure and inform. Figure 4.1 illustrates the protocol which consists of an ordered exchange of CAs between two agents initiated by a request CA. Agent sending the request now knows that the response will be either "not-understood" (receiving agent doesn't understand the message), refuse (agent doesn't want to co-operate) or agree (agent has agreed to co-operate).

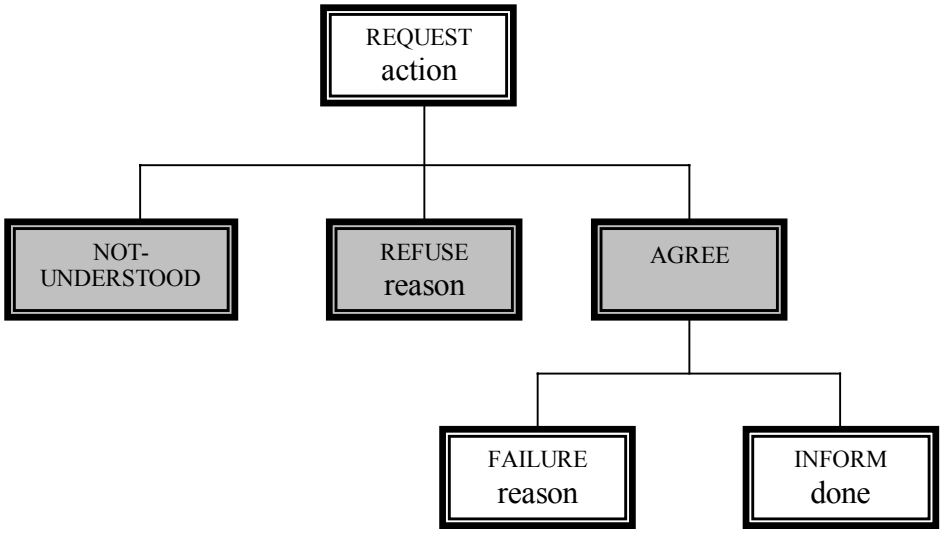


Figure 4.1: FIPA-REQUEST protocol

4.2 SOFTWARE

Version of FIPA-OS used is 1.03. The entire FIPA-OS code base has been migrated

from the previous version to use Java 1.2. This FIPA-OS build has been created using Java 1.2.2 JDK, so that's the version of Java used to develop the code.

FIPA-OS content language and agent profile parsers currently support XML\RDF encoding, extra third party software is needed to run the platform. XML parser from IBM, XML4J (version 1.1.16) – this includes SAX (version 1.0). To parser RDF, RDF parser SiRPAC is needed (version 1.14).

To develop the Java code, development tool called Kawa was used – it's an integrated development environment (IDE) used to build Java applications on win32 environment [11].

4.3 CODING STANDARDS

There are many reasons why adopting coding standards is a good idea. Especially, long lasting software is almost never maintained by its original author, and *"Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly."* [12]. Coding in this project uses mostly Suns standards for Java code [12], except few differences described in Table 4.1.

Standard	Example	Explanation

Open brace "{" appears at the start of next line of the declaration statement.	`public void method()`
	`{`
	`}`
		Improves readability, makes it more clear where a start of a block is.

Blank space should appear on both sides of statements and variables in brackets.	`print( "Hello " + name );`	Improves readability.
If variable name consists of more than one word, words should be separated by underscores "_".	`int big_number = 1000;`	Improves readability, makes it easier to differentiate between variables and methods.

Global variables and should start with underscore "_" and static variables should start with double underscore "__". Local variables should never start with underscore. This does not apply to constants.	`int _number = 1;`
	`static int __snumber = 10;`
	
	…
	`int local_number = 1;`
		Improves readability, with one look reader can tell whether variable is local or global.

Table 4.1: Coding Standard Exceptions

4.4 SCENARIO

Although the focus of this project in on the principle of mobility and not applications that could be built on top of it, the prototype implemented requires some kind of functionality. Since usually agents move to location because of its resources, the implementation was designed to work on local hard disk.

The implemented scenario the user wants to find files from computers in the network, but wants to search local hard disks that are not shared and therefore not visible. User can then send the mobile agent to the location and agent can do the search and then return with the results.

4.5 CODE DEVELOPMENT

FIPA defines the protocols used in mobility scenario, so the communication model was throughout the development constant. The model dictates the behaviour of the agents. Figure 4.2 illustrates the messages sent, and the actions related them.

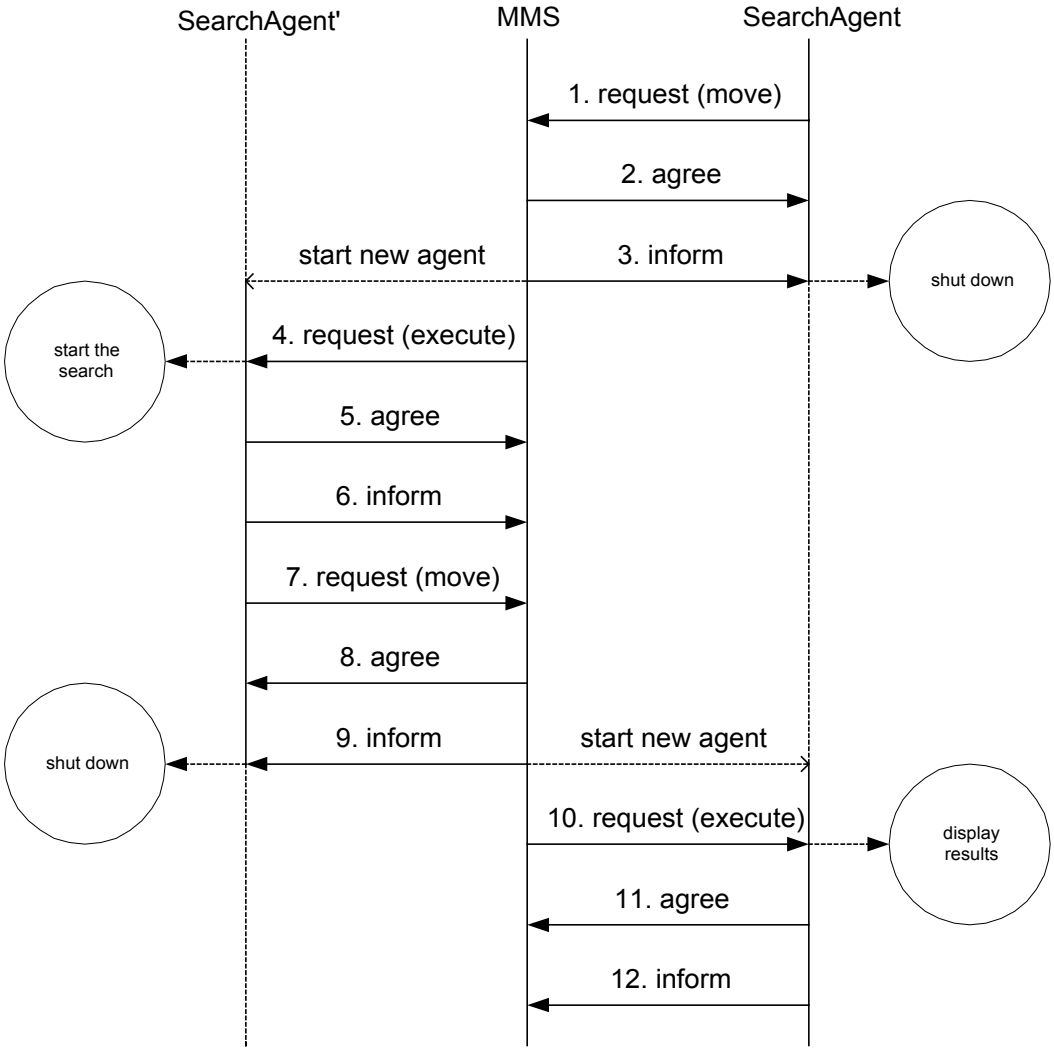


Figure 4.2: Communication and Action Model

Three models were developed to reach a working and final model. Next, these models are introduced in detail.

4.5.1 FIRST IMPLEMENTATION

The first model comprised of three agents, two GUIs, and initial helper and container classes.

In the first model, user would start up the MMS that would then call "Starting GUI".

Starting GUI gives the user a possibility to first give details for the search (directory, search criteria), and will after the search show the results. User could also decide whether to start a mobile or a stationary agent. If stationary agent was chosen, SearchAgent was started, and if mobile agent was chosen, MobileSearchAgent was started. MMS would then wait for request for move.

SearchAgent would perform the search, by using helper class SearchFilter (implements java.io.FilenameFilter) and send the result to MMS by an ACL message.

Before MobileSearchAgent (extended SearchAgent) would start, a "Mobile Agent Detail GUI" would start prompting the user for mobile agent details (from "Appendix B: Mobility Ontology"). These details would be stored in contained classes FipaMobileAgentDescription, FipaMobileAgentProfile, FipaMobileAgentSystem, FipaMobileAgentLanguage and FipaMobileAgentOs, that reflect the ontology hierarchy. In the first model the agent wouldn't actually do anything but send the request for move – and the MMS wouldn't do anything else but reply.

Working in this implementation was really only the stationary SearchAgent, and even though the first model was quite far away from actual working mobile agent system, it did contain many elements that would remain unchanged till the final working model. The helper class SearchFilter was in its original form, and container classes changed very little till the final version. The GUIs were pretty much in their final stages – all of this code made a solid base for the future implementations.

4.5.2 SECOND IMPLEMENTATION

The second implementation consisted of two agents with integrated GUIs, and the

helper and container classes.

In the second version, user would start up the MMS – but this time the starting GUI was integrated into the MMS. MMS is the only agent that ever calls that GUI, so there was no reason to keep it as a separate class. Otherwise the GUI looked and performed as the previous version: asks the user details for the search (directory, search criteria), and after the search show the results. User could decide whether to start a mobile or a stationary agent. MMS would then wait for request for move at this stage.

Second time around both search agents have been put into one. It was argued that SearchAgent was in fact nothing else than stationary MobileSearchAgent. Due to this the GUI was integrated into the agent itself – it no longer needed to be separate since it wasn't being used by anything else. The agent now implemented com.objectspace.voyager.mobility.IMobile interface since it the aim to use Voyager for mobility.

Stationary SearchAgent would behave exactly as before. Mobile SearchAgent would ask user the details and send the move request to MMS as before. The intention was, that agent would start the GUI on pre-arrival (IMobility method that fires the first thing when the object is moved to destination location) and would start search post-arrival (IMobility method that fires when move is completed).

The attempt to use Voyager was unsuccessful, and still the only thing working in this implementation was the stationary SearchAgent. It turned out that moving objects with Voyager is not safe if the object has references to it from outside Voyager – this is of course the case in FIPA-OS. FIPA-OS message transport channel has references to agents, and that prevents the use of Voyager. Different things were tried out to conquer this

problem, like trying to severe all the connections to message transport, but the possible solutions quickly started to look like answers to a completely different question. One possible solution was to shut down the agent and store all the instance data – but that is the method for moving agent state… This led to the final implementation.

4.6 FINAL IMPLEMENTATION

In the final implementation, there are two agents, one integrated GUI and one separate GUI (without an agent).

4.6.1 FUNCTIONALITY

In the final version, user would start up the separate Starting GUI and MMS that doesn't have a GUI at all. GUI looked and performed as the previous version (Figure 4.3): asks the user details for the search (directory, search criteria), but doesn't display the result – that is left for the agent. User can decide whether to start a mobile or a stationary agent. Mobile agent detail GUI looked also like before, now it displays default values that the user can use if he or she wants to. Default address will be the platforms address (from the platform profile), and so on. Mobile agent detail GUI is shown in Figure 4.4.

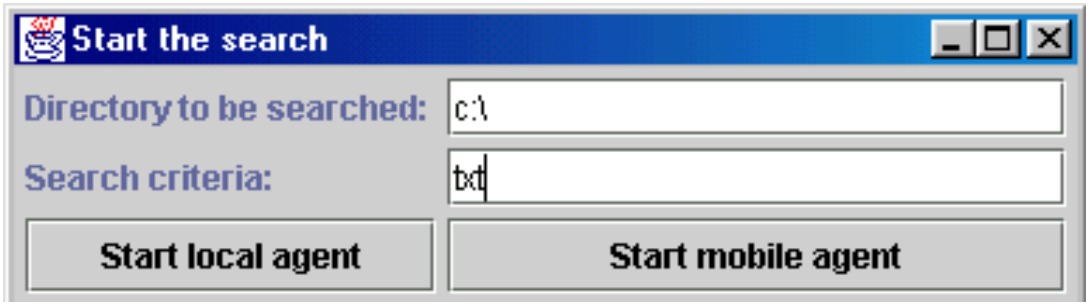


Figure 4.3: Screenshot of mobility.gui.StartingGUI

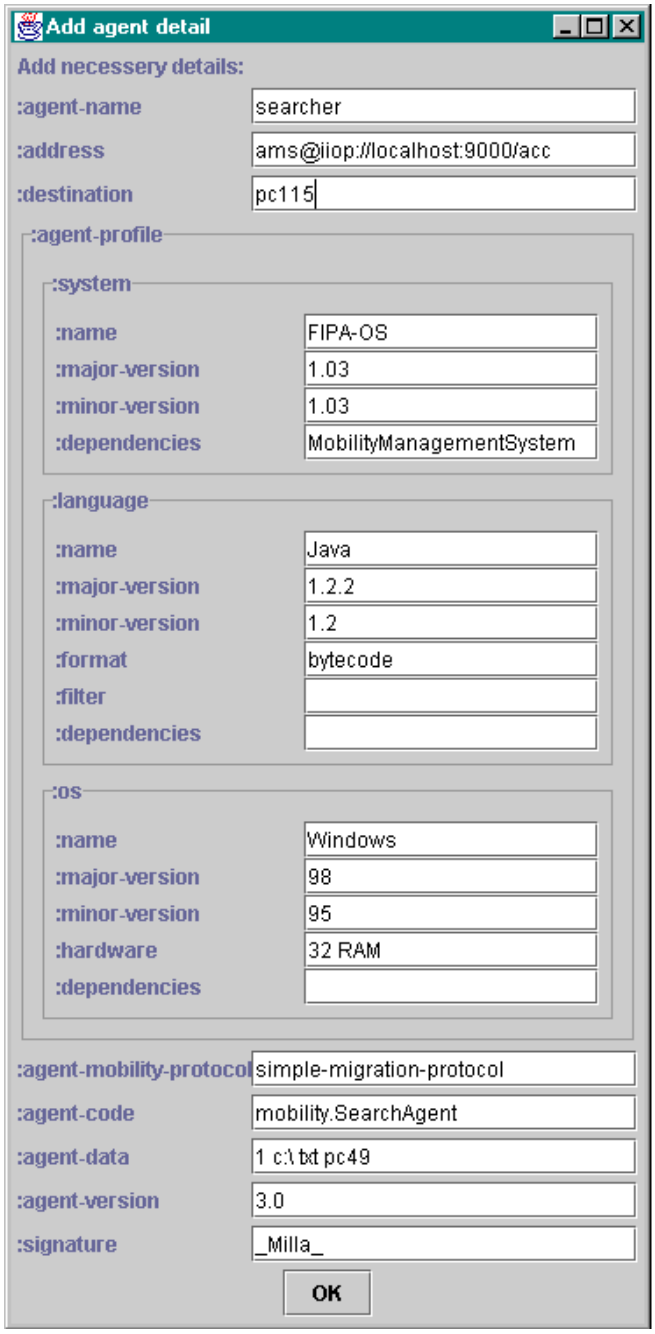


Figure 4.4: Screenshot of Mobile Agent Detail GUI

MMS behaves very much as before – state transition diagram can be seen in Figure 4.5.

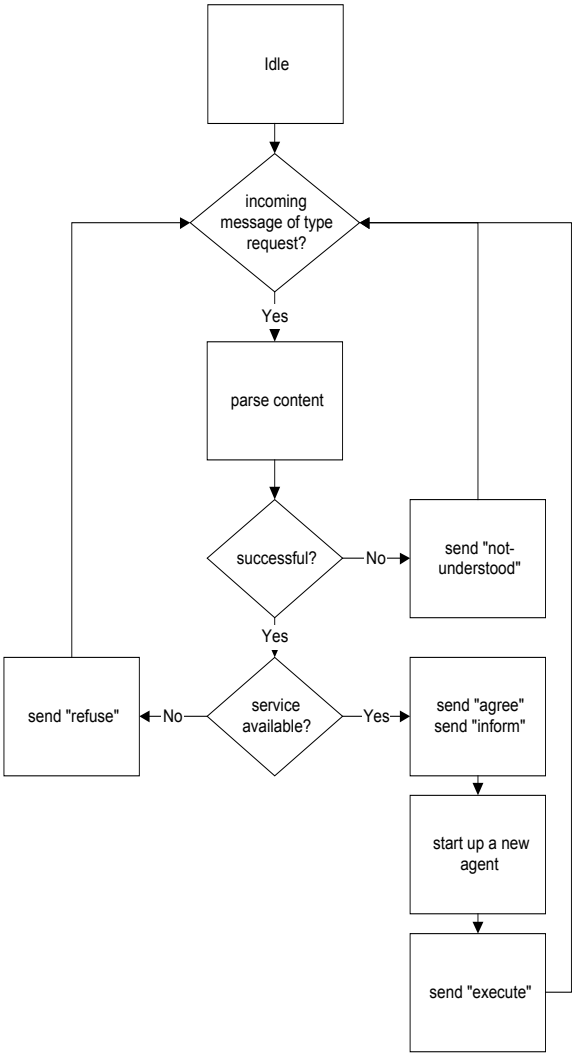


Figure 4.5: MMS State Transition Diagram

The previous versions had worked with default values – then MMS always assumed that if the agent wanted to be moved it would be a default location. This is of course not the case in reality – message that the agent sends must be parsed to be able to interpret it. Mobility ontology is defined in SL – and there are no generic parsers for SL (apart from parsers that tell that a string is syntactically correct). There is no time within the scope of the project to write a parser – but there is no reason why the ontology language couldn't be changed into language that would have generic parsers.

XML was chosen as the content language, because of authors familiarity with it, and because there are free generic parsers that can be used to parse XML documents. Here is an example of an XML document that are used in the scenario:

```
<?xml version="1.0"?>
<!DOCTYPE action SYSTEM "description.dtd">
<action to="mms">
    <move-state>
        <fipa-mobile-agent-description>
                <agent-name>searcher</agent-name>
                <address>iiop://holly:9000/acc</address>
                <destination>iiop://holly:10000/acc</destination>
                <agent-profile>
                        <system>
                        <name>FIPA-OS</name>
                        <major-version>1.03</major-version>
                        <minor-version>1.03</minor-version>
                        <dependencies>MobilityManagementSystem</dependencies>
                        </system>
                        <language>
                        <name>Java</name>
                        <minor-version>1.2</minor-version>
                        <major-version>1.2.2</major-version>
                        <format>bytecode</format>
                        </language>
                        <os>
                        <name>Windows</name>
                        <major-version>98</major-version>
                        <minor-version>95</minor-version>
                        <hardware>32 RAM</hardware>
                        </os>
                </agent-profile>
                <agent-mobility-protocol>simple-migration-protocol
                </agent-mobility-protocol>
                <agent-code>mobility.SearchAgent</agent-code>
                <agent-data>1 c:\ milla vaio</agent-data>
                <agent-version>3.0</agent-version>
                <signature>_Milla_</signature>
```

```
        </fipa-mobile-agent-description>
    </move-state>
</action>
```

Mobility ontology's SL coding has been changed to XML very simply by changing SLs ":keyword data" format into XMLs "<keyword>data</keyword>" format. Furthermore, agent management "action" keyword with "move-state" (normally coded in SL) is added to the document so that it contains all the same information as the old version.

MMS will compare content message to the values that it reads from its profile. Chapter 4.6.3 explains in more detail how agent profiles were used.

The stationary SearchAgent works as before – when agent is started, it performs the search, displays results and shuts down. When agent is stationary and if it receives ACL messages, it will always send "not-understood" message back – since it doesn't need to have any communication with any other agents, it doesn't understand them.

State transition diagram for mobile SearchAgent is in Figure 4.6 on page 47. Mobile SearchAgent is started three times in its lifetime, so the state management is very important. When agent starts up and receives message its behaviour is dictated according to what state it is in. Following states are used:

0: STATIONARY – agent is not mobile, doesn't send messages and sends "not-understood" messages if it receives any. If agent is stationary, it will spend its entire lifecycle in state 0.

1: START – agent is mobile, and has started for the first time. When started, it

will immediately send a request to move. Then, when "inform" message with "move" content arrives it will move to state 2 and shut down.

2: MOVED – agent is mobile, and has just arrived to the destination. When started, it will wait for "execute" message, then perform the search and then send a request for move. When "inform" message with "move" content arrives it will move to state 3 and shut down.

3: FINISHING – agent is mobile and has just arrived back to original location. When started, it will wait for "execute" message, then display the results and shut down.

In a real situation, the agent would always call AMS to request the move. In this scenario implemented, AMS is left out and the agent deals directly with the MMS'. The setting needs two MMS', and the agent needs to know somehow what MMS to call at what time. Hard coding this to the agent would be inconvenient; because that would either depend on the user to know what to name the MMS', or the agent would have to compiled again every time before running it to update the information. Either of the alternatives is not practical.

FIPA-OS requires agents to have agent profiles; RDF documents that define agent specific information that the user can configure without having to hard code anything. The platform itself has a platform profile that contains among other things name of the AMS – name of the platform – and default protocols used in communication.

Profiles are meant to be used for information like this – information that the user can easily configure and maintain. In agent profile, two MMS' are given names, where the agent

can read them when it needs to send messages. Chapter 4.6.3 explains in more detail how agent profiles were used.

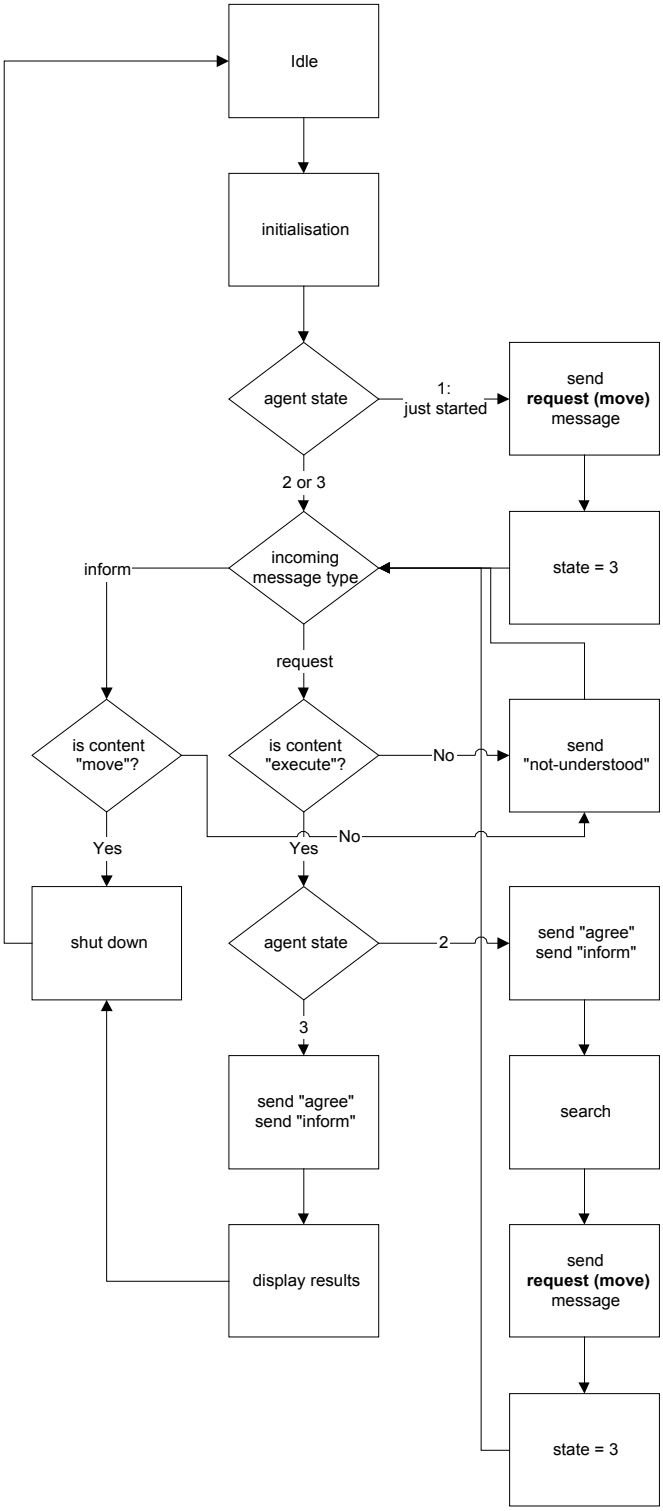


Figure 4.6 Mobile SearchAgent State Transition Diagram

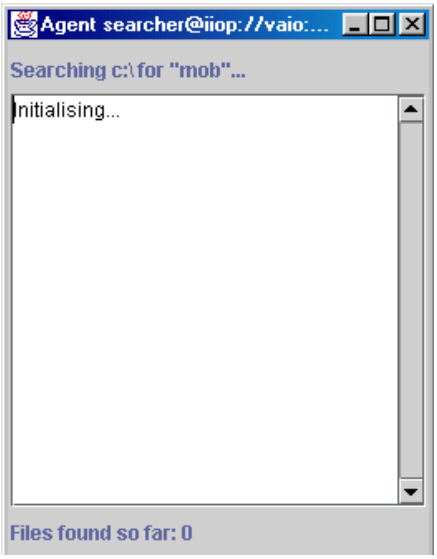


Figure 4.7: Screenshot of SearchAgents Search GUI

Screenshots of SearchAgents GUIs – search GUI and results GUI – are shown in Figure 4.7 and Figure 4.8.

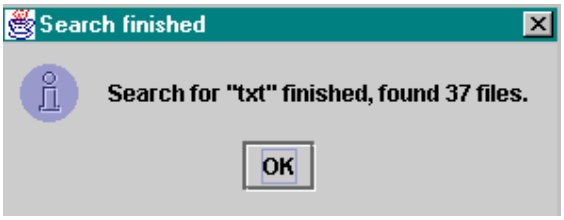


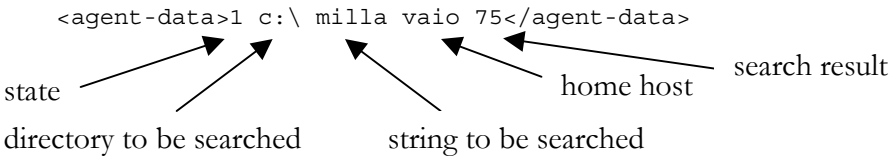
Figure 4.8: Search Finished GUI

4.6.2 MOVING STATE

The most important issue is how the actual move happens – how the state is transferred. State is transferred in the ACL message in a string format, in the "agent-data" field of the mobile agent description.

In this prototype, the instance data is just a string that contains the information separated by spaces (" "). In a small application like this – MMS has concentrated on

moving just this one agent – it is feasible to just rely on the order of the information when parsing it. There are five pieces of instance data that the agent needs:



State is needed, so that agent knows what it was doing when it is started up – agent is started up three times during its lifecycle. Search criteria is needed so that the agent knows what the user wants to search – in this scenario user is only interested in how many files are found, not what or where. Home host name is also needed so that the agent knows where to ask to be moved the second time when it's returning where it started. Search result is needed to complete the task: notify the user of the outcome. State and search result are the only variable in the data, the search criteria and the home host are constants throughout.

4.6.3 USE OF AGENT PROFILES

FIPA-OS supports the use of agent profiles, and the idea that every agent has its own freely configurable profile. However, the code support is mainly there for the default profiles and although there is some code ready to support non-standard profiles – there isn't any documentation.

Some changes were required to the FIPA-OS code to get the non-standard profiles working; also one new content object was added to aw.ont.profile package to store the new information. The changes made are listed in Table 4.2.

Class	Explanation	Change

Class	Explanation	Change
aw.agent.Profile	Profile is a helper class containing the profile information of an agent. It parses the RDF document and populates the appropriate content object.	Profile was storing the non-standard data into a hashtable, the type of the object as a key. This meant that there could be only one object of each type in the hashtable – not very practical solution. A change was made so that the objects were stored in the hashtable by their name – id – so that more than one object of the same type could be stored.
aw.agent.AgentProfile	AgentProfile is the "main" profile class; agents use it to create their profiles. The constructor takes in the location of the platform profile and the name of the agent, parses the profiles and populates the content objects. It uses methods in the Profile class to achieve this.	Hashtable containing the non-standard content objects is visible to the AgentProfile, but they're not visible outside it. This means that the agent has no way of accessing them. A get method was added to AgentProfile so the hashtable could be accessed from outside the class.

Table 4.2: Changes Made to FIPA-OS

All the content classes must implement the ProfileObejct interface from package aw.ont.profile, and must be packaged to the same package themselves. Classes must implement two methods, populate – parse the content – and getName – returns the id of the object. Addition to this the class should contain and give access methods to the information it contains, in this case the object is called "MMS" and the information contained is called "name", it contains the name of the MMS in question. Following is an example of the MMS information in the profile SearchAgent uses:

```
<!-- MMS1 - the home MMS -->
<sa:MMS rdf:about="mms1">
    <sa:name>mms1</sa:name>
</sa:MMS>

<!-- MMS2 - the destination MMS -->
<sa:MMS rdf:about="mms2">
    <sa:name>mms2</sa:name>
</sa:MMS>
```

The "sa" namespace used refers in the tags refers to SearchAgent schema (not actually used to validate the schema), that can be found in "Appendix D: SearchAgent RDF Schema".

MMS uses its agent profile as well. The mobility service it provides in coded in its profile, like system and language information. Here is an example of mobile agent description data MMS profile has:

```
<mad:MobileAgentDescription rdf:about="description">
    <mad:agent-mobility-protocol>simple-migration-protocol
     </mad:agent-mobility-protocol>

    <mad:system-name>FIPA-OS</mad:system-name>
    <mad:system-major-version>1.03</mad:system-major-version>
    <mad:system-minor-version>1.03</mad:system-minor-version>
    <mad:system-dependencies>MobilityManagementSystem
     </mad:system-dependencies>

    <mad:language-name>Java</mad:language-name>
    <mad:language-major-version>1.2.2</mad:language-major-version>
    <mad:language-minor-version>1.2</mad:language-minor-version>
    <mad:language-format>bytecode</mad:language-format>

    <mad:os-hardware>Java</mad:os-hardware>
</mad:MobileAgentDescription>
```

Full profile can be seen in "Appendix I: Agent Profiles" and the RDF schema defining the "mad" namespace is in "Appendix E: MobileAgentDescription RDF Schema".

4.6.4 CLASS HIERARCHY

The packages and the classes in the implementation are in Table 4.3. The Java code for these classes can be found in "Appendix H: Java Code".

Package	Classes	Explanation
mobility	MobilityManagementSystem SearchAgent	The main package that contains both agents at the root.
mobility.util	SearchFilter	Utility class – the search filter.
mobility.parser.xml	MobileAgentSystemHandler	Parser is in it's own package, in the future it's possible that there can be many parsers
mobility.ont.fipaman	FipaMobileAgentDescription FipaMobileAgentProfile FipaMobileAgentSystem FipaMobileAgentLanguage FipaMobileAgentOs	Ontologies are in their own packages; fipaman package is reserved for management ontologies, which mobile agent management ontology is.
mobility.gui	StartingGUI	Graphical components.
aw.ont.profile	MMS MobileAgentDescription	Java container object based on an RDF object.

Table 4.3: Class Hierarchy

4.7 TEST PLAN

The software being a prototype doesn't aim to be completely fault tolerant. However, the agents are expected to function within the FIPA-REQUEST protocol. This means that once request message is received, agents should answer either of these three:

- NOT-UNDERSTOOD: if the agent can't parse the content of the message, or if SearchAgent while waiting for the "execute" receives anything else.
- REFUSE: if the agent receives a request to do something it can't – specifically, if MMS receives a request to move with service it can't provide (for example wrong type of mobility).
- AGREE: if everything is fine and the agent is ready to accept the request. This is followed by INFORM.

These three answers define the test cases – since there are three levels of success. If the agents don't crash – even if the return message is not agree – the scenario has been partially

successful since agents behave like FIPA agents.

The first case is the "not-understood" message, the MMS' failure to understand the message received. Details on how to test this case is in Table 4.4 – in this case only one message is needed because MMS will only send "not-understood" when the parse doesn't succeed.

Expected result: NOT-UNDERSTOOD		
Sender	**Case**	**Test Message**
iotestagent[10]	1.0	(request :sender iotestagent@iiop://localhost:9000/acc :receiver mms2@iiop://localhost:9000/acc :content (testing) :protocol fipa-request :ontology fipa-mobile-agent-management :conversation-id test-123)

Table 4.4: Test Message for Testing NOT-UNDERSTOOD

Second case is the "refuse" message. MMS sends this message when it can parse the message, but can't mach the services required by the requesting agent. Typically agent would be trying to ask a move to different location to where the MMS is located, or would require a different mobility system to operate (like Voyager). Details of the test cases are in Table 4.5 – for saving space only relevant part of the content of the message is shown (everything else is correct by default).

Expected result: REFUSE		
Sender	**Case**	**Test Message**
SearchAgent	2.0	<system> <name>Voyager</name> </system>
	2.1	<system> <name>FIPA-OS</name> <major-version>1.0.1</major-version> <minor-version>1.0.1</minor-version> </system>

[10] SearchAgent can't send syntactically incorrect messages – that feature is hard coded. Instead, FIPA-OS' iotestagent that is used to send any given ACL messages is used.

Expected result: REFUSE		
Sender	**Case**	**Test Message**
	2.2	<system> <name>FIPA-OS</name> <major-version>1.0.3</major-version> <minor-version>1.0.3/minor-version> <dependencies>Voyager</dependencies> </system>
	2.3	<language> <name>C++</name> </language>
	2.4	<language> <name>Java</name> <minor-version>1.1.7</minor-version> <major-version>1.1.8</major-version> <format>bytecode</format> </language>
	2.5	<language> <name>Java</name> <minor-version>1.2</minor-version> <major-version>1.2.2</major-version> <format>source</format> </language>
	2.6	<agent-mobility-protocol>full-migration-protocol </agent-mobility-protocol>
	2.7	<agent-code>aw.platform.DirectoryFacilitator </agent-code>

Table 4.5: Test Messages for Testing REFUSE

The third case is the "agree" message. MMS sends this message when it can parse the received message and can provide the services required. The obvious test case is the one where the service required is the same as the service offered – but there are other cases as well. For example, when dealing with Java – the operating system doesn't matter. The requesting agent may claim to require Windows – but if it doesn't actually specify any Windows specific components that it – the request is ignored. The implementation is pure Java, and therefore platform independent. The test cases are in Table 4.6 – only relevant parts of the content of the ACL message are displayed.

Expected Result: AGREE		
Sender	**Case**	**Test Message**

Expected Result: AGREE		
Sender	**Case**	**Test Message**
SearchAgent	3.0	<fipa-mobile-agent-description> <agent-name>searcher</agent-name> <address>iiop://holly:9000/acc</address> <destination>iiop://holly:10000/acc</destination> <agent-profile> <system> <name>FIPA-OS</name> <major-version>1.03</major-version> <minor-version>1.03</minor-version> <dependencies> MobilityManagementSystem </dependencies> </system> <language> <name>Java</name> <minor-version>1.2</minor-version> <major-version>1.2.2</major-version> <format>bytecode</format> </language> </agent-profile> <agent-mobility-protocol> simple-migration-protocol </agent-mobility-protocol> <agent-code>mobility.SearchAgent</agent-code> <agent-data>1 c:\ milla vaio 0</agent-data> </fipa-mobile-agent-description>
	3.1	<fipa-mobile-agent-description> <agent-name>searcher</agent-name> <address>iiop://holly:9000/acc</address> <destination>iiop://holly:10000/acc</destination> <agent-profile> <system> <name>FIPA-OS</name> <major-version>1.03</major-version> <minor-version>1.03</minor-version> </system> <language> <name>Java</name> <minor-version>1.2</minor-version> <major-version>1.2.2</major-version> <format>bytecode</format> </language> <os> <name>Windows</name> <major-version>98</major-version> <minor-version>95</minor-version> <hardware>32 RAM</hardware> </os> </agent-profile> <agent-mobility-protocol> simple-migration-protocol </agent-mobility-protocol> <agent-code>mobility.SearchAgent</agent-code> <agent-data>1 c:\ txt vaio 0</agent-data> <agent-version>3.0</agent-version> <signature>_Milla_</signature> </fipa-mobile-agent-description>

Table 4.6: Test Messages for Testing AGREE

Although timing and efficiency would normally be amongst the topics tested, it was decided not to test these. This is because at the moment, there is no technology that these results could be compared against – and by themselves they wouldn't really mean anything.

Because of RAD, all the individual components of the agents are tested while developing them. This includes components like agent descriptions (package mobility.ont.fipaman), the parser (mobility.parser.xml.MobileAgentDescription), profile objects (package aw.ont.profile), search filter (mobility.util.SearchFilter) and GUIs.

5 RESULTS AND DISCUSSION

5.1 RESULTS

Full test messages used will be in "Appendix J: Test Messages".

5.1.1 HARDWARE USED

Two networked computers at Nottingham Trent University's computer lab were used. The two computers hardware was identical, and have following specification:

- Windows NT 4.0
- x86 Family 5 Model 8 Stepping 12, AT/AT Compatible
- 130 MB of RAM

5.1.2 TEST RESULTS

The final test results can be seen in Table 3.1. Screenshots of the testing can be seen in "**Error! Reference source not found.**".

Case	Expected Result?	Notes
1.0	✓	Everything as planned.
2.0	✓	Everything as planned.
2.1	✓	Everything as planned.
2.2	✓	Everything as planned.
2.3	✓	Everything as planned.
2.4	✓	Everything as planned.
2.5	✓	Everything as planned.
2.6	✓	Everything as planned.
2.7	✓	Everything as planned.
3.0	✓	Took several tries – home MMS kept running out of memory. All other applications have to be shut down.
3.1	✓	Everything as planned.

Table 3.1: Test Results

Due to comprehensive testing of the sub-components during the development phase, the final testing was very successful. All the tests went well, only problems encountered were technical, and not related to the software.

5.2 MOBILITY VS. MESSAGING

Is mobility necessary? What can mobile agents do that strategically situated stationary agents can't do? This scenario – like most mobile agent scenarios – could have been implemented by sending messages to a remote agent at the desired location and ask it to perform the search – and it would have been faster too. Why use mobile agents then?

Let's think about a scenario, where an agent would perform a short, but critical task, perhaps once a day. This task could be for example a virus check that would be done every day when system maintenance requires it. Now, to be able to do the check, the agent will have to be running all the time to be able to get the message and will end up using unnecessary resources at the location. It may be performing the task for two minutes a day, and be idle eight hours.

When using mobile agents, whether it is Voyager or MMS, there will have to be a server running in the computer to wait for mobile agents. Whether this is better than having an idle agent running depends on number of other possible mobile agents that might want to migrate to this location.

There are occasions, where mobility is the only alternative, but those are mainly for military purposes where use of bandwidth is very limited and may be dangerous. It has been argued that agents (applications) could migrate into mobile devices and then stay there while the connection to the network is severed. However, in situations where agent

will have to be independent, it doesn't matter if it's mobile or not. After the connection has been restored it doesn't matter whether the agent is going to migrate back, or if it's just going to send a message to another agent who needs the information it has gathered while it was out of touch – result is the same.

Within FIPA there has been a lot of discussion about the mobility (first defined in FIPA 98 specification), and this issue is going to be one of the main concerns within FIPA in 2000. One of the purposes of this project was to help in this dialog – to provide some kind of feedback to the mobility specifications. Since FIPA-OS was at the time the only open source FIPA platform available, it made sense to use it – and in the spirit of open source, the code produced for this project will be open source as well and will be submitted to FIPA-OS community.

5.3 LIMITATIONS

Mobile agent systems will always be dependent on the platform it has been implemented, whether it is Voyager or aglets, or in this case FIPA-OS. This can't be helped and it limits the use of the technology. This particular application needs the MMS (or the AMS) to be in the platform, and in the right physical place.

Limitations however can be useful as well – there has to be a way for the owner of the system to control what software runs and what software is potentially dangerous. Mobile agents can potentially negotiate the security lever/method before they ask to be moved. This way a level of security could be achieved.

Biggest limitation to this system, however, is the fact that the agents have to be modified to be mobile agents so that they can store their instance data – and then be able

to take the data in while initialising. In this particular example, it's not a problem – but the agents could potentially be very large and hold megabytes or even gigabytes of information. However, if agent is designed to be mobile – this isn't as likely to happen since an agent with large database (for example) shouldn't have any reason to move) – and a lot can be achieved with good design.

5.3.1 INSTANCE DATA

At the moment, instance data in encoded just as a string with pieces of data separated by spaces (" "). With MMS dedicated to moving only this one particular agent, and with only four pieces of data, this is feasible solution. Would this work in more general cases, or is some other kind of solution needed, like some kind of encoding?

MMS doesn't comment on agent-data field – it's only a part of agent description. In the scenario implemented, the mobile agent description is given to the agent in the constructor. Agent uses this information mainly to send move messages, but agent-data field is used to decide what to do next.

There could be a convention that MMS would always pass on the mobile agent description to the agent when starting – and would let the agent deal with the data. This way the MMS when receiving a move request would only have to check that it could provide the required services, and let the agent interpret the agent data. Also, MMS' operation would always remain the same and to accommodate for receiving new agents MMS wouldn't have to be changed.

5.3.2 EXECUTE

Execute command has been implemented as an ACL message. However, FIPA classifies this as a platform internal issue, and therefore doesn't specify how to implement it. The idea of the act is to activate the agent – so what other alternatives might there be?

Just the starting up the agent could be seen as an execute act. This would mean that the MMS (or the AMS) wouldn't have any control over the agent once it was started. If using execute as a message, MMS could start up the agent – and not letting it execute until given permission – perform other duties first, like informing the AMS of the move, and then ask the agent to execute. This will save some additional time, and give more flexibility.

Agent will have to be able to receive messages to receive the "execute" command, so it will have to ignore (send "not-understood"s as defined in FIPA protocols) all the other messages it may receive. It will have to recognise messages coming from the agent that started it (maybe its owner?) and only respond to that. Without the "execute" command agent will not do anything – which means that if the MMS crashed in between, the agent will never "wake-up" and perform. This can be a weak point in the implementation.

6 CONCLUSIONS AND FUTURE WORK

This chapter discusses the conclusions that can be drawn from the work done in this project. State mobility is compared to code mobility, and future work that is not included due to lack of time is suggested.

6.1 STATE VS. CODE MOBILITY

State mobility is often overlooked in favour of code mobility. Systems supporting code mobility – like Voyager – often aim to be "platform independent", but still require bindings to the system making it less so. Mobility system based on Voyager will then only be able to move agents using Voyager, the same goes for aglets.

It seems that it's not possible to provide a platform independent mobility. However, this can also be taken as an advantage. Security is often described as the major stepping stone in agent mobility – for example, how to recognise and prevent hostile agents coming to the platform. With platform independent mobility, security would be almost impossible to solve – the more specific the mobility type is, the easier the security and the restrictions are to implement.

Code mobility is often chosen because it seems to be the easier one to implement, and it favours "platform independence". If platform independence is discarded – the implementation is matter of opinion.

State mobility implemented in this project is very straightforward: state is sent as a string – the Mobility Management System doesn't analyse this data but lets the agent handle it. This reduces the load of MMS – and who better to interpret the agent data than the agent who sent it.

Although in the prototype produced MMS is only able to move one specific agent – this could easily be automated. Security could be added so that only certain trusted agents are authorised to move, and MMS could check the whether agent is authorised before the move. Since there has to be an MMS in every host agents want to move to, this information could be very specific. In very secure locations maybe only one agent whose presence is necessary is trusted enough – in more relaxed areas the list can be longer.

Not much work has been done on state mobility – it mentioned, but seldom implemented. This project has also been a study into this much-unknown field.

6.2 FUTURE WORK

The project has produced a prototype, and much more work could be done with the subject.

6.2.1 FIPA COMPLIANCE

The scenario implemented isn't fully FIPA compliant. To run this scenario, only two agents are present, whereas on a FIPA platform AMS, DF and ACC have to be present. FIPA agents, for example, have to always register with the AMS.

6.2.2 MMS

In the scenario implemented, MMS is only able to move the SearchAgent – the agents constructor is hard coded. If in the future, the convention of adding a constructor with FipaMobileAgentDescription to agent desiring mobility would be used, then MMS could make starting up agents automatic. MMS could look for a constructor with the

FipaMobileAgentDescription, and dynamically load up the agent using that constructor.

6.2.3 AMS

For future work, it would be useful to integrate MMS into the FIPA-OS platform. MMS could be a delegate agent for the AMS – when there are MMS' in the platform, AMS supports mobility to the locations where MMS' are. Then agents would always contact AMS (they wouldn't even know about the existence of delegates) and from their point of view – and FIPAs – AMS would do the moving.

[1] O'BRIEN, P. D. and NICOL, R. C., 1998. FIPA – towards a standard for software agents. **BT Technol J**, volume 16 No 3 July, 51-59.

[2] **FIPA 98 Specification Part 11: Agent Management Support for Mobility** [online]. Foundation for Intelligent Physical Agents at Geneva, Switzerland, 23 October 1998. Available at <http://www.fipa.org/spec/fipa8a27.doc> [Accessed 1 September 1999]

[3] **FIPA 99 Specification Part 14, draft 0.2.2: Nomadic Application Support**. Foundation for Intelligent Physical Agents at Geneva, Switzerland, 15 September 1999.

[4] **FIPA-OS Information** [online]. Nortel Networks. Available at <http://www.nortelnetworks.com/products/announcements/fipa/info.html> [Accessed 26 February 2000]

[5] LANGE, D. B., February 19, 1997. **Java Aglet Application Programming Interface (J-AAPI) White Paper - Draft 2** [online]. IBM Tokyo Research Laboratory. Available at <http://www.trl.ibm.co.jp/aglets/JAAPI-whitepaper.html> [Accessed 26 February 2000]

[6] OSHIMA M., KARJOTH, G., and ONO, K., September 8, 1998. **Aglets Specification 1.1 Draft** [online]. IBM Tokyo Research Laboratory. Available at <http://www.trl.ibm.co.jp/aglets/spec11.html> [Accessed 26 February 2000]

[7] GLASS, G., 1999. **Overview of Voyager: ObjectSpace's Product Family for State-of-the-Art Distributed Computing** [online]. ObjectSpace, 1999. Available at

<http://www.objectspace.com/products/documentation/VoyagerOverview.pdf> [Accessed 27 February 2000]

[8] **ORB 3.2 Developer Guide** [online]. ObjectSpace, 1995-1999. Available at <http://www.objectspace.com/products/documentation/voyager_html_docs/orb/index.htm> [Accessed 27 February 2000]

[9] **FIPA 98 Specification Part 1: Agent Management** [online]. Foundation for Intelligent Physical Agents at Geneva, Switzerland, 23 October 1998. Available at <http://www.fipa.org/spec/fipa8a23.doc> [Accessed 1 September 1999]

[10] **Extensible Markup Language (XML) 1.0** [online]. W3C, 1998. Available at <http://www.w3.org/TR/1998/REC-xml-19980210> [Accessed 7 March 2000]

[11] **Kawa Home Page** [online]. Tek-Tools, 1999. Available at <http:\\www.tek-tools.com\kawa>

[12] **Code Conventions for the Java Programming Language** [online]. Sun Microsystems, 1999. Available at <http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html> [Accessed 11 March 2000]

[13] BUCKLE, P., 2000. FIPA-OS [online]. In: **FACTS workshop, Ipswich, 28 February 2000.** Available at <http://www.labs.bt.com/profsoc/facts/workshop/index.htm> [Accessed March 13 2000]

[14] **PersonalJava Technology White Paper** [online]. Sun Microsystems, 1998. Available at <http://java.sun.com/products/personaljava/pj_white.pdf> [Accessed

27 March 2000]

[15] **Java Technology: An Early History** [online]. Sun Microsystems, 1998. Available at <http://java.sun.com/features/1998/05/birthday.html> [Accessed 27 March 2000]

[16] GOSLING, J. **A Brief History of the Green Project** [online]. James Gosling. Available at <http://java.sun.com/people/jag/green/index.html> [Accessed 27 March 2000]

[17] FRANKLIN, S., GRAESSE, A., 1996. **Software Agents: An Overview** [online]. BT Laboratories, 1996. Available at <http://www.cs.umbc.edu/agents/papers/ao.ps> [Accessed 28 March 2000]

[18] **Overview of Servlets** [online]. Sun Microsystems, 2000. Available at <http://web2.java.sun.com/docs/books/tutorial/servlets/overview/index.html> [Accessed 28 March 2000]

HAUBEN, M. **History of ARPANET** [online]. Michael Hauben. Available at <http://www.dei.isep.ipp.pt/docs/arpa.html> [Accessed 27 March 2000]

History of the World Wide Web [online]. Deutschen Welle, 1999. Available at <http://www.w3history.org/> [Accessed 27 March 2000]

FIPA 97 Specification Part 1, Version 2.0: Agent Management [online]. Foundation for Intelligent Physical Agents at Geneva, Switzerland, 23 October 1998. Available at <http://www.fipa.org/spec/f8a21.doc> [Accessed 1 September 1999]

FIPA 97 Specification Part 2, Version 2.0: Agent Communication Language [online]. Foundation for Intelligent Physical Agents at Geneva, Switzerland, 23 October 1998. Available at <http://www.fipa.org/spec/f8a22.zip> [Accessed 1 September 1999]

CHESS, D., HARRISON, C. and KERSHENBAUM A., 16.3.1995. **Mobile Agents: Are They a Good Idea?** [online]. IBM Research Division: T. J. Watson Research Centre, New York. Available at <http://www.research.ibm.com/massdist/mobag.ps> [Accessed 17 November 1999]

IBM Aglets Software Development Kit Home Page [online]. IBM. Available at

<http://www.trl.ibm.co.jp/aglets/index.html> [Accessed 26 February 2000]

SCHNEIDER, M., 1996-1999. **Distributed Objects & Components: Mobile Agents** [online]. The Distributed Systems Group in the Information Systems Institute of the Technical University of Vienna. Available at <http://www.infosys.tuwien.ac.at/cetus/oo_mobile_agents.html> [Accessed 17 November 1999]

VENNERS, B., April 1997. **Under the Hood: The architecture of aglets** [online]. JavaWorld. Available at <http://www.javaworld.com/javaworld/jw-04-1997/jw-04-hood.html> [Accessed 26 February 2000]

VENNERS, B., May 1997. **Solve Real Life Problems with Aglets** [online]. JavaWorld. Available at < http://www.javaworld.com/javaworld/jw-05-1997/jw-05-hood.html> [Accessed 26 February 2000]

ObjectSpace Voyager Home Page [online]. ObjectSpace, 1995-2000. Available at <http://www.objectspace.com/products/prodVoyager.asp> [Accessed 27 February 2000]

GLASS, G., 1999. **The ObjectSpace Voyager Universal ORB** [online]. ObjectSpace, 1999. Available at

<http://www.objectspace.com/products/documentation/VgerUnivOrb.pdf> [Accessed 27 February 2000]

AÑEZ, J., 1999. **Achieving Synergy: Voyager ORB 3.0 from ObjectSpace** [online]. SIGS Publications, 1999-2000. Available at <http://www.objectspace.com/company/javareport-dec99.asp> [Accessed 27 February 2000]

ObjectSpace Voyager, GeneralMagic Odyssey and IBM Aglets: A Comparison [online]. ObjectSpace, June 1997. Available at <http://www.infosys.tuwien.ac.at/Research/Agents/archive/VoyagerAgentComparisons.PDF> [Accessed 27 February 2000]

KINIRY, J. and ZIMMERMAN, D., August 1997. **A Hands-on Look at Java Mobile Agents** [online]. IEEE Internet Computing, July/August 1997. Available at <http://www.infosys.tuwien.ac.at/Research/Agents//archive/w4021.pdf> [Accessed 27 February 2000]

SAX 1.0: The Simple API for XML [online]. Available at <http://www.megginson.com/SAX/sax.html> [Accessed 7 March 2000]

HUMPHREY, W. S., 1999. **Introduction to the Personal Software Process**. Reading, Massachusetts, USA: Addison Wesley.

INTERNATIONAL MERITO FORUM, 2000. **What's New in Distributed Software Technology? Helsinki, 20 – 21 March 2000.** Helsinki: International Merito Forum.

BRICKLEY, D., GUHA, R. V., 2000. **Resource Description Framework (RDF) Schema Specification 1.0: W3C Candidate Recommendation** [online]. W3C, 27 March 2000. Available at <http://www.w3.org/TR/2000/CR-rdf-schema-20000327/> [Accessed 28 March 2000]

APPENDIX A: SYMBOLS, TERMS AND DEFINITIONS IN FIPA

Symbol	Definition
ACC	Agent Communication Channel
ACL	Agent Communication Language
AMS	Agent Management System
AP	Agent Platform
DF	Directory Facilitator
GUID	Global Unique Identifier
HAP	Home Agent Platform
IPMT	Internal Platform Message Transport

Symbols used in FIPA [2]

Term	Definition
Stationary agent	An agent that executes only upon the AP where it begins executing and is reliant upon it.
Mobile agent	An agent that is not reliant upon the AP where it began executing and can subsequently transport itself between APs.
Mobility	The property or characteristic of an agent that allows it to travel between APs. When a platform does not support the mobility, it answers with errors to mobility queries.
Location	The location of an agent is associated with the AP.
Agent migration	The process by which an agent transports itself between APs.
Agent cloning	The process by which an agent creates a copy of itself on an AP.
Agent invocation	The process by which an agent can create another instance of an agent on an AP.
Agent state	Describes the execution state, or attribute values of an agent.
Agent code	The set of instructions used by an agent.
Agent data	Any data associated with an agent.

Terms and definitions used in FIPA [2],[3]

APPENDIX B: MOBILITY ONTOLOGY

FIPA-MOBILE-AGENT-DESCRIPTION [2],[3]

Parameter	Description	Action	
		move	*transfer*
:agent-name	Denotes the GUID of the agent.	Mandatory	Mandatory
:address	Denotes the communication address of the agent.	Mandatory	Optional
:destination	Denotes the destination AMS of the agent.	Optional	Optional
:agent-profile	Denotes the specification of the requirements of the agent.	Optional	Optional
:agent-mobility-protocol	Denotes the protocol used for agent mobility.	Optional	Optional
:agent-code	Denotes the code base of the agent.	Mandatory	Optional
:agent-codebase-name	Denotes the name of code base.	Mandatory	Optional
:agent-data	Denotes any data associated with the agent.	Optional	Optional
:agent-version	Denotes the version of the agent.	Optional	Optional
:signature	Denotes the encrypted identity and authority of the agent.	Optional	Mandatory

FIPA-MOBILE-AGENT-PROFILE [2]

Parameter	Description	Action
		move
:system	Denotes the mobile agent system environment required by the agent.	Optional
:language	Denotes the language environment required by the agent.	Mandatory
:os	Denotes the operating system environment required by the agent.	Optional

FIPA-MOBILE-AGENT-SYSTEM [2]

Parameter	Description	Action
		move
:name	Denotes the name of the mobile agent system.	Optional
:major-version	Denotes the major version number of the mobile agent system.	Optional
:minor-version	Denotes the minor version number of the mobile agent system.	Optional
:dependencies	Denotes the dependencies required by the agent for the mobile agent system.	Optional

FIPA-MOBILE-AGENT-LANGUAGE [2]

Parameter	Description	Action
		move
:name	Denotes the name of the language.	Mandatory
:major-version	Denotes the major version number of the language.	Mandatory
:minor-version	Denotes the minor version number of the language.	Optional
:format	Denotes the format of the code base of the agent.	Mandatory
:filter	Denotes any filter that should be executed over the code base before execution.	Optional
:dependencies	Denotes any dependencies required by the agent for the language.	Optional

FIPA-MOBILE-AGENT-OS [2]

Parameter	Description	Action
		move
:name	Denotes the name of the operating system.	Optional
:major-version	Denotes the major version number of the operating system.	Optional
:minor-version	Denotes the minor version number of the operating system.	Optional
:hardware	Denotes the name of the hardware.	Optional
:dependencies	Denotes any dependencies required by the agent for the operating system.	Optional

APPENDIX C: "DESCRIPTION" DATA TYPE DEFINITION

```
<!ELEMENT action (move | move-state | move-codebase | transfer) >
<!ATTLIST action to NMTOKEN #REQUIRED >
<!ELEMENT move (fipa-mobile-agent-description) >
<!ELEMENT move-state (fipa-mobile-agent-description) >
<!ELEMENT move-codebase (fipa-mobile-agent-description) >
<!ELEMENT transfer (fipa-mobile-agent-description) >
<!ELEMENT fipa-mobile-agent-description (agent-name, address, destination?,
agent-profile?, agent-mobility-protocol?, agent-code, agent-data?, agent-
version?, signature?) >
<!ELEMENT agent-name (#PCDATA) >
<!ELEMENT address (#PCDATA) >
<!ELEMENT destination (#PCDATA) >
<!ELEMENT agent-profile (system?, language, os?) >
<!ELEMENT system (name?, major-version?, minor-version?, dependencies?) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT major-version (#PCDATA) >
<!ELEMENT minor-version (#PCDATA) >
<!ELEMENT dependencies (#PCDATA) >
<!ELEMENT language (name, major-version, minor-version?, format, filter?,
dependencies?) >
<!ELEMENT format (#PCDATA) >
<!ELEMENT filter (#PCDATA) >
<!ELEMENT os (name?, major-version?, minor-version?, hardware?, dependencies?)
>
<!ELEMENT hardware (#PCDATA) >
<!ELEMENT agent-mobility-protocol (#PCDATA) >
<!ELEMENT agent-code (#PCDATA) >
<!ELEMENT agent-data (#PCDATA) >
<!ELEMENT agent-version (#PCDATA) >
<!ELEMENT signature (#PCDATA) >
```

APPENDIX D: SEARCHAGENT RDF SCHEMA

```
<rdf:RDF xml:lang="en"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdfs:Class rdf:ID="MMS">
    <rdfs:comment>The class of MMS entities.</rdfs:comment>
</rdfs:Class>

<rdf:Property ID="name">
    <rdfs:comment>The name property of MMS.</rdfs:comment>
    <rdfs:domain rdf:resource="#MMS"/>
</rdf:Property>
```

APPENDIX E: MOBILEAGENTDESCRIPTION RDF SCHEMA

```
<rdf:RDF xml:lang="en"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdfs:Class rdf:ID="MobileAgentDefinition">
    <rdfs:comment>The class of Mobile Agent Definition entities.</rdfs:comment>
</rdfs:Class>

<rdf:Property ID="agent-mobility-protocol">
    <rdfs:comment>The agent mobility protocol name.</rdfs:comment>
    <rdfs:domain rdf:resource="#MobileAgentDefinition"/>
</rdf:Property>

<rdf:Property ID="system-name">
    <rdfs:comment>The system name.</rdfs:comment>
    <rdfs:domain rdf:resource="#MobileAgentDefinition"/>
</rdf:Property>

<rdf:Property ID="system-major">
    <rdfs:comment>The system major version.</rdfs:comment>
    <rdfs:domain rdf:resource="#MobileAgentDefinition"/>
</rdf:Property>

<rdf:Property ID="system-minor">
    <rdfs:comment>The system minor version.</rdfs:comment>
    <rdfs:domain rdf:resource="#MobileAgentDefinition"/>
</rdf:Property>

<rdf:Property ID="system-dependencies">
    <rdfs:comment>The possible system dependencies.</rdfs:comment>
    <rdfs:domain rdf:resource="#MobileAgentDefinition"/>
</rdf:Property>

<rdf:Property ID="language-name">
    <rdfs:comment>The language name agent will use.</rdfs:comment>
    <rdfs:domain rdf:resource="#MobileAgentDefinition"/>
</rdf:Property>

<rdf:Property ID="language-major">
    <rdfs:comment>The language major version.</rdfs:comment>
    <rdfs:domain rdf:resource="#MobileAgentDefinition"/>
```

```
</rdf:Property>

<rdf:Property ID="language-minor">
    <rdfs:comment>The language minor version.</rdfs:comment>
    <rdfs:domain rdf:resource="#MobileAgentDefinition"/>
</rdf:Property>

<rdf:Property ID="language-format">
    <rdfs:comment>Format of the language agent needs.</rdfs:comment>
    <rdfs:domain rdf:resource="#MobileAgentDefinition"/>
</rdf:Property>

<rdf:Property ID="hardware-os">
    <rdfs:comment>Any hardware the agent needs.</rdfs:comment>
    <rdfs:domain rdf:resource="#MobileAgentDefinition"/>
</rdf:Property>
```

APPENDIX F: FILES NEEDED TO RUN THE SOFTWARE

Java source files (compiled version in the classpath):

```
AgentProfile.java

FipaMobileAgentDescription.java

FipaMobileAgentLanguage.java

FipaMobileAgentOs.java

FipaMobileAgentProfile.java

FipaMobileAgentSystem.java

MMS.java

MobileAgentDescription.java

MobileAgentDescriptionHandler.java

MobilityManagementSystem.java

Profile.java

SearchAgent.java

SearchFilter.java

StartingGUI.java
```

Agent profiles:

```
platform.profile

mms.profile[11]

searcher.profile[12]
```

FIPA-OS (in the classpath)[13]:

[11] If there are two MMS', names will be as defined in agent profile.

```
FIPA_OS_v1_03_debug.jar

FIPA_OS_v1_03_CT_debug.jar

FIPA_OS_v1_03_platform_debug.jar
```

Third party software (in the classpath):

```
SiRPAC-1.14 classes

xml4j_1_1_16.jar
```

[12] This profile is named according to the agents name; the default is "searcher".

[13] Demonstration tested with "debug" versions of the software, but non-debug jars should work as well.

APPENDIX G: INSTRUCTIONS FOR RUNNING THE SOFTWARE

Assumptions:

- FIPA-OS 1.03 installed with third party software to hosts used in the demonstration, and all classes are in the classpath.
- Agent profiles have been set up according to the FIPA-OS instructions for all hosts. Profiles needed are in Appendix I: Agent Profiles.
- Demonstration Java classes have been compiled and the location of the class files is in the classpath before FIPA-OS classes[14] for the hosts.
- description.dtd (from "Appendix C: "Description" Data Type Definition") that is needed to validate the XML document, must be in the same directory as where you're going to run the code.

Run the software following steps:

1. Start up both the MMS' by giving them two parameters from the command line, first the location of the platform profile and then the name of the MMS. If you're using two hosts, the MMS' names have to be different (and correspond to the names in the agent profiles). Example:

   ```
   C:\>java mobility.MobilityManagementSystem c:\profiles\platform.profile mms1
   ```

   This will start up an agent called "mms1" for the platform defined in "c:\profiles\platform.profile" file.

2. Start up SearchGUI by giving it one parameter from the command line: the location of the platform profile. Example:

```
C:\>java mobility.gui.StartingGUI c:\profiles\platform.profile
```

This will start up the GUI that can start up agents for the platform defined in "c:\profiles\platform.profile" file. The GUI can be seen in **Error! Reference source not found.**

3. Enter the search criteria for the search to be performed: the directory and the string to be searched. Then choose whether to start local or mobile agent by pressing the appropriate buttons. Since starting the local agent is going to end the scenario by next returning the results, from now on we're interested what happens if mobile agent is chosen.

4. Next, the mobile description GUI appears as seen in **Error! Reference source not found.**. Most of the information is filled in as the default information. If the default information is suitable, only field that will have to be filled in is the destination. Enter here the DNS name for the host of the destination MMS. It's important that the "agent-data" field is not changed. When all the data has been entered into the form, press the button to start the agent.

5. Wait. Following things should happen:

   I The agent sends a message to the destination MMS requesting the move. MMS accepts, and the agent shuts down.

   II Destination MMS starts up a new agent and sends it a request to execute. Agent receives this, accepts and starts the search. You can follow the progress of the search from the agents GUI (as displayed in **Error! Reference source not found.**).

[14] Modified FIPA-OS classes need to be the ones to used overriding the classes in the FIPA-OS jars.

III After agent has finished it sends a message to home MMS requesting a move back. MMS accepts, and agent shuts down.

IV Home MMS starts up the agent and sends it a request to execute. Agent receives this, accepts and displays the results (as displayed in **Error! Reference source not found.**).

6. Click "OK".

APPENDIX H: JAVA CODE

AGENTPROFILE

Only one method is added to the file – to save space only it is shown:

```
public Hashtable getNonStandards()
{
  return _profile.getProfiles();
}
```

```
/**
  * Java object of the fipa-mobile-agent-desrciprion.
  * <br>
  * @author Milla Makelainen, 2000
  */

package mobility.ont.fipaman;

public class FipaMobileAgentDescription
{
    private String _agent_name = null;
    private String _address = null;
    private String _destination = null;
    private FipaMobileAgentProfile _agent_profile = null;
    private String _agent_mobility_protocol = null;
    private String _agent_code = null;
    private String _agent_data = null;
    private String _agent_version = null;
    private String _signature = null;

    /**
    * Default constructor, instanciates all the instance variables.
    */
    public FipaMobileAgentDescription()
    {
        _agent_name = new String();
        _address = new String();
        _destination = new String();
        _agent_profile = new FipaMobileAgentProfile();
        _agent_mobility_protocol = new String();
        _agent_code = new String();
        _agent_data = new String();
        _agent_version = new String();
        _signature = new String();
    }

//--------------------------------------------------------------------------//
// GET/SET functions                                                        //
//--------------------------------------------------------------------------//
    public void setAgentName(String name)
    {
        _agent_name = name;
    }

    public String getAgentName()
    {
        return _agent_name;
    }

    public void setAddress(String address)
    {
        _address = address;
    }

    public String getAddress()
    {
        return _address;
    }

    public void setDestination(String destination)
    {
        _destination = destination;
    }

    public String getDestination()
    {
        return _destination;
    }
```

```
public void setAgentProfile(FipaMobileAgentProfile profile)
{
    _agent_profile = profile;
}

public FipaMobileAgentProfile getAgentProfile()
{
    return _agent_profile;
}

public void setAgentMobilityProtocol(String protocol)
{
    _agent_mobility_protocol = protocol;
}

public String getAgentMobilityProtocol()
{
    return _agent_mobility_protocol;
}

public void setAgentCode(String code)
{
    _agent_code = code;
}

public String getAgentCode()
{
    return _agent_code;
}

public void setAgentData(String data)
{
    _agent_data = data;
}

public String getAgentData()
{
    return _agent_data;
}

public void setAgentVersion(String version)
{
    _agent_version = version;
}

public String getAgentVersion()
{
    return _agent_version;
}

public void setSignature(String signature)
{
    _signature = signature;
}

public String getSignature()
{
    return _signature;
}

public String toString()
{
    return (" <fipa-mobile-agent-description>" +
        " <agent-name>" + _agent_name + "</agent-name>" +//agent-name is mandatory
        ((!_address.equals("")) ? (" <address>" + _address + "</address>") : "") +
        ((!_destination.equals("")) ? (" <destination>" + _destination +
        "</destination>") : "") +
        ((!_agent_profile.isEmpty()) ? (" <agent-profile>" +
        _agent_profile.toString() + " </agent-profile>") : "") +
        ((!_agent_mobility_protocol.equals("")) ? (" <agent-mobility-protocol>" +
```

```
            _agent_mobility_protocol + "</agent-mobility-protocol>") : "") +
            " <agent-code>" + _agent_code + "</agent-code>" +//agent-code mandatory
            ((!_agent_data.equals("")) ? (" <agent-data>" + _agent_data +
            "</agent-data>") : "") +
            ((!_agent_version.equals("")) ? (" <agent-version>" + _agent_version +
            "</agent-version>") : "") +
            ((!_signature.equals("")) ? (" <signature>" + _signature +
            "</signature>") : "") +
            " </fipa-mobile-agent-description>"
            );
    }


}
```

```
/**
  * Java object of the fipa-mobile-agent-language.
  * <br>
  * @author Milla Makelainen, 2000
  */

package mobility.ont.fipaman;

public class FipaMobileAgentLanguage
{
    private String _name = null;
    private String _major_version = null;
    private String _minor_version = null;
    private String _format = null;
    private String _filter = null;
    private String _dependencies = null;

  /**
    * Default constructor, instanciates all the instance variables.
    */
    public FipaMobileAgentLanguage()
    {
        _name = new String();
        _major_version = new String();
        _minor_version = new String();
        _format = new String();
        _filter = new String();
        _dependencies = new String();
    }
//-------------------------------------------------------------------------//
// GET/SET functions                                                       //
//-------------------------------------------------------------------------//

    public void setName(String name)
    {
        _name = name;
    }

    public String getName()
    {
        return _name;
    }

    public void setMajorVersion(String version)
    {
        _major_version = version;
    }

    public String getMajorVersion()
    {
        return _major_version;
    }

    public void setMinorVersion(String version)
    {
        _minor_version = version;
    }

    public String getMinorVersion()
    {
        return _minor_version;
    }

    public void setFormat(String format)
    {
        _format = format;
    }
```

```
    public String getFormat()
    {
        return _format;
    }

    public void setFilter(String filter)
    {
        _filter = filter;
    }

    public String getFilter()
    {
        return _filter;
    }

    public void setDependencies(String dependencies)
    {
        _dependencies = dependencies;
    }

    public String getDependencies()
    {
        return _dependencies;
    }

    public boolean isEmpty()
    {
        return (((_name == null) || (_name.equals(""))) &&
                        ((_major_version == null) || (_major_version.equals(""))) &&
                        ((_minor_version == null) || (_minor_version.equals(""))) &&
                        ((_format == null) || (_format.equals(""))) &&
                        ((_filter == null) || (_filter.equals(""))) &&
                        ((_dependencies == null) || (_dependencies.equals(""))) );
    }

    public String toString()
    {
        return ( " <name>" + _name + "</name>" +            //name is mandatory
             " <major-version>" + _major_version
            + "</major-version>" +//major-version is mandatory
            ((_minor_version.equals("")) ? (" <minor-version>" + _minor_version +
            "</minor-version>") : "") +
            " <format>" + _format +   "</format>" +  //format is mandatory
            ((!_filter.equals("")) ? (" <filter>" + _filter + "</filter>") : "") +
            ((!_dependencies.equals("")) ? (" <dependencies>" + _dependencies +
            "</dependencies>") : "") );
    }
}
```

```
/**
  * Java object of the fipa-mobile-agent-os.
  * <br>
  * @author Milla Makelainen, 2000
  */

package mobility.ont.fipaman;

public class FipaMobileAgentOs
{
    private String _name = null;
    private String _major_version = null;
    private String _minor_version = null;
    private String _hardware = null;
    private String _dependencies = null;

  /**
    * Default constructor, instanciates all the instance variables.
    */
    public FipaMobileAgentOs()
    {
        _name = new String();
        _major_version = new String();
        _minor_version = new String();
        _hardware = new String();
        _dependencies = new String();
    }
//-------------------------------------------------------------------------//
// GET/SET functions                                                       //
//-------------------------------------------------------------------------//

    public void setName(String name)
    {
        _name = name;
    }

    public String getName()
    {
        return _name;
    }

    public void setMajorVersion(String version)
    {
        _major_version = version;
    }

    public String getMajorVersion()
    {
        return _major_version;
    }

    public void setMinorVersion(String version)
    {
        _minor_version = version;
    }

    public String getMinorVersion()
    {
        return _minor_version;
    }

    public void setHardware(String hw)
    {
        _hardware = hw;
    }

    public String getHardware()
    {
```

```
        return _hardware;
    }

    public void setDependencies(String dependencies)
    {
        _dependencies = dependencies;
    }

    public String getDependencies()
    {
        return _dependencies;
    }

    public boolean isEmpty()
    {
        return (((_name == null) || (_name.equals(""))) &&
                       ((_major_version == null) || (_major_version.equals(""))) &&
                       ((_minor_version == null) || (_minor_version.equals(""))) &&
                       ((_hardware == null) || (_hardware.equals(""))) &&
                       ((_dependencies == null) || (_dependencies.equals(""))) );
    }

    public String toString()
    {
        return ( ((!_name.equals("")) ? ("<name>" + _name + "</name>") : "") +
            ((!_major_version.equals("")) ? (" <major-version>" + _major_version +
            "</major-version>") : "") +
            ((!_minor_version.equals("")) ? (" <minor-version>" + _minor_version +
            "</minor-version>") : "") +
            ((!_hardware.equals("")) ? (" <hardware>" + _hardware +
            "</hardware>") : "") +
            ((!_dependencies.equals("")) ? (" <dependencies>" + _dependencies +
            "</dependencies>") : "") );
    }
}
```

```
/**
  * Java object of the fipa-mobile-agent-profile.
  * <br>
  * @author Milla Makelainen, 2000
  */

package mobility.ont.fipaman;

public class FipaMobileAgentProfile
{
    private FipaMobileAgentSystem _system = null;
    private FipaMobileAgentLanguage _language = null;
    private FipaMobileAgentOs _os = null;

  /**
    * Default constructor, instanciates all the instance variables.
    */
    public FipaMobileAgentProfile()
    {
        _system = new FipaMobileAgentSystem();
        _language = new FipaMobileAgentLanguage();
        _os = new FipaMobileAgentOs();
    }
//-------------------------------------------------------------------------//
// GET/SET functions                                                       //
//-------------------------------------------------------------------------//

    public void setSystem(FipaMobileAgentSystem system)
    {
        _system = system;
    }

    public FipaMobileAgentSystem getSystem()
    {
        return _system;
    }

    public void setLanguage(FipaMobileAgentLanguage language)
    {
        _language = language;
    }

    public FipaMobileAgentLanguage getLanguage()
    {
        return _language;
    }

    public void setOs(FipaMobileAgentOs os)
    {
        _os = os;
    }

    public FipaMobileAgentOs getOs()
    {
        return _os;
    }

    public boolean isEmpty()
    {
        return ((_system.isEmpty()) && (_language.isEmpty()) && (_os.isEmpty()));
    }

    public String toString()
    {
        return ( ((!_system.isEmpty()) ? (" <system>" + _system.toString() +
        " </system>") : "") +
        (" <language>" + _language.toString() + " </language>") + //language is mandatory
        ((!_os.isEmpty() ) ? (" <os>" + _os.toString() + " </os>") : "") );
```

```
    }
}
```

```
/**
  * Java object of the fipa-mobile-agent-system.
  * <br>
  * @author Milla Makelainen, 2000
  */

package mobility.ont.fipaman;

public class FipaMobileAgentSystem
{
    private String _name = null;
    private String _major_version = null;
    private String _minor_version = null;
    private String _dependencies = null;

  /**
    * Default constructor, instanciates all the instance variables.
    */
    public FipaMobileAgentSystem()
    {
        _name = new String();
        _major_version = new String();
        _minor_version = new String();
        _dependencies = new String();
    }

//--------------------------------------------------------------------------//
// GET/SET functions                                                        //
//--------------------------------------------------------------------------//
    public void setName(String name)
    {
        _name = name;
    }

    public String getName()
    {
        return _name;
    }

    public void setMajorVersion(String version)
    {
        _major_version = version;
    }

    public String getMajorVersion()
    {
        return _major_version;
    }

    public void setMinorVersion(String version)
    {
        _minor_version = version;
    }

    public String getMinorVersion()
    {
        return _minor_version;
    }

    public void setDependencies(String dependencies)
    {
        _dependencies = dependencies;
    }

    public String getDependencies()
    {
        return _dependencies;
    }
```

```
    public boolean isEmpty()
    {
        return (((_name == null) || (_name.equals(""))) &&
                ((_major_version == null) || (_major_version.equals(""))) &&
                ((_minor_version == null) || (_minor_version.equals(""))) &&
                ((_dependencies == null) || (_dependencies.equals(""))) );
    }

    public String toString()
    {
        return ( ((!_name.equals("")) ? (" <name>" + _name + "</name>") : "") +
            ((!_major_version.equals("")) ? (" <major-version>" + _major_version +
            "</major-version>") : "") +
            ((!_minor_version.equals("")) ? (" <minor-version>" + _minor_version +
            "</minor-version>") : "") +
            ((!_dependencies.equals("")) ? (" <dependencies>" + _dependencies +
            "</dependencies>") : "") );
    }

}
```

```
//----------------------------------------------------------------------------
//
// The Original Code is Nortel Networks' FIPA-OS (Foundation for Intelligent
// Physical Agents - Open Source).
//
// The Initial Developer of the Original Code is Nortel Networks Corporation.
// Portions created by Nortel Networks Corporation or its subsidiaries are
// Copyright (c) 1999 Nortel Networks Corporation. All Rights Reserved.
//
// Contributor: Milla Makelainen, 2000
//
//----------------------------------------------------------------------------

package aw.ont.profile;

import aw.ont.profile.ProfileObject;
import aw.agent.conversation.content.Content;
import aw.tools.DIAGNOSTICS;

import java.util.*;

 /**
 *
 * This class stores parameters related to the MMS object a mobile agent can use.
 * It is populated by automatic parsing of the Agent profile.
 * It will only be created and populated if an appropriate entry is made in the
 * profile.
 * The RDF Database object in the profile must have the same name including
 * capitalization as this class. The parser will rely on this to work out what
 * type of data object is to be populated.
 *
 */

public class MMS implements ProfileObject
{

//----------------------------------------------------------------------------
// CONSTANTS
//----------------------------------------------------------------------------

    public final static String NAME = "name";

//----------------------------------------------------------------------------
// INSTANCE VARIABLES
//----------------------------------------------------------------------------

   private String _mms_name = "";
   private String _profile_object_name = "";

//----------------------------------------------------------------------------
// CONSTRUCTOR
//----------------------------------------------------------------------------

  /**
   * Empty default constructor.
   */
   public MMS()
     {
     }

//----------------------------------------------------------------------------
// PRIVATE METHODS
//----------------------------------------------------------------------------

  /**
     *
     * Method used to get the text occuring after the # in the URL's
     * occuring in Triples.
```

```
     **/
     private String getType( String url )
     {
          if(url.indexOf("#") >= 0)
          {
             return url.substring( url.indexOf( '#' ) + 1 );
          }
          else
          {
             return new String(url);
          }
     }

//---------------------------------------------------------------------
// PUBLIC METHODS
//---------------------------------------------------------------------

   /**
     * This passes a Content object to the MMS object.
     * It then extracts the Data from the Content object and populate itself.
     * This method is specified by the "ProfileObject" interface.
     */
   public void populate(Content content)
     {
        // search through the various attributes set as Data in the Content object.
        // When one is identified that corresponds to a particular named parameter
        // of a Database object copy its value to the appropriate attribute held in
        // this object.

        // Content objects retrieved from a profile are named according to the instance
        // variable they are referenced with in the profile. An example of this might be
        // "mms1". This is set as being the name of this instance of the MMS
        // object.
        _profile_object_name = new String(content.getName());

        aw.tools.DIAGNOSTICS.println("MMS object name: " + _profile_object_name);

        aw.tools.DIAGNOSTICS.println("Details of Content object passed to
              \"populate\" method:");
        aw.tools.DIAGNOSTICS.println(content.toString());

        // retrieved a set of all the names of Attributes held in the Content object.
        aw.tools.DIAGNOSTICS.println("Attributes held in content object:");
        Set attributes = content.getAttributeNames();

        // form an iteration of these names
        Iterator iteration_of_names = attributes.iterator();

        while(iteration_of_names.hasNext())
        {
             // retrieve the name of the next attribute.
             String value_name = (String) iteration_of_names.next();
             aw.tools.DIAGNOSTICS.println("Attribute: " + value_name);

            // Retrieve the object that corresponds to this Attribute name.
            Object value = content.getAttribute(value_name);

           // Test to see what type of object "value" is.
           // It may be an instance of "Double", "Long", "String" or "Content".
             if(value instanceof Double)
             {
                   //shouldn't be
                aw.tools.DIAGNOSTICS.println("Value is a Double: " + value);
             }
             else if(value instanceof Long)
             {
                   //shouldn't be
                aw.tools.DIAGNOSTICS.println("Value is a Long: " + value);
             }
             else if(value instanceof String)
             {
```

```
                aw.tools.DIAGNOSTICS.println("Value is a String: " + value);
                String value_string = new String((String) value);
                value_name = getType(value_name);

                // Test to see whether the name of the Attribute matches
                // the attributes that may be stored in a Database object.

                if(value_name.equals(NAME))
                {
                   setMMSName(new String(value_string));
                }
                else
                {
                   aw.tools.DIAGNOSTICS.println("Attribute name does not correspond
                        to any Attribute");
                   aw.tools.DIAGNOSTICS.println("that may be set in a MMS object.");
                }

             }
             else if(value instanceof Content)
             {
                aw.tools.DIAGNOSTICS.println("Value is a Content object:");
                aw.tools.DIAGNOSTICS.println(new Content((Content)value).toString());
             }
             else
             {
                aw.tools.DIAGNOSTICS.println("Error. Value is not a recognised object.");

             }

             aw.tools.DIAGNOSTICS.println();

          } // while(iteration_of_names.hasNext())

       } // populate(Content content)


       /**
       * This allows the object to be interrogated to establish its name.
       * A profile data object will be named according to the instance variable
       * it was referenced with in the Agent profile.
       * This method is specified by the "ProfileObject" interface.
       */
       public String getName()
       {
            return new String(_profile_object_name);
       } // getName()

     /**
       * Sets the name of the MMS.
       * @param name The name of the MMS
       */
       public void setMMSName(String name)
       {
          _mms_name = new String(name);

       }


      /**
       * Returns the name of the MMS
       * @return String The name of the MMS
       */
       public String getMMSName()
       {
          return new String(_mms_name);

       }


}
```

```
//--------------------------------------------------------------------------
//
// The Original Code is Nortel Networks' FIPA-OS (Foundation for Intelligent
// Physical Agents - Open Source).
//
// The Initial Developer of the Original Code is Nortel Networks Corporation.
// Portions created by Nortel Networks Corporation or its subsidiaries are
// Copyright (c) 1999 Nortel Networks Corporation. All Rights Reserved.
//
// Contributor: Milla Makelainen, 2000
//
//--------------------------------------------------------------------------

package aw.ont.profile;

import aw.ont.profile.ProfileObject;
import aw.agent.conversation.content.Content;
import aw.tools.DIAGNOSTICS;

import java.util.*;

/**
 *
 * This class stores parameters related to the MobileAgentDescription an MMS should
 * use for storing information about the services it offers (from the user).
 * It is populated by automatic parsing of the Agent profile.
 * It will only be created and populated if an appropriate entry is made in the
 * profile.
 * The RDF MMS object in the profile must have the same name including
 * capitalization as this class. The parser will rely on this to work out what
 * type of data object is to be populated.
 *
 */

public class MobileAgentDescription implements ProfileObject
{

//--------------------------------------------------------------------------
// CONSTANTS
//--------------------------------------------------------------------------

    private final static String MOBILITY_PROTOCOL = "agent-mobility-protocol";

    private final static String SYSTEM_NAME = "system-name";
    private final static String SYSTEM_MAJOR = "system-major-version";
    private final static String SYSTEM_MINOR = "system-minor-version";
    private final static String SYSTEM_DEPENDENCY = "system-dependencies";

    private final static String LANGUAGE_NAME = "language-name";
    private final static String LANGUAGE_MAJOR = "language-major-version";
    private final static String LANGUAGE_MINOR = "language-minor-version";
    private final static String LANGUAGE_FORMAT = "language-format";

    private final static String OS_HARDWARE = "os-hardware";


//--------------------------------------------------------------------------
// INSTANCE VARIABLES
//--------------------------------------------------------------------------

   private String _protocol_name = "";
   private String _system_name = "";
   private String _system_major = "";
   private String _system_minor = "";
   private String _system_dependency = "";
   private String _language_name = "";
   private String _language_major = "";
   private String _language_minor = "";
```

```
   private String _language_format = "";
   private String _os_hardware = "";

   private String _profile_object_name = "";

//-------------------------------------------------------------------------
// CONSTRUCTOR
//-------------------------------------------------------------------------

   /**
   * Default empty constructor.
   */
   public MobileAgentDescription()
     {
     }

//-------------------------------------------------------------------------
// PRIVATE METHODS
//-------------------------------------------------------------------------

  /**
     * Method used to get the text occuring after the # in the URL's
     * occuring in Triples.
     **/
     private String getType( String url )
     {
          if(url.indexOf("#") >= 0)
          {
             return url.substring( url.indexOf( '#' ) + 1 );
          }
          else
          {
             return new String(url);
          }
     }

//-------------------------------------------------------------------------
// PUBLIC METHODS
//-------------------------------------------------------------------------

   /**
     * This passes a Content object to the MobileAgentDescription object.
     * It then extracts the Data from the Content object and populate itself.
     * This method is specified by the "ProfileObject" interface.
     */
   public void populate(Content content)
     {
        // search through the various attributes set as Data in the Content object.
        // When one is identified that corresponds to a particular named parameter
        // of a Database object copy its value to the appropriate attribute held in
        // this object.

        // Content objects retrieved from a profile are named according to the instance
        // variable they are referenced with in the profile.
        // This is set as being the name of this instance of the Database object.
        _profile_object_name = new String(content.getName());

        aw.tools.DIAGNOSTICS.println("MobileAgentDescription object name: " +
                                     _profile_object_name);

        aw.tools.DIAGNOSTICS.println("Details of Content object passed to \"populate\"
                                     method:");
        aw.tools.DIAGNOSTICS.println(content.toString());

        // retrieved a set of all the names of Attributes held in the Content object.
        aw.tools.DIAGNOSTICS.println("Attributes held in content object:");
        Set attributes = content.getAttributeNames();

        // form an iteration of these names
        Iterator iteration_of_names = attributes.iterator();
```

```
while(iteration_of_names.hasNext())
{
     // retrieve the name of the next attribute.
     String value_name = (String) iteration_of_names.next();
     aw.tools.DIAGNOSTICS.println("Attribute: " + value_name);

     // Retrieve the object that corresponds to this Attribute name.
   Object value = content.getAttribute(value_name);

   // Test to see what type of object "value" is.
   // It may be an instance of "Double", "Long", "String" or "Content".
     if ( (value instanceof String) || (value instanceof Long) ||
        (value instanceof Double) )
     {
        aw.tools.DIAGNOSTICS.println("Value is a String, Double or Long
            (converted into a string): " + value);
        String value_string = new String(""+ value);
        aw.tools.DIAGNOSTICS.println("value_string " + value_string);
        value_name = getType(value_name);

        // Test to see whether the name of the Attribute matches
        // the attributes that may be stored in a Database object.

        if(value_name.equals(SYSTEM_NAME))
        {
           setSystemName(new String(value_string));
        }
        else if(value_name.equals(SYSTEM_DEPENDENCY))
        {
           setSystemDependency(new String(value_string));
        }
        else if(value_name.equals(SYSTEM_MAJOR))
        {
           setSystemMajor(new String(value_string));
        }
        else if(value_name.equals(SYSTEM_MINOR))
        {
           setSystemMinor(new String(value_string));
        }
        else if(value_name.equals(LANGUAGE_FORMAT))
        {
           setLanguageFormat(new String(value_string));
        }
        else if(value_name.equals(LANGUAGE_NAME))
        {
           setLanguageName(new String(value_string));
        }
        else if(value_name.equals(LANGUAGE_MAJOR))
        {
           setLanguageMajor(new String(value_string));
        }
        else if(value_name.equals(LANGUAGE_MINOR))
        {
           setLanguageMinor(new String(value_string));
        }
        else if(value_name.equals(OS_HARDWARE))
        {
           setOsHardware(new String(value_string));
        }
        else if(value_name.equals(MOBILITY_PROTOCOL))
        {
           setMobilityProtocol(new String(value_string));
        }
        else
        {
           aw.tools.DIAGNOSTICS.println("Attribute name does not correspond
                                        to any Attribute");
           aw.tools.DIAGNOSTICS.println("that may be set in a Database object.");
        }

     }
```

```
            else if(value instanceof Content)
            {
               aw.tools.DIAGNOSTICS.println("Value is a Content object:");
               aw.tools.DIAGNOSTICS.println(new Content((Content)value).toString());
            }
            else
            {
               aw.tools.DIAGNOSTICS.println("Error. Value is not a recognised object.");

            }

            aw.tools.DIAGNOSTICS.println();

         } // while(iteration_of_names.hasNext())

      } // populate(Content content)


      /**
      * This allows the object to be interrogated to establish its name.
      * A profile data object will be named according to the instance variable
      * it was referenced with in the Agent profile.
      * This method is specified by the "ProfileObject" interface.
      */
      public String getName()
      {

           return new String(_profile_object_name);

      } // getName()

   /**
      * Set the protocol.
      * @param name the protocol name
      */
      public void setProtocolName(String name)
      {
         _protocol_name = new String(name);

      }

  /**
      * Get the protocol
      * @return the protocol name
      */
      public String getProtocolName()
      {
         return new String(_protocol_name);

      }

    /**
      * Set the system name
      * @param name the system name
      */
      public void setSystemName(String name)
      {
         _system_name = new String(name);

      }

      /**
      * Get the system name.
      * @return the system name
      */
      public String getSystemName()
      {
         return new String(_system_name);

      }
```

```
/**
  * Set the system major version.
  * @param name the system major version
  */
  public void setSystemMajor(String name)
  {
     _system_major = new String(name);

  }

  /**
  * Get the system major version.
  * @return system major version
  */
  public String getSystemMajor()
  {
     return new String(_system_major);

  }

/**
  * Set the system minor version.
  * @param name the system minor version
  */
  public void setSystemMinor(String name)
  {
     _system_minor = new String(name);

  }

  /**
  * Get the system minor version
  * @return the system minor version
  */
  public String getSystemMinor()
  {
     return new String(_system_minor);

  }

/**
  * Set the system dependencies.
  * @param name the system dependency
  */
  public void setSystemDependency(String name)
  {
     _system_dependency = new String(name);

  }

  /**
  * Get the system dependencies
  * @return the system dependencies
  */
  public String getSystemDependency()
  {
     return new String(_system_dependency);

  }

/**
  * Set the language name.
  * @param name the language name
  */
  public void setLanguageName(String name)
  {
     _language_name = new String(name);

  }

  /**
```

```
  * Get the language name.
  * @return the language name
  */
  public String getLanguageName()
  {
     return new String(_language_name);

  }

/**
  * Set the language major version.
  * @param name the language major version
  */
  public void setLanguageMajor(String name)
  {
     _language_major = new String(name);

  }

  /**
  * Get the language major version
  * @return the language major version
  */
  public String getLanguageMajor()
  {
     return new String(_language_major);

  }

/**
  * Set the language minor version.
  * @set name the language minor version
  */
  public void setLanguageMinor(String name)
  {
     _language_minor = new String(name);

  }

  /**
  * Get the language minor version.
  * @return the language minor version
  */
  public String getLanguageMinor()
  {
     return new String(_language_minor);

  }

/**
  * Set the language format.
  * @param name the language format
  */
  public void setLanguageFormat(String name)
  {
     _language_format = new String(name);

  }

  /**
  * Get the language format.
  * @return the language format
  */
  public String getLanguageFormat()
  {
     return new String(_language_format);

  }

/**
  * Set os hardware.
```

```
     * @param the os hardware
     */
     public void setOsHardware(String name)
     {
        _os_hardware = new String(name);

     }

     /**
     * Get the os hardware
     * @return the os hardware
     */
     public String getOsHardware()
     {
        return new String(_os_hardware);

     }

     /**
      * Set the mobility protocol.
      * @param name the mobility protocol
      */
     public void setMobilityProtocol(String name)
     {
        _protocol_name = new String(name);

     }

     /**
      * Get the mobility protocol.
      * @return the mobility protocol
      */
     public String getMobilityProtocol()
     {
        return new String(_protocol_name);

     }
}
```

```
/**
  * MobilityAgentDescriptionHandler parses the XML documents containing this
  * data.
  * <br>
  * @author Milla Makelainen, 2000
  */

package mobility.parser.xml;

import mobility.ont.fipaman.*;

import org.xml.sax.*;

import com.ibm.xml.parser.SAXDriver;

import java.util.Stack;
import java.io.*;

public class MobileAgentDescriptionHandler implements DocumentHandler, ErrorHandler
{
    private FipaMobileAgentDescription   _description   = null;
    private Stack                        _elements      = null;

    private final static String          AGENT_NAME         = "agent-name";
    private final static String          ADDRESS            = "address";
    private final static String          DESTINATION        = "destination";
    private final static String          NAME               = "name";
    private final static String          MAJOR_VERSION      = "major-version";
    private final static String          MINOR_VERSION      = "minor-version";
    private final static String          DEPENDENCIES       = "dependencies";
    private final static String          FORMAT             = "format";
    private final static String          FILTER             = "filter";
    private final static String          HARDWARE           = "hardware";
    private final static String AGENT_MOBILITY_PROTOCOL = "agent-mobility-protocol";
    private final static String          AGENT_CODE         = "agent-code";
    private final static String          AGENT_DATA         = "agent-data";
    private final static String          AGENT_VERSION      = "agent-version";
    private final static String          SIGNATURE          = "signature";
    private final static String          LANGUAGE           = "language";
    private final static String          SYSTEM             = "system";
    private final static String          OS                 = "os";

    private boolean _move = false;

 /**
   * Receive notification of ignorable whitespace in element content.
   * This method doesn't do anything.- whitespaces are ignored.
   *
   * <p>Validating Parsers must use this method to report each chunk
   * of ignorable whitespace (see the W3C XML 1.0 recommendation,
   * section 2.10): non-validating parsers may also use this method
   * if they are capable of parsing and using content models.</p>
   *
   * <p>SAX parsers may return all contiguous whitespace in a single
   * chunk, or they may split it into several chunks; however, all of
   * the characters in any single event must come from the same
   * external entity, so that the Locator provides useful
   * information.</p>
   *
   * <p>The application must not attempt to read from the array
   * outside of the specified range.</p>
   *
   * @param ch The characters from the XML document.
   * @param start The start position in the array.
   * @param length The number of characters to read from the array.
   * @exception org.xml.sax.SAXException Any SAX exception, possibly
   *            wrapping another exception.
   * @see #characters
```

```
  */

   public void ignorableWhitespace(char[] array, int start, int finnish) {}

/**
  * Receive notification of the beginning of a document. Initialise all
  * instance variables.
  *
  * <p>The SAX parser will invoke this method only once, before any
  * other methods in this interface or in DTDHandler (except for
  * setDocumentLocator).</p>
  *
  * @exception org.xml.sax.SAXException Any SAX exception, possibly
  *            wrapping another exception.
  */

   public void startDocument()
   {
       _description = new FipaMobileAgentDescription();
       _elements = new Stack();

       print("Start parsing XML document");
   }

/**
  * Receive notification of the beginning of an element. Put the element
  * to a stack to wait for the characters.
  *
  * <p>The Parser will invoke this method at the beginning of every
  * element in the XML document; there will be a corresponding
  * endElement() event for every startElement() event (even when the
  * element is empty). All of the element's content will be
  * reported, in order, before the corresponding endElement()
  * event.</p>
  *
  * <p>If the element name has a namespace prefix, the prefix will
  * still be attached.  Note that the attribute list provided will
  * contain only attributes with explicit values (specified or
  * defaulted): #IMPLIED attributes will be omitted.</p>
  *
  * @param name The element type name.
  * @param atts The attributes attached to the element, if any.
  * @exception org.xml.sax.SAXException Any SAX exception, possibly
  *            wrapping another exception.
  * @see #endElement
  * @see org.xml.sax.AttributeList
  */

   public void startElement(String name, AttributeList atts)
   {
       _elements.push( name );

       if ( name.equals("move") || name.equals("move-state") )
       {
           _move = true;
       }
   }

/**
  * Receive notification of character data.
  *
  * <p>The Parser will call this method to report each chunk of
  * character data.  SAX parsers may return all contiguous character
  * data in a single chunk, or they may split it into several
  * chunks; however, all of the characters in any single event
  * must come from the same external entity, so that the Locator
  * provides useful information.</p>
  *
  * <p>The application must not attempt to read from the array
  * outside of the specified range.</p>
  *
```

```
* <p>Note that some parsers will report whitespace using the
* ignorableWhitespace() method rather than this one (validating
* parsers must do so).</p>
*
* @param ch The characters from the XML document.
* @param start The start position in the array.
* @param length The number of characters to read from the array.
* @exception org.xml.sax.SAXException Any SAX exception, possibly
*            wrapping another exception.
* @see #ignorableWhitespace
* @see org.xml.sax.Locator
*/

public void characters(char[] ch, int start, int length)
{
    //All the fipa-mobile-agent-description tags
    if ( _elements.peek().equals(AGENT_NAME) )
    {
        _description.setAgentName( new String(ch, start, length) );
    }

    if ( _elements.peek().equals(ADDRESS) )
    {
        _description.setAddress( new String(ch, start, length) );
    }

    if ( _elements.peek().equals(DESTINATION) )
    {
        _description.setDestination( new String(ch, start, length) );
    }

    if ( _elements.peek().equals(AGENT_MOBILITY_PROTOCOL) )
    {
        _description.setAgentMobilityProtocol( new String(ch, start, length) );
    }

    if ( _elements.peek().equals(AGENT_CODE) )
    {
        _description.setAgentCode( new String(ch, start, length) );
    }

    if ( _elements.peek().equals(AGENT_DATA) )
    {
        _description.setAgentData( new String(ch, start, length) );
    }

    if ( _elements.peek().equals(AGENT_VERSION) )
    {
        _description.setAgentVersion( new String(ch, start, length) );
    }

    if ( _elements.peek().equals(SIGNATURE) )
    {
        _description.setSignature( new String(ch, start, length) );
    }

    //fipa-mobile-agent-profile
    if ( _elements.peek().equals(NAME) )
    {
        String temp = (String) _elements.pop();

        if ( _elements.peek().equals(LANGUAGE) )
        {
            FipaMobileAgentProfile profile = _description.getAgentProfile();
            FipaMobileAgentLanguage language = profile.getLanguage();
            language.setName( new String(ch, start, length) );
            profile.setLanguage(language);
            _description.setAgentProfile( profile );
        }
        if ( _elements.peek().equals(SYSTEM) )
        {
```

```
            FipaMobileAgentProfile profile = _description.getAgentProfile();
            FipaMobileAgentSystem system = profile.getSystem();
            system.setName( new String(ch, start, length) );
            profile.setSystem(system);
            _description.setAgentProfile( profile );
        }
        if ( _elements.peek().equals(OS) )
        {
            FipaMobileAgentProfile profile = _description.getAgentProfile();
            FipaMobileAgentOs os = profile.getOs();
            os.setName( new String(ch, start, length) );
            profile.setOs(os);
            _description.setAgentProfile( profile );
        }

        _elements.push( temp );
    }

    if ( _elements.peek().equals(MAJOR_VERSION) )
    {
        String temp = (String) _elements.pop();

        if ( _elements.peek().equals(LANGUAGE) )
        {
            FipaMobileAgentProfile profile = _description.getAgentProfile();
            FipaMobileAgentLanguage language = profile.getLanguage();
            language.setMajorVersion( new String(ch, start, length) );
            profile.setLanguage(language);
            _description.setAgentProfile( profile );
        }
        if ( _elements.peek().equals(SYSTEM) )
        {
            FipaMobileAgentProfile profile = _description.getAgentProfile();
            FipaMobileAgentSystem system = profile.getSystem();
            system.setMajorVersion( new String(ch, start, length) );
            profile.setSystem(system);
            _description.setAgentProfile( profile );
        }
        if ( _elements.peek().equals(OS) )
        {
            FipaMobileAgentProfile profile = _description.getAgentProfile();
            FipaMobileAgentOs os = profile.getOs();
            os.setMajorVersion( new String(ch, start, length) );
            profile.setOs(os);
            _description.setAgentProfile( profile );
        }

        _elements.push( temp );
    }

    if ( _elements.peek().equals(MINOR_VERSION) )
    {
        String temp = (String) _elements.pop();

        if ( _elements.peek().equals(LANGUAGE) )
        {
            FipaMobileAgentProfile profile = _description.getAgentProfile();
            FipaMobileAgentLanguage language = profile.getLanguage();
            language.setMinorVersion( new String(ch, start, length) );
            profile.setLanguage(language);
            _description.setAgentProfile( profile );
        }
        if ( _elements.peek().equals(SYSTEM) )
        {
            FipaMobileAgentProfile profile = _description.getAgentProfile();
            FipaMobileAgentSystem system = profile.getSystem();
            system.setMinorVersion( new String(ch, start, length) );
            profile.setSystem(system);
            _description.setAgentProfile( profile );
        }
        if ( _elements.peek().equals(OS) )
```

```
    {
        FipaMobileAgentProfile profile = _description.getAgentProfile();
        FipaMobileAgentOs os = profile.getOs();
        os.setMinorVersion( new String(ch, start, length) );
        profile.setOs(os);
        _description.setAgentProfile( profile );
    }

    _elements.push( temp );
}

if ( _elements.peek().equals(DEPENDENCIES) )
{
    String temp = (String) _elements.pop();

    if ( _elements.peek().equals(SYSTEM) )
    {
        FipaMobileAgentProfile profile = _description.getAgentProfile();
        FipaMobileAgentSystem system = profile.getSystem();
        system.setDependencies( new String(ch, start, length) );
        profile.setSystem(system);
        _description.setAgentProfile( profile );
    }
    if ( _elements.peek().equals(LANGUAGE) )
    {
        FipaMobileAgentProfile profile = _description.getAgentProfile();
        FipaMobileAgentLanguage language = profile.getLanguage();
        language.setDependencies( new String(ch, start, length) );
        profile.setLanguage(language);
        _description.setAgentProfile( profile );
    }
    if ( _elements.peek().equals(OS) )
    {
        FipaMobileAgentProfile profile = _description.getAgentProfile();
        FipaMobileAgentOs os = profile.getOs();
        os.setDependencies( new String(ch, start, length) );
        profile.setOs(os);
        _description.setAgentProfile( profile );
    }

    _elements.push( temp );
}

//language only
if ( _elements.peek().equals(FORMAT) )
{
    FipaMobileAgentProfile profile = _description.getAgentProfile();
    FipaMobileAgentLanguage language = profile.getLanguage();
    language.setFormat( new String(ch, start, length) );
    profile.setLanguage(language);
    _description.setAgentProfile( profile );
}

if ( _elements.peek().equals(FILTER) )
{
    FipaMobileAgentProfile profile = _description.getAgentProfile();
    FipaMobileAgentLanguage language = profile.getLanguage();
    language.setFilter( new String(ch, start, length) );
    profile.setLanguage(language);
    _description.setAgentProfile( profile );
}

//os only
if ( _elements.peek().equals(HARDWARE) )
{
    FipaMobileAgentProfile profile = _description.getAgentProfile();
    FipaMobileAgentOs os = profile.getOs();
    os.setHardware( new String(ch, start, length) );
    profile.setOs(os);
    _description.setAgentProfile( profile );
}
```

```
    }

    /**
     * StartElement() event for every endElement() event (even when the
   * element is empty).</p>
   *
   * <p>If the element name has a namespace prefix, the prefix will
   * still be attached to the name.</p>
   *
   * @param name The element type name
   * @exception org.xml.sax.SAXException Any SAX exception, possibly
   *            wrapping another exception.
   */

    public void endElement(String name)
    {
        _elements.pop();
    }

 /**
   * Receive notification of the end of a document.
   *
   * <p>The SAX parser will invoke this method only once, and it will
   * be the last method invoked during the parse.  The parser shall
   * not invoke this method until it has either abandoned parsing
   * (because of an unrecoverable error) or reached the end of
   * input.</p>
   *
   * @exception org.xml.sax.SAXException Any SAX exception, possibly
   *            wrapping another exception.
   */

    public void endDocument()
    {
        print("End XML document");
    }

    /**
   * Receive notification of a processing instruction. Not implemented.
   *
   * @param target The processing instruction target.
   * @param data The processing instruction data, or null if
   *        none was supplied.
   * @exception org.xml.sax.SAXException Any SAX exception, possibly
   *            wrapping another exception.
   */

  public void processingInstruction(String target, String data) {}

  /**
   * Receive an object for locating the origin of SAX document events.
   * Not implemented.
   *
   * @param locator An object that can return the location of
   *                any SAX document event.
   * @see org.xml.sax.Locator
   */

    public void setDocumentLocator(Locator locator) {}

 /**
   * Receive notification of a warning.
   *
   * <p>SAX parsers will use this method to report conditions that
   * are not errors or fatal errors as defined by the XML 1.0
   * recommendation.  The default behaviour is to take no action.</p>
   *
   * <p>The SAX parser must continue to provide normal parsing events
   * after invoking this method: it should still be possible for the
   * application to process the document through to the end.</p>
   *
```

```
  * @param exception The warning information encapsulated in a
  *                  SAX parse exception.
  * @exception org.xml.sax.SAXException Any SAX exception, possibly
  *            wrapping another exception.
  * @see org.xml.sax.SAXParseException
  */

   public void warning(SAXParseException exception)
   {
       print("Warning: " + exception);
   }

/**
  * Receive notification of a recoverable error.
  *
  * <p>This corresponds to the definition of "error" in section 1.2
  * of the W3C XML 1.0 Recommendation.  For example, a validating
  * parser would use this callback to report the violation of a
  * validity constraint.  The default behaviour is to take no
  * action.</p>
  *
  * <p>The SAX parser must continue to provide normal parsing events
  * after invoking this method: it should still be possible for the
  * application to process the document through to the end.  If the
  * application cannot do so, then the parser should report a fatal
  * error even if the XML 1.0 recommendation does not require it to
  * do so.</p>
  *
  * @param exception The error information encapsulated in a
  *                  SAX parse exception.
  * @exception org.xml.sax.SAXException Any SAX exception, possibly
  *            wrapping another exception.
  * @see org.xml.sax.SAXParseException
  */

   public void error(SAXParseException exception)
   {
       print("Recoverable error: " + exception);
   }

/**
  * Receive notification of a non-recoverable error.
  *
  * <p>This corresponds to the definition of "fatal error" in
  * section 1.2 of the W3C XML 1.0 Recommendation.  For example, a
  * parser would use this callback to report the violation of a
  * well-formedness constraint.</p>
  *
  * <p>The application must assume that the document is unusable
  * after the parser has invoked this method, and should continue
  * (if at all) only for the sake of collecting addition error
  * messages: in fact, SAX parsers are free to stop reporting any
  * other events once this method has been invoked.</p>
  *
  * @param exception The error information encapsulated in a
  *                  SAX parse exception.
  * @exception org.xml.sax.SAXException Any SAX exception, possibly
  *            wrapping another exception.
  * @see org.xml.sax.SAXParseException
  */

   public void fatalError(SAXParseException exception) throws SAXParseException
   {
       print("FATAL ERROR: " + exception);
       throw exception;
   }

/**
  * Return the object created.
  *
  * @return the mobile agent description
```

```
*/
 public FipaMobileAgentDescription getDescription()
 {
     return _description;
 }

  /**
* Has the act been move.
*
* @return boolean - if the act has been claimed
*/

 public boolean isMove()
 {
     return _move;
 }

 /**
  * Debug print method. Prints out the name of the class first - useful
  * when two or more agents are running on the same VM.
  * @param line The line to be printed out.
  */

 private void print(String line)
 {
     System.out.println(this.getClass().toString().substring(6) + " | " + line);
 }}
```

```
/**
  * MobilityManagementSystem is delegate agent of the AMS. Agents can request
  * to be moved to a remote location, MMS can perform this move.
  * <br>
  * @author Milla Makelainen, 2000
  */

package mobility;

import aw.agent.*;
import aw.agent.conversation.*;
import aw.agent.conversation.protocol.FIPARequest;
import aw.parser.acl.*;
import aw.fipa.*;
import aw.ont.profile.MobileAgentDescription;

import mobility.util.*;
import mobility.ont.fipaman.*;
import mobility.parser.xml.MobileAgentDescriptionHandler;

import java.io.*;
import java.net.InetAddress;

public class MobilityManagementSystem extends AgentWorldAgent
{

    //MMS can only move this agent
    private final static String SEARCH_AGENT_CODE = "mobility.SearchAgent";

    //mobile agent description from the agent profile
    private MobileAgentDescription _agent_description = null;

    //platform profile location
    private String _platform_profile_location = null;

    /**
   * The main entry point of the program. Start the program with two parameters:
   * the platform profile location and the name of the agent.
   */

    public static void main (String[] args)
    {
        MobilityManagementSystem mms = new MobilityManagementSystem(args[0], args[1],
                                          "Milla", true);
    }

    /**
     * Basic AWA constructor.
     *
     * @param platform_profile_location location of the platform profile
     * @param agent_guid globally unique identifier
     * @param ownership the ownership of the agent
     * @param using_push_method boolean selecting message passing method
     */
    public MobilityManagementSystem(String platform_profile_location, String agent_name,
                                     String ownership, boolean using_push_method)
    {
        super( platform_profile_location, agent_name, ownership, using_push_method,
                                  true, true);

        _platform_profile_location = platform_profile_location;

        //start receiving messages
        startPushing();

        //get the mobile agent data from the profile
        _agent_description = (MobileAgentDescription)
                this.getProfile().getNonStandards().get("description");
```

```
        print( "************************************" );
        print( "*  MOBILITY MANAGEMENT SYSTEM READY  *" );
        print( "************************************" );
        print( "MMS name is " + this.getGUID() );
        print( "Supported mobility protocol: " +
                        _agent_description.getMobilityProtocol() );
        print( "Supported mobility system: " + _agent_description.getSystemName() );
        print( "Supported system major version: " +
                        _agent_description.getSystemMajor() );
        print( "Supported system minor version: " +
                        _agent_description.getSystemMinor() );
        print( "Supported system dependencies: " +
                        _agent_description.getSystemDependency()  );
        print( "Supported language: " + _agent_description.getLanguageName() );
        print( "Supported language major version: "+
                        _agent_description.getLanguageMajor() );
        print( "Supported language minor version: "+
                        _agent_description.getLanguageMinor() );
        print( "Supported language format: " + _agent_description.getLanguageFormat() );
        print( "Supported hardware: " + _agent_description.getOsHardware() );
    }

    /**
     * This method overrides the notify method in AgentWorldAgent.class
     * This would be called when there is a new message received and conversation
     * manager decides that there is action required or when a coversation is finished
     *
     * @param conv The conversation that there is a new message recevied or it has been
     * finished.
     */

    public void notify( Conversation conv )
    {
        // if the message is not sent by this agent
        if
(!conv.getMessage(conv.getLatestMessageIndex()).getSender().equals(this.getGUID()))
        {
            // handle the message according to the message type
            try
            {
                ACLMessage acl = conv.getMessage(conv.getLatestMessageIndex());

                if (acl.getMessageType().equals("request"))
                {
                    handleRequest(conv);
                }
                else if (acl.getMessageType().equals("inform"))
                { //do nothing
                }
                else if (acl.getMessageType().equals("failure"))
                { //do nothing
                }
                else if (acl.getMessageType().equals("refuse"))
                { //do nothing
                }
                else if (acl.getMessageType().equals("agree"))
                { //do nothing
                }
                else if (acl.getMessageType().equals("not-understood"))
                { //do nothing
                }
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
        }
        // the conversation has come to an end
        else if (conv.getState() == Conversation.CONVERSATION_END) {
            print("conversation with id "+
```

```
    conv.getMessage(conv.getLatestMessageIndex()).getConversationID()+" has
finished!! ");
    }
}

/**
 * When agent receives a request message, it's handled here. Normally it would
 * be request to move the agent to a different location.
 *
 */

private void handleRequest(Conversation conv)
{
    print("\"request\" message received");

    ACLMessage acl = conv.getMessage(conv.getLatestMessageIndex());
    String content = acl.getContent();

    //message is possible in brackets
    if ( content.startsWith("(") )
    {
        if ( content.endsWith(")") )
        {
            content = content.substring(1, content.length()-1);
        }
    }

    //parse message
    SAXDriver parser = new SAXDriver();
    MobileAgentDescriptionHandler handler = new MobileAgentDescriptionHandler();

    parser.setDocumentHandler( handler );
    parser.setErrorHandler( handler );
    try
    {
        parser.parse(new InputSource(new StringReader(content)));

        //if it's a move message
        if ( handler.isMove() )
        {
            print("Message is request for move!");

            //get the parsed message as an object
            FipaMobileAgentDescription description = handler.getDescription();

            //check that the required service exists!
            //compare to the info given in the agent profile
            boolean all_in_order = true;

            //agent-profile must be defined
            FipaMobileAgentProfile profile = description.getAgentProfile();
            if ( !profile.isEmpty() )
            {
                print("\tProfile provided");

                //system important! must be FIPA-OS
                FipaMobileAgentSystem system = profile.getSystem();
                if ( !system.isEmpty() && all_in_order )
                {
                    //system name
                    print("\tSystem provided");
                    if ( ( system.getName().equals("") ||
                    system.getName().equalsIgnoreCase("FIPA-OS") ) && all_in_order )
                    {
                        print("\t\t- name " + system.getName() + " accepted");
                    }
                    else
                    {
                        print("Wrong system name");
                        if ( all_in_order )
                        {
```

```
                sendRefuse(conv, "agent-profile-unsupported");
            }
            all_in_order = false;
        }
        //system major version
        //FUTURE CHANGE: handle as numbers?
        if ( ( system.getMajorVersion().equals("") ||
            system.getMajorVersion().equalsIgnoreCase(
        _agent_description.getSystemMajor() ) ) && all_in_order )
        {
            print("\t\t- major-version " + system.getMajorVersion() +
                        " accepted");
        }
        else
        {
            print("Wrong major vesion (" + system.getMajorVersion()
                    + ") - version supported is " +
                _agent_description.getSystemMajor());
            if ( all_in_order )
            {
                sendRefuse(conv, "agent-profile-unsupported");
            }
            all_in_order = false;
        }
        //system minor version
        //FUTURE CHANGE: handle as numbers?
        if ( ( system.getMinorVersion().equals("") ||
                system.getMinorVersion().equalsIgnoreCase(_agent_descripti
            on.getSystemMinor()) ) && all_in_order )
        {
            print("\t\t- minor-version " + system.getMinorVersion()
                + " accepted");
        }
        else
        {
            print("Wrong minor-version (" + system.getMinorVersion() +
                     ") - version supported is " +
                _agent_description.getSystemMinor());
            if ( all_in_order )
            {
                sendRefuse(conv, "agent-profile-unsupported");
            }
            all_in_order = false;
        }
        //system dependencies
        if ( ( system.getDependencies().equals("") ||
                system.getDependencies().equalsIgnoreCase(_agent_descripti
            on.getSystemDependency()) ) && all_in_order )
        {
            print("\t\t- dependency " + system.getDependencies() +
                " accepted");
        }
        else
        {
            print("Wrong dependencies (" + system.getDependencies() +
            ") - version supported is " +
        _agent_description.getSystemDependency());
            if ( all_in_order )
            {
                sendRefuse(conv, "agent-profile-unsupported");
            }
            all_in_order = false;
        }
    }

    //language must be Java, version 1.2
    FipaMobileAgentLanguage language = profile.getLanguage();
    if ( !language.isEmpty() && all_in_order )
    {
        print("\tLanguage provided");
        if ( ( language.getName().equals("") ||
```

```
        language.getName().equalsIgnoreCase(_agent_description.get
    LanguageName()) ) && all_in_order )
{
    print("\t\t- name " + language.getName() + " accepted");
}
else
{
    print("Wrong language name");
    if ( all_in_order )
    {
        sendRefuse(conv, "agent-profile-unsupported");
    }
    all_in_order = false;
}
//language major version
//FUTURE CHANGE: handle as numbers?
if ( ( language.getMajorVersion().equals("") ||
        language.getMajorVersion().equalsIgnoreCase(_agent_descrip
    tion.getLanguageMajor()) ) && all_in_order )
{
    print("\t\t- major-version " + language.getMajorVersion() +
        " accepted");
}
else
{
    print("Wrong major vesion (" + language.getMajorVersion() +
              ") - version supported is " +
           _agent_description.getLanguageMajor());
    if ( all_in_order )
    {
        sendRefuse(conv, "agent-profile-unsupported");
    }
    all_in_order = false;
}
//language minor version
//FUTURE CHANGE: handle as numbers?
if ( ( language.getMinorVersion().equals("") ||
        anguage.getMinorVersion().equalsIgnoreCase(_agent_descript
    ion.getLanguageMinor()) ) && all_in_order )
{
    print("\t\t- minor-version " + language.getMinorVersion()
        + " accepted");
}
else
{
    print("Wrong minor-version (" + language.getMinorVersion() +
           ") - version supported is " +
        _agent_description.getLanguageMinor());
    if ( all_in_order )
    {
        sendRefuse(conv, "agent-profile-unsupported");
    }
    all_in_order = false;
}
//language dependencies
if ( language.getDependencies().equals("") && all_in_order )
{
    print( "\t\t- no dependencies " );
}
else
{
    print("No language dependencies supported");
    if ( all_in_order )
    {
        sendRefuse(conv, "agent-profile-unsupported");
    }
    all_in_order = false;
}
//language format
if ( ( language.getFormat().equals("") ||
```

```
                    language.getFormat().equalsIgnoreCase(_agent_descriptio
                n.getLanguageFormat()) ) && all_in_order )
            {
                print("\t\t- format " + language.getFormat() + " accepted");
            }
            else
            {
                print("Wrong format (" + language.getFormat() + ") - version
                    supported is " + _agent_description.getLanguageFormat());
                if ( all_in_order )
                {
                    sendRefuse(conv, "agent-profile-unsupported");
                }
                all_in_order = false;
            }
        }

        //os doesn't matter because of Java

    }
    else
    {
        print("No profile provided");
        if ( all_in_order )
        {
            sendRefuse(conv, "agent-profile-unsupported");
        }
        all_in_order = false;
    }

    //protocol must be simple-migration-protocol
    String protocol = description.getAgentMobilityProtocol();

    if ( (protocol.equals("") ||
        protocol.equalsIgnoreCase(_agent_description.getMobilityProtocol()) )
    && all_in_order )
    {
        print("\tMobility protocol \"" + protocol + "\" supported");
    }
    else
    {
        print("Mobility protocol not supported");
        if ( all_in_order )
        {
            sendRefuse(conv, "unwilling-to-perform");
        }
        all_in_order = false;

    }

    //agent code must be mobility.SearchAgent
    String code = description.getAgentCode();

    if ( code.equalsIgnoreCase(SEARCH_AGENT_CODE) && all_in_order )
    {
        print("\tCode \"" + code + "\" supported");
    }
    else
    {
        print("Agent not supported");
        if ( all_in_order )
        {
            sendRefuse(conv, "unwilling-to-perform");
        }
        all_in_order = false;
    }

    //agent data that the agent needs - just check that it's there
    String data = description.getAgentData();

    if ( !code.equals("") && all_in_order )
```

```
{
    print("\tAgent data provided");
}
else
{
    print("Agent data missing");
    if ( all_in_order )
    {
        sendRefuse(conv, "unwilling-to-perform");
    }
    all_in_order = false;
}

//agent version doesn't matter
//no need for signature

//check that destination is achievable
//MMS can only move agent to a location where it is itself
String destination = description.getDestination();
try
{
    if ( ( destination.indexOf( InetAddress.getLocalHost().
            getHostName() ) > -1) && all_in_order )
    {
        print( "\tDestination " + destination + " achiavable" );
    }
    else
    {
        print( "Can't move to " + destination + ", local host is " +
            InetAddress.getLocalHost().getHostName() );
        if ( all_in_order )
        {
            sendRefuse(conv, "unwilling-to-perform");
        }
        all_in_order = false;
    }
}
catch ( java.net.UnknownHostException uhe )
{
    print("No hostname for this computer!");
    print( "Can't move to " + destination );
    if ( all_in_order )
    {
        sendRefuse(conv, "unwilling-to-perform");
    }
    all_in_order = false;
}

//everything is as needed
if ( all_in_order )
{
    print("Everything is in order so... ");
    sendAgree(conv);
    sendInform(conv);

    //agent to new state:
    //take old state (first character in the string)
    int new_state = Character.getNumericValue(
        description.getAgentData().charAt(0) );
    //add one to it
    new_state++;
    //set the new agent data by putting the new state
    //instead of the old ne
    description.setAgentData( new_state +
        description.getAgentData().substring(1) );
    //create new agent
    //in this case since only one agent is supported, constructor
    //can be used
    //FUTURE: search for suitable constructor and dynamically load the
    //agent
    SearchAgent agent = new SearchAgent(_platform_profile_location,
```

```
                    description.getAgentName(), this.getGUID(), true, description);
                //send a command to execute
                sendExecute(agent.getGUID());
            }
        }
    }
    catch ( Exception e )
    {
        print("Exception while parsing the content of incoming ACL message");
        e.printStackTrace();
        sendNotUnderstood(conv, "unrecognised-attribute-value");
    }


}

/**
 * Send agreement message in the conversation.
 * @param conv Current conversation
 */

private void sendAgree(Conversation conv)
{
    ACLMessage agr_msg = conv.getFilledInMessage();

    agr_msg.setMessageType( "agree" );
    agr_msg.setOntology( "fipa-mobile-agent-management" );
    agr_msg.setLanguage( "XML" );
    agr_msg.setReceiver(conv.getSender(conv.getLatestMessageIndex()));
    agr_msg.setInReplyTo(conv.getMessage(conv.getLatestMessageIndex()).
    getReplyWith());
    agr_msg.setContent(conv.getMessage(conv.getLatestMessageIndex()).getContent());
    forward(agr_msg);

    print("sent \"agree\"!");

}

/**
 * Send inform message - last message in the conversation.
 * @param conv Current conversation
 */

private void sendInform(Conversation conv)
{
    ACLMessage inf_msg = conv.getFilledInMessage();

    inf_msg.setMessageType( "inform" );
    inf_msg.setOntology( "fipa-mobile-agent-management" );
    inf_msg.setLanguage( "XML" );
    inf_msg.setReceiver(conv.getSender(conv.getLatestMessageIndex()));

inf_msg.setInReplyTo(conv.getMessage(conv.getLatestMessageIndex()).getReplyWith());
    inf_msg.setContent(conv.getMessage(conv.getLatestMessageIndex()).getContent());
    forward(inf_msg);

    print("Sent \"inform\"");
}

/**
 * Send refuse message. Agent understands the request, but refuses it.
 * @param conv Current conversation
 * @param error The error message
 */


private void sendRefuse(Conversation conv, String error)
{
    ACLMessage rfs_msg = conv.getFilledInMessage();

    rfs_msg.setMessageType( "refuse" );
```

```
    rfs_msg.setOntology( "fipa-mobile-agent-management" );
    rfs_msg.setLanguage( "XML" );
    rfs_msg.setReceiver( conv.getSender(conv.getLatestMessageIndex()) );
    rfs_msg.setInReplyTo(
            conv.getMessage(conv.getLatestMessageIndex()).getReplyWith() );
    rfs_msg.setContent( error );
    forward(rfs_msg);

    print( "Sent \"refuse\"!" );

}

/**
 * Send not-understod message. Agent can't parse the message.
 * @param conv Current conversation
 * @param error The error message
 */


private void sendNotUnderstood(Conversation conv, String error)
{
    ACLMessage nu_msg = conv.getFilledInMessage();

    nu_msg.setMessageType( "not-understood" );
    nu_msg.setOntology( "fipa-mobile-agent-management" );
    nu_msg.setLanguage( "XML" );
    nu_msg.setReceiver( conv.getSender(conv.getLatestMessageIndex()) );
    nu_msg.setInReplyTo( conv.getMessage(conv.getLatestMessageIndex()).
                          getReplyWith() );
    nu_msg.setContent( error );
    forward(nu_msg);

    print( "Sent \"not-understood\"!" );

}

/**
 * Send execute message to an agent who has just been moved.
 * @param agent Name of the agent message goes to.
 */

private void sendExecute(String agent)
{
    try
    {
        ACLMessage exe_msg = getNewConversation("fipa-request").getFilledInMessage();

        exe_msg.setMessageType( "request" );
        exe_msg.setOntology( "fipa-mobile-agent-management" );
        exe_msg.setLanguage( "SL" );
        exe_msg.setReceiver(agent);
        exe_msg.setContent("execute");
        forward(exe_msg);

        print("Sent \"execute\"");
    }
    catch (Exception e)
    {
        print("Problem with sending a message: ");
        e.printStackTrace();
    }
}

/**
 * Debug print method. Prints out the name of the class first - useful
 * when two or more agents are running on the same VM.
 * @param line The line to be printed out.
 */

private void print(String line)
{
```

```
            System.out.println(this.getClass().toString().substring(6) + " | " + line);
        }
    }
```

Only one line has been changed – to save space only that is shown (from `private void assignNewObject(Content content)`, last line):

```
            _profile_objects.put(profile_object.getName(), profile_object);
```

```
/**
  * SearchAgent is a prototype mobile agent. When started, it will send a request
  * to be moved. When arriving the destination, it will perform the search and
  * ask to be moved back home. When arriving, it will display the results and shut
  * down.
  * <br>
  * @author Milla Makelainen, 2000
  */

package mobility;

import aw.agent.*;
import aw.fipa.AgentGUID;
import aw.agent.conversation.*;
import aw.parser.acl.*;
import aw.ont.profile.MMS;

import mobility.util.*;
import mobility.ont.fipaman.FipaMobileAgentDescription;

import java.io.*;
import java.util.StringTokenizer;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class SearchAgent extends AgentWorldAgent
{
    //text area on the search GUI
    private JTextArea                _text_area = null;
    //label on search GUI telling how many files have been found
    private JLabel                   _label2    = null;
    //variable containing number of files found
    private int                      _files_found= 0;

    //constant text for _label2
    private static final String LABEL2TEXT   = "Files found so far: ";
    //standard protocol used
    private static final String PROTOCOL     = "fipa-request";
    //constants for the agent states
    private static final int      STATIONARY = 0;
    private static final int      START      = 1;
    private static final int      MOVED      = 2;
    private static final int      FINISHING  = 3;

    //directory the search is going to be done
    private String                           _dir           = null;
    //string to be searched
    private String                           _file          = null;
    //agents current state - default is that agent is not mobile
    private int                              _state         = STATIONARY;
    //name of the agents home host
    private String                           _home          = null;

    //agents mobile agent description
    private FipaMobileAgentDescription _des = null;

    /**
     * AWA constructor called by MMS.
     *
     * @param platform_profile_location location of the platform profile
     * @param agent_guid globally unique identifier
     * @param ownership the ownership of the agent
     * @param using_push_method boolean selecting message passing method
     * @param description fipa-mobile-agent-desciprion
     */
    public SearchAgent(String platform_profile_location, String agent_name,
                       String ownership, boolean using_push_method,
```

```
                    FipaMobileAgentDescription description)
{
    super( platform_profile_location, agent_name, ownership, using_push_method,
           true, false);

    //parse agent data field - relies on the order of data
    StringTokenizer st = new StringTokenizer(description.getAgentData());
    _state = st.hasMoreTokens() ? (new Integer(st.nextToken()).intValue()) : 0;
    _dir = st.hasMoreTokens() ? st.nextToken() : null;
    _file = st.hasMoreTokens() ? st.nextToken() : null;
    _home = st.hasMoreTokens() ? st.nextToken() : null;

    _des = description;

    //start receiving messages
    startPushing();

    //if this is the first time agent is started
    if ( _state == START )
    {
        //get the remote destination MMS name from the agent profile
        String mms_name = ( (MMS) this.getProfile().getNonStandards().
                              get("mms2") ).getMMSName();
        //send move message to that MMS
        sendMove( mms_name + "@" + new AgentGUID(this.getGUID()).getAddress(),
                  description.getDestination() );
    }

    //print out the state
    print( "My state is " + _state );
}

/**
 * Stationary AWA constructor.
 *
 * @param platform_profile_location location of the platform profile
 * @param agent_guid globally unique identifier
 * @param ownership the ownership of the agent
 * @param using_push_method boolean selecting message passing method
 */
public SearchAgent(String platform_profile_location, String agent_name,
                   String ownership, boolean using_push_method )
{
    super( platform_profile_location, agent_name, ownership, using_push_method,
           true, false);

    //do the search
    searchGUI(_dir, _file);
    //shut down
    super.shutdown();
}

/**
 * Method that searches a directory for a file.
 *
 * @param directory_to_be_searched directory to be searched
 * @param file_to_be_searched file name to be searched
 */
public int searchForFile(String directory_to_be_searched,
                         String file_to_be_searched)
{
    int number_of_files = 0;  //number of files found matching the criteria

    //look into subfolders as well, return number of files found
    File file = new File(directory_to_be_searched);

    if ( !file.exists() || !file.isDirectory() )
    {
        //if it's not a directory, or doesn't exist
        if (!file.exists())
        {
```

```
                print("File " + directory_to_be_searched + " doesn't exists");
            }
            else
            {
                print(directory_to_be_searched + " isn't a directory");
            }
            return number_of_files;
        }

        //otherwise continue, since it's a directory and exists
        SearchFilter filter = new SearchFilter(file_to_be_searched);
        String[] file_list = file.list(filter);

        for (int i = 0; i<file_list.length; i++)
        {
            File temp_file = new File(directory_to_be_searched + File.separator +
                                      file_list[i]);

            //check if its a directory
            if ( temp_file.isDirectory() )
            {
                //if it is, recursivly search the directory
                number_of_files += searchForFile(temp_file.toString(),
                                                 file_to_be_searched);
            }
            else //it's a file and matches the criteria
            {
                //update the search GUI
                update( temp_file.getPath() );
                //one more file found
                number_of_files++;
            }
        }

        return number_of_files;
    }

    /**
     * This method overrides the notify method in AgentWorldAgent.class
     * This would be called when there is a new message received and conversation
     * manager decides that there is action required or when a coversation is finished
     *
     * @param conv The conversation that there is a new message recevied or it has been
     * finished.
     */

    public void notify( Conversation conv )
    {
        // if the message is not sent by this agent
        if
(!conv.getMessage(conv.getLatestMessageIndex()).getSender().equals(this.getGUID()))
        {
            // handle the message according to the message type
            try
            {
                ACLMessage acl = conv.getMessage(conv.getLatestMessageIndex());

                if (acl.getMessageType().equals("request"))
                {
                    handleRequest(conv);
                }
                else if (acl.getMessageType().equals("inform"))
                {
                    handleInform(conv);
                }
                else if (acl.getMessageType().equals("failure"))
                { //do nothing
                }
                else if (acl.getMessageType().equals("refuse"))
                { //do nothing
                }
```

```
            else if (acl.getMessageType().equals("agree"))
            { //do nothing
            }
            else if (acl.getMessageType().equals("not-understood"))
            { // do nothing
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }

    }
    // the conversation has come to an end
    else if (conv.getState() == Conversation.CONVERSATION_END) {
        print("conversation with id "
        + conv.getMessage(conv.getLatestMessageIndex()).getConversationID()+
        " has finished!! ");
    }

    print( "My state is " + _state );

}

/**
 * If agent receives a request, it should be the execute command.
 * Action depends on the agent state.
 *
 * @param conv The conversation the message belongs to.
 */
private void handleRequest(Conversation conv)
{
    ACLMessage acl = conv.getMessage(conv.getLatestMessageIndex());

    //if content is execute
    if (acl.getContent().equalsIgnoreCase("execute") )
    {
        print("Received execute command");
        switch ( _state )
        {
            case STATIONARY:      //shouldn't happen, since it doesn't receive
                                  //messages
                                  sendNotUnderstood( conv );
                                  break;
            case START:           //if it hasn't moved yet makes no sense
                                  sendNotUnderstood( conv );
                                  break;
            case MOVED:           //just moved, so start the GUI and the search
                                  sendAgree(conv);
                                  sendInform(conv);
                                  searchGUI(_dir, _file);
                                  searchForFile(_dir, _file);
                                  //once the search has been done, ask for move
                                  //MMS name from the agent platform
                                  String mms_address = ( (MMS)
                                  this.getProfile().getNonStandards().get("mms1") ).
                                  getMMSName();
                                  sendMove( mms_address + "@" +
                                   new AgentGUID(this.getGUID()).getAddress(),_home);
                                  break;
            case FINISHING:       //just come back - just display the results.
                                  sendAgree(conv);
                                  sendInform(conv);
                                  end(_file);
                                  shutdown();
                                  break;
            default:              //state is different to the defined ones
                                  print("Something went very wrong! Wrong state!");
                                  break;
        }
    }
```

```
    else //if the message isn't an execute
    {
        print( "Was expecting to get \"execute\", got \""
            + acl.getContent() + "\" instead" );
        sendNotUnderstood(conv);
    }
}

/**
 * If agent receives an inform, it move should have been accepted.
 * Action SHOULD depend on the agent state (not at the moment
 * - FUTURE work).
 *
 * @param conv The conversation the message belongs to.
 */
private void handleInform(Conversation conv)
{
    ACLMessage acl = conv.getMessage(conv.getLatestMessageIndex());

    print( "\"inform\" message received" );

    //check that content has <move-state> in it
    if ( acl.getContent().indexOf("<move-state>") > -1 )
    {
        //if so, shut down
        shutdown();
    }
    else
    {
        print( "content not understood: " + acl.getContent() );
        sendNotUnderstood( conv );
    }
}

/**
 * Sends request for move to specified destination from specified agent.
 * @param receiver The receiving MMS
 * @param destination The remote location (host name)
 */
private void sendMove( String receiver, String destination )
{
    //change the destination in the agent description
    _des.setDestination( destination );

    try
    {
        ACLMessage request = getNewConversation(PROTOCOL).getFilledInMessage();

        request.setMessageType("request");
        request.setSender(this.getGUID());
        request.setReceiver(receiver);
        //set content so that fipa-mobile-agent-description is wrapped inside
        //<action> command
        request.setContent("(<?xml version=\"1.0\"?>" +
                " <!DOCTYPE action SYSTEM \"description.dtd\">" +
                " <action to=\"mms\"> <move-state> " + _des +
                " </move-state> </action>)");
        request.setOntology("fipa-mobile-agent-management");
        request.setLanguage("XML");

        forward(request);
        print( this.getGUID() + " sent a message: " + request.toString() );
    }
    catch (UnknownProtocolException upe)
    {
        upe.printStackTrace();
    }

}

/**
```

```
 * Sends agree in a specified conversation.
 * @param conv The on-going conversation
 */
private void sendAgree( Conversation conv )
{
    ACLMessage agr_msg = conv.getFilledInMessage();

    agr_msg.setMessageType( "agree" );
    agr_msg.setOntology( "fipa-mobile-agent-management" );
    agr_msg.setLanguage( "SL" );
    agr_msg.setReceiver(conv.getSender(conv.getLatestMessageIndex()));
    agr_msg.setInReplyTo(conv.getMessage(conv.getLatestMessageIndex()).
            getReplyWith());
    agr_msg.setContent(conv.getMessage(conv.getLatestMessageIndex()).toString());
    forward(agr_msg);

    print("Sent \"agree\"");
}

/**
 * Sends inform in a specified conversation.
 * @param conv The on-going conversation
 */
private void sendInform( Conversation conv )
{
    ACLMessage inf_msg = conv.getFilledInMessage();

    inf_msg.setMessageType( "inform" );
    inf_msg.setOntology( "fipa-mobile-agent-management" );
    inf_msg.setLanguage( "SL" );
    inf_msg.setReceiver(conv.getSender(conv.getLatestMessageIndex()));
    inf_msg.setInReplyTo(conv.getMessage(conv.getLatestMessageIndex()).
            getReplyWith());
    inf_msg.setContent(conv.getMessage(conv.getLatestMessageIndex()).toString());
    forward(inf_msg);

    print("Sent \"inform\"");
}

/**
 * Sends not-understood in a specified conversation.
 * @param conv The on-going conversation
 */
private void sendNotUnderstood( Conversation conv )
{
    ACLMessage not_msg = conv.getFilledInMessage();

    not_msg.setMessageType( "not-understood" );
    not_msg.setOntology( "fipa-mobile-agent-management" );
    not_msg.setLanguage( "SL" );
    not_msg.setReceiver(conv.getSender(conv.getLatestMessageIndex()));
    not_msg.setInReplyTo(conv.getMessage(conv.getLatestMessageIndex()).
            getReplyWith());
    not_msg.setContent(conv.getMessage(conv.getLatestMessageIndex()).toString());
    forward(not_msg);

    print("Sent \"not-understood\"");
}

/**
 * The search GUI.
 * @param dir The directory to be searched
 * @param file The string to be searched
 */
private void searchGUI(String dir, String file)
{
    //new JFrame with panels
    JFrame frame = new JFrame("Agent " + this.getGUID() + " searching...");
    frame.setSize(new Dimension(300, 300));
    JPanel content_pane = new JPanel();
```

```
    frame.setContentPane(content_pane);
    content_pane.setLayout(new BorderLayout());
    content_pane.setBorder(BorderFactory.createEmptyBorder(2,2,2,2));

    frame.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        });
    //first label
    JLabel label1 = new JLabel("Searching " + dir + " for \"" + file + "\"...");
    label1.setOpaque(true);
    label1.setPreferredSize(new Dimension(10, 30));

    //text area, global so it can be updated from another method
    _text_area = new JTextArea("Initialising...\n");
    _text_area.setOpaque(true);

    JScrollPane area_scroll_pane = new JScrollPane(_text_area,
    ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,
    ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
    area_scroll_pane.setPreferredSize(new Dimension(250, 250));

    _label2 = new JLabel(LABEL2TEXT + _files_found);
    _label2.setOpaque(true);
    _label2.setPreferredSize(new Dimension(10, 30));

    content_pane.add(label1, BorderLayout.NORTH);
    content_pane.add(area_scroll_pane, BorderLayout.CENTER);
    content_pane.add(_label2, BorderLayout.SOUTH);

    frame.pack();
    frame.setVisible(true);

}

/**
 * This is the methdod that agents call when they want to update the GUI. It's
 * always called when a file is found so that found file name can be displayed.
 *
 * @param file The new file just found
 */

public void update(String file)
{
    _text_area.append(file + "\n");
    _label2.setText(LABEL2TEXT + ++_files_found);
}


/**
 * When search is finished, a dialog pops up telling the results.
 *
 */

public void end(String search_condition)
{
    JOptionPane.showMessageDialog(new JFrame(),
    "Search for \"" + search_condition + "\" finished, found " +
    _files_found + " files.",
    "Search finished",
    JOptionPane.INFORMATION_MESSAGE);
}

/**
 * Debug print method. Prints out the name of the class first - useful
 * when two or more agents are running on the same VM.
 * @param line The line to be printed out.
 */
```

```
    private void print(String line)
    {
        System.out.println(this.getClass().toString().substring(6) + " | " + line);
    }

}
```

```
/**
  * SearchFilter implements FilenameFileter interface: instances of classes that
  * implement this interface are used to filter filenames. These instances
  * are used to filter directory listings in the list method of class File,
  * and by the Abstract Window Toolkit's file dialog component. <br>
  * Used by SearchAgent.
  * <br>
  * @author Milla Makelainen, 2000
  */

package mobility.util;

import java.io.*;

public class SearchFilter implements FilenameFilter
{
    private String _search_condition = null;

    /**
    * Constructor setting the search condidition.
    *
    * @param search_condition the search condition
    */
    public SearchFilter(String search_condition)
    {
        _search_condition = search_condition.toLowerCase();
    }

    /**
     * Accepts files that contain the search condition and all directories.
     * @param dir directory, ignored
     * @param file the file/directory name
     * @return if conditions are met
     */
    public boolean accept(File dir, String file)
    {

        file = file.toLowerCase();

        if (file.indexOf(_search_condition) > -1)
        {
            return true;
        }
        else
        {
            if ( ( new File(dir + File.separator + file) ).isDirectory() )
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

```
/**
  * Starting GUI, the main starting point for the Search Agent scenario.
  * <br>
  * @author Milla Makelainen, 2000
  */

package mobility.gui;

import java.awt.*;
import java.awt.event.*;
import java.net.*;
import javax.swing.*;

import mobility.*;
import mobility.ont.fipaman.*;

import aw.agent.AgentProfile;

public class StartingGUI implements ActionListener
{
    private JFrame _detail_frame = null;
    private JTextField _field_search_directory = null;
    private JTextField _field_search_criteria = null;
    private JTextField _field_agent_name = null;
    private JTextField _field_address = null;
    private JTextField _field_destination = null;
    private JTextField _field_system_name = null;
    private JTextField _field_system_major_version = null;
    private JTextField _field_system_minor_version = null;
    private JTextField _field_system_dependencies = null;
    private JTextField _field_language_name = null;
    private JTextField _field_language_major_version = null;
    private JTextField _field_language_minor_version = null;
    private JTextField _field_language_format = null;
    private JTextField _field_language_filter = null;
    private JTextField _field_language_dependencies = null;
    private JTextField _field_os_name = null;
    private JTextField _field_os_major_version = null;
    private JTextField _field_os_minor_version = null;
    private JTextField _field_os_hardware = null;
    private JTextField _field_os_dependencies = null;
    private JTextField _field_agent_mobility_protocol = null;
    private JTextField _field_agent_code = null;
    private JTextField _field_agent_data = null;
    private JTextField _field_agent_version = null;
    private JTextField _field_signature = null;
    private JButton _button1 = null;
    private JButton _button2 = null;
    private String _last_action_command = null;
    private String _platform = null;
    private FipaMobileAgentDescription _description = null;
    private String _agent_data = null;
    private static String _profile = null;

  /**
   * The main entry point of the program. Start the program with one parameter:
   * the platform profile location.
   */

    public static void main (String[] args)
    {
        StartingGUI gui = new StartingGUI(args[0]);
    }

  /**
   * The constructor.
   *
   * @param platform the platform profile location
```

```
*/
 public StartingGUI(String platform)
 {
     _platform = platform;

     //new frame with panels using GridBag layout.
     JFrame frame = new JFrame("Start the search");
     frame.setSize(new Dimension(300, 300));
     JPanel content_pane = new JPanel();

     frame.setContentPane(content_pane);
     GridBagLayout layout = new GridBagLayout();
     GridBagConstraints c = new GridBagConstraints();
     content_pane.setLayout(layout);
     content_pane.setBorder(BorderFactory.createEmptyBorder(2,2,2,2));
     c.fill = GridBagConstraints.HORIZONTAL;
     c.insets = new Insets(2,2,2,2);

     frame.addWindowListener(new WindowAdapter()
         {
             public void windowClosing(WindowEvent e)
             {
                 System.exit(0);
             }
         });

     //setting labels and text fields
     JLabel label1 = new JLabel("Directory to be searched: ");
     c.weightx = 0.5;
     c.gridx = 0;
     c.gridy = 0;
     layout.setConstraints(label1, c);
     content_pane.add(label1);

     JLabel label2 = new JLabel("Search criteria: ");
     c.weightx = 0.5;
     c.gridx = 0;
     c.gridy = 1;
     layout.setConstraints(label2, c);
     content_pane.add(label2);

     _field_search_directory = new JTextField(20);
     c.weightx = 0.5;
     c.gridx = 1;
     c.gridy = 0;
     layout.setConstraints(_field_search_directory, c);
     content_pane.add(_field_search_directory);

     _field_search_criteria = new JTextField(20);
     c.weightx = 0.5;
     c.gridx = 1;
     c.gridy = 1;
     layout.setConstraints(_field_search_criteria, c);
     content_pane.add(_field_search_criteria);

     //buttons
     _button1 = new JButton("Start local agent");
     _button1.setActionCommand("Local");
     _button1.addActionListener(this);
     c.weightx = 0.0;
     c.gridx = 0;
     c.gridy = 2;
     layout.setConstraints(_button1, c);
     content_pane.add(_button1);

     _button2 = new JButton("Start mobile agent");
     _button2.setActionCommand("Mobile");
     _button2.addActionListener(this);
     c.weightx = 0.0;
     c.gridx = 1;
     c.gridy = 2;
```

```
        layout.setConstraints(_button2, c);
        content_pane.add(_button2);

        frame.pack();
        frame.setVisible(true);

    }

    /**
        * Invoked when an action occurs.
        *
        * @param e the ActionEvent
        */

  public void actionPerformed(ActionEvent e)
  {
    _last_action_command = e.getActionCommand();

    //if button pressed is local agent
    if (_last_action_command.equals("Local"))
    {
        System.out.println("Start local agent");
        SearchAgent agent = new SearchAgent( _platform, "searcher", "searcher", true );
        }
        else //if button pressed is mobile agent
        {
            if (e.getActionCommand().equals("Mobile"))
        {
            System.out.println("Start mobile agent");
            _agent_data = _field_search_directory.getText() + " " +
                          _field_search_criteria.getText();
            addMobileAgentDetail();
        }
        else //it's the mobile agent detail
            {
                if (e.getActionCommand().equals("Confirm agent details"))
                {
                    FipaMobileAgentDescription description = new
                                                        FipaMobileAgentDescription();
                    FipaMobileAgentProfile profile = new FipaMobileAgentProfile();
                    FipaMobileAgentSystem system = new FipaMobileAgentSystem();
                    FipaMobileAgentLanguage language = new FipaMobileAgentLanguage();
                    FipaMobileAgentOs os = new FipaMobileAgentOs();

                    //get the data from the form and put it into a new object
                    //system
                    system.setName(_field_system_name.getText());
                    system.setMajorVersion(_field_system_major_version.getText());
                    system.setMinorVersion(_field_system_minor_version.getText());
                    system.setDependencies(_field_system_dependencies.getText());

                    //language
                    language.setName(_field_language_name.getText());
                    language.setMajorVersion(_field_language_major_version.getText());
                    language.setMinorVersion(_field_language_minor_version.getText());
                    language.setFormat(_field_language_format.getText());
                    language.setFilter(_field_language_filter.getText());
                    language.setDependencies(_field_language_dependencies.getText());

                    //os
                    os.setName(_field_os_name.getText());
                    os.setMajorVersion(_field_os_major_version.getText());
                    os.setMinorVersion(_field_os_minor_version.getText());
                    os.setHardware(_field_os_hardware.getText());
                    os.setDependencies(_field_os_dependencies.getText());

                    //profile
                    profile.setSystem(system);
                    profile.setLanguage(language);
                    profile.setOs(os);
```

```
                //description
                description.setAgentName(_field_agent_name.getText());
                description.setAddress(_field_address.getText());
                description.setDestination(_field_destination.getText());
                description.setAgentProfile(profile);
                description.setAgentMobilityProtocol
                                    (_field_agent_mobility_protocol.getText());
                description.setAgentCode(_field_agent_code.getText());
                description.setAgentData(_field_agent_data.getText());
                description.setAgentVersion(_field_agent_version.getText());
                description.setSignature(_field_signature.getText());

                _description = description;
                _detail_frame.dispose();

                SearchAgent agent = new SearchAgent(_platform,
                description.getAgentName(), description.getAgentName(),
                true, description);

            //disable buttons so no agents can be started until this one has finished
            _button1.setEnabled(false);
            _button2.setEnabled(false);
            }
        }
}

}

/**
 * GUI for getting the mobile agent detail
 */
private void addMobileAgentDetail()
{
    //create new frame with panels usgin GridBag layout
    _detail_frame = new JFrame("Add agent detail");
    _detail_frame.setSize(new Dimension(300, 300));
    JPanel content_pane = new JPanel();

    _detail_frame.setContentPane(content_pane);
    GridBagLayout layout = new GridBagLayout();
    GridBagConstraints c = new GridBagConstraints();
    content_pane.setLayout(layout);
    content_pane.setBorder(BorderFactory.createEmptyBorder(2,2,2,2));
    c.fill = GridBagConstraints.HORIZONTAL;
    c.insets = new Insets(2,2,2,2);

    _detail_frame.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        });

    //add labels and text fields with default information
    JLabel label1 = new JLabel("Add necessery details:");
    c.weightx = 0.5;
    c.gridx = 0;
    c.gridy = 0;
    layout.setConstraints(label1, c);
    content_pane.add(label1);

    JLabel label2 = new JLabel(":agent-name");
    .weightx = 0.5;
    c.gridx = 0;
    c.gridy = 1;
    layout.setConstraints(label2, c);
    content_pane.add(label2);

    _field_agent_name = new JTextField("searcher");
    c.weightx = 0.5;
```

```
c.gridx = 1;
c.gridy = 1;
layout.setConstraints(_field_agent_name, c);
content_pane.add(_field_agent_name);

JLabel label3 = new JLabel(":address");
c.weightx = 0.5;
c.gridx = 0;
c.gridy = 2;
layout.setConstraints(label3, c);
content_pane.add(label3);

try
{
    String ams = ( new AgentProfile(_profile, "searcher") ).
                  getPlatformGUID().toString();
    _field_address = new JTextField( ams.substring(ams.indexOf("@")), 20 );
}
catch ( Exception e )
{
    _field_address = new JTextField( "iiop://localhost:9000/acc", 20 );
}
c.weightx = 0.5;
c.gridx = 1;
c.gridy = 2;
layout.setConstraints(_field_address, c);
content_pane.add(_field_address);

JLabel label4 = new JLabel(":destination");
c.weightx = 0.5;
c.gridx = 0;
c.gridy = 3;
layout.setConstraints(label4, c);
content_pane.add(label4);

_field_destination = new JTextField("", 20);
c.weightx = 0.5;
c.gridx = 1;
c.gridy = 3;
layout.setConstraints(_field_destination, c);
content_pane.add(_field_destination);

//agent profile
JPanel profile_content_pane = new JPanel();
GridBagLayout profile_layout = new GridBagLayout();
GridBagConstraints profile_c = new GridBagConstraints();
profile_c.fill = GridBagConstraints.HORIZONTAL;
profile_c.insets = new Insets(2,2,2,2);

profile_content_pane.setLayout(profile_layout);

profile_c.weightx = 0.5;
profile_content_pane.setBorder(
BorderFactory.createCompoundBorder(
       BorderFactory.createTitledBorder(":agent-profile"),
       BorderFactory.createEmptyBorder(5,5,5,5)));

//system
JPanel system_content_pane = new JPanel();
GridBagLayout system_layout = new GridBagLayout();
GridBagConstraints system_c = new GridBagConstraints();

system_content_pane.setLayout(system_layout);

system_c.fill = GridBagConstraints.HORIZONTAL;

system_c.weightx = 0.5;
system_content_pane.setBorder(
BorderFactory.createCompoundBorder(
       BorderFactory.createTitledBorder(":system"),
       BorderFactory.createEmptyBorder(5,5,5,5)));
```

```
JLabel label7 = new JLabel(":name");
system_c.weightx = 0.5;
system_c.gridx = 0;
system_c.gridy = 0;
system_layout.setConstraints(label7, system_c);
system_content_pane.add(label7);

_field_system_name = new JTextField("FIPA-OS", 10);
system_c.weightx = 0.5;
system_c.gridx = 1;
system_c.gridy = 0;
system_layout.setConstraints(_field_system_name, system_c);
system_content_pane.add(_field_system_name);

JLabel label8 = new JLabel(":major-version");
system_c.weightx = 0.5;
system_c.gridx = 0;
system_c.gridy = 1;
system_layout.setConstraints(label8, system_c);
system_content_pane.add(label8);

_field_system_major_version = new JTextField("1.03", 10);
system_c.weightx = 0.5;
system_c.gridx = 1;
system_c.gridy = 1;
system_layout.setConstraints(_field_system_major_version, system_c);
system_content_pane.add(_field_system_major_version);

JLabel label9 = new JLabel(":minor-version");
system_c.weightx = 0.5;
system_c.gridx = 0;
system_c.gridy = 2;
system_layout.setConstraints(label9, system_c);
system_content_pane.add(label9);

_field_system_minor_version = new JTextField("1.03", 10);
system_c.weightx = 0.5;
system_c.gridx = 1;
system_c.gridy = 2;
system_layout.setConstraints(_field_system_minor_version, system_c);
system_content_pane.add(_field_system_minor_version);

JLabel label10 = new JLabel(":dependencies");
system_c.weightx = 0.5;
system_c.gridx = 0;
system_c.gridy = 3;
system_layout.setConstraints(label10, system_c);
system_content_pane.add(label10);

_field_system_dependencies = new JTextField("MobilityManagementSystem", 10);
system_c.weightx = 0.5;
system_c.gridx = 1;
system_c.gridy = 3;
system_layout.setConstraints(_field_system_dependencies, system_c);
system_content_pane.add(_field_system_dependencies);

//put it in profile pane
profile_c.weightx = 0.5;
profile_c.gridx = 0;
profile_c.gridy = 0;
profile_layout.setConstraints(system_content_pane, profile_c);
profile_content_pane.add(system_content_pane);

//language
JPanel language_content_pane = new JPanel();
GridBagLayout language_layout = new GridBagLayout();
GridBagConstraints language_c = new GridBagConstraints();

language_content_pane.setLayout(language_layout);
```

```
language_c.fill = GridBagConstraints.HORIZONTAL;

language_c.weightx = 0.5;
language_content_pane.setBorder(
BorderFactory.createCompoundBorder(
        BorderFactory.createTitledBorder(":language"),
        BorderFactory.createEmptyBorder(5,5,5,5)));

JLabel label12 = new JLabel(":name");
language_c.weightx = 0.5;
language_c.gridx = 0;
language_c.gridy = 0;
language_layout.setConstraints(label12, language_c);
language_content_pane.add(label12);

_field_language_name = new JTextField("Java", 10);
language_c.weightx = 0.5;
language_c.gridx = 1;
language_c.gridy = 0;
language_layout.setConstraints(_field_language_name, language_c);
language_content_pane.add(_field_language_name);

JLabel label13 = new JLabel(":major-version");
language_c.weightx = 0.5;
language_c.gridx = 0;
language_c.gridy = 1;
language_layout.setConstraints(label13, language_c);
language_content_pane.add(label13);

_field_language_major_version = new JTextField("1.2.2", 10);
language_c.weightx = 0.5;
language_c.gridx = 1;
language_c.gridy = 1;
language_layout.setConstraints(_field_language_major_version, language_c);
language_content_pane.add(_field_language_major_version);

JLabel label14 = new JLabel(":minor-version");
language_c.weightx = 0.5;
language_c.gridx = 0;
language_c.gridy = 2;
language_layout.setConstraints(label14, language_c);
language_content_pane.add(label14);

_field_language_minor_version = new JTextField("1.2", 10);
language_c.weightx = 0.5;
language_c.gridx = 1;
language_c.gridy = 2;
language_layout.setConstraints(_field_language_minor_version, language_c);
language_content_pane.add(_field_language_minor_version);

JLabel label15 = new JLabel(":format");
language_c.weightx = 0.5;
language_c.gridx = 0;
language_c.gridy = 3;
language_layout.setConstraints(label15, language_c);
language_content_pane.add(label15);

_field_language_format = new JTextField("bytecode", 10);
language_c.weightx = 0.5;
language_c.gridx = 1;
language_c.gridy = 3;
language_layout.setConstraints(_field_language_format, language_c);
language_content_pane.add(_field_language_format);

JLabel label16 = new JLabel(":filter");
language_c.weightx = 0.5;
language_c.gridx = 0;
language_c.gridy = 4;
language_layout.setConstraints(label16, language_c);
language_content_pane.add(label16);
```

```
_field_language_filter = new JTextField("", 10);
language_c.weightx = 0.5;
language_c.gridx = 1;
language_c.gridy = 4;
language_layout.setConstraints(_field_language_filter, language_c);
language_content_pane.add(_field_language_filter);

JLabel label17 = new JLabel(":dependencies");
anguage_c.weightx = 0.5;
language_c.gridx = 0;
language_c.gridy = 5;
language_layout.setConstraints(label17, language_c);
language_content_pane.add(label17);

_field_language_dependencies = new JTextField("", 10);
language_c.weightx = 0.5;
language_c.gridx = 1;
language_c.gridy = 5;
language_layout.setConstraints(_field_language_dependencies, language_c);
language_content_pane.add(_field_language_dependencies);

//put it in profile pane
profile_c.weightx = 0.5;
profile_c.gridx = 0;
profile_c.gridy = 1;
profile_layout.setConstraints(language_content_pane, profile_c);
profile_content_pane.add(language_content_pane);

//os
JPanel os_content_pane = new JPanel();
GridBagLayout os_layout = new GridBagLayout();
GridBagConstraints os_c = new GridBagConstraints();

os_content_pane.setLayout(os_layout);

os_c.fill = GridBagConstraints.HORIZONTAL;

os_c.weightx = 0.5;
os_content_pane.setBorder(
       BorderFactory.createCompoundBorder(
              BorderFactory.createTitledBorder(":os"),
               BorderFactory.createEmptyBorder(5,5,5,5)));

JLabel label19 = new JLabel(":name");
os_c.weightx = 0.5;
os_c.gridx = 0;
os_c.gridy = 0;
os_layout.setConstraints(label19, os_c);
os_content_pane.add(label19);

_field_os_name = new JTextField("", 10);
os_c.weightx = 0.5;
os_c.gridx = 1;
os_c.gridy = 0;
os_layout.setConstraints(_field_os_name, os_c);
os_content_pane.add(_field_os_name);

JLabel label28 = new JLabel(":major-version");
os_c.weightx = 0.5;
os_c.gridx = 0;
os_c.gridy = 1;
os_layout.setConstraints(label28, os_c);
os_content_pane.add(label28);

_field_os_major_version = new JTextField("", 10);
os_c.weightx = 0.5;
os_c.gridx = 1;
os_c.gridy = 1;
os_layout.setConstraints(_field_os_major_version, os_c);
os_content_pane.add(_field_os_major_version);
```

```
JLabel label20 = new JLabel(":minor-version");
os_c.weightx = 0.5;
os_c.gridx = 0;
os_c.gridy = 2;
os_layout.setConstraints(label20, os_c);
os_content_pane.add(label20);

_field_os_minor_version = new JTextField("", 10);
os_c.weightx = 0.5;
os_c.gridx = 1;
os_c.gridy = 2;
os_layout.setConstraints(_field_os_minor_version, os_c);
os_content_pane.add(_field_os_minor_version);

JLabel label21 = new JLabel(":hardware");
os_c.weightx = 0.5;
os_c.gridx = 0;
os_c.gridy = 3;
os_layout.setConstraints(label21, os_c);
os_content_pane.add(label21);

_field_os_hardware = new JTextField("32 RAM", 10);
os_c.weightx = 0.5;
os_c.gridx = 1;
os_c.gridy = 3;
os_layout.setConstraints(_field_os_hardware, os_c);
os_content_pane.add(_field_os_hardware);

JLabel label22 = new JLabel(":dependencies");
os_c.weightx = 0.5;
os_c.gridx = 0;
os_c.gridy = 4;
os_layout.setConstraints(label22, os_c);
os_content_pane.add(label22);

_field_os_dependencies = new JTextField("", 10);
os_c.weightx = 0.5;
os_c.gridx = 1;
os_c.gridy = 4;
os_layout.setConstraints(_field_os_dependencies, os_c);
os_content_pane.add(_field_os_dependencies);

//put it in profile pane
profile_c.weightx = 0.5;
profile_c.gridx = 0;
profile_c.gridy = 2;
profile_layout.setConstraints(os_content_pane, profile_c);
profile_content_pane.add(os_content_pane);

//put it in content pane
c.gridwidth = 2;
c.weightx = 0.5;
c.gridx = 0;
c.gridy = 4;
layout.setConstraints(profile_content_pane, c);
content_pane.add(profile_content_pane);

JLabel label23 = new JLabel(":agent-mobility-protocol ");
c.weightx = 0.5;
c.gridx = 0;
c.gridy = 19;
layout.setConstraints(label23, c);
content_pane.add(label23);

_field_agent_mobility_protocol = new JTextField("simple-migration-protocol", 10);
c.weightx = 0.5;
c.gridx = 1;
c.gridy = 19;
layout.setConstraints(_field_agent_mobility_protocol, c);
content_pane.add(_field_agent_mobility_protocol);
```

```
JLabel label24 = new JLabel(":agent-code");
c.weightx = 0.5;
c.gridx = 0;
c.gridy = 20;
layout.setConstraints(label24, c);
content_pane.add(label24);

_field_agent_code = new JTextField("mobility.SearchAgent", 10);
c.weightx = 0.5;
c.gridx = 1;
c.gridy = 20;
layout.setConstraints(_field_agent_code, c);
content_pane.add(_field_agent_code);

JLabel label25 = new JLabel(":agent-data");
c.weightx = 0.5;
c.gridx = 0;
c.gridy = 21;
layout.setConstraints(label25, c);
content_pane.add(label25);

//try to get the local host name
try
{
    _field_agent_data = new JTextField(1 + " " + _agent_data + " " +
                        InetAddress.getLocalHost().getHostName() + " 0", 10);
}
catch ( UnknownHostException uhe )
{
    _field_agent_data = new JTextField(1 + " " + _agent_data + " ERROR 0", 10);
}
c.weightx = 0.5;
c.gridx = 1;
c.gridy = 21;
layout.setConstraints(_field_agent_data, c);
content_pane.add(_field_agent_data);

JLabel label26 = new JLabel(":agent-version");
c.weightx = 0.5;
c.gridx = 0;
c.gridy = 22;
layout.setConstraints(label26, c);
content_pane.add(label26);

_field_agent_version = new JTextField("", 10);
c.weightx = 0.5;
c.gridx = 1;
c.gridy = 22;
layout.setConstraints(_field_agent_version, c);
content_pane.add(_field_agent_version);

JLabel label27 = new JLabel(":signature");
weightx = 0.5;
c.gridx = 0;
c.gridy = 23;
layout.setConstraints(label27, c);
content_pane.add(label27);

_field_signature = new JTextField("Milla", 10);
c.weightx = 0.5;
c.gridx = 1;
c.gridy = 23;
layout.setConstraints(_field_signature, c);
content_pane.add(_field_signature);

c.fill = GridBagConstraints.VERTICAL;
JButton button1 = new JButton("OK");
button1.setActionCommand("Confirm agent details");
button1.addActionListener(this);
c.weightx = 0.0;
c.gridwidth = 4;
```

```
            c.gridx = 0;
            c.gridy = 24;
            layout.setConstraints(button1, c);
            content_pane.add(button1);

            _detail_frame.pack();
            _detail_frame.setVisible(true);

        }

    }
```

APPENDIX I: AGENT PROFILES

PLATFORM.PROFILE

```
<?xml version="1.0"?>

<!--    ************************************************************************
       Open Source Copyright Notice and License: FIPA-OS

       1. The programs and other works made available to you in these files
          ("the Programs") are Copyright (c) 1999 - 2000 Nortel Networks
          Corporation, 8200 Dixie Road, Suite 100, Brampton, Ontario, Canada
          L6R 5P6.
          All rights reserved.

       2. Your rights to copy, distribute and modify the Programs are as set
          out in the Nortel Networks FIPA-OS Public License, a copy of which
          can be found in file "FIPA_OS_Public_Licence.txt" and the latest
          version can also be found at http://www.nortelnetworks.com/fipa-os.
          By downloading the files containing the Programs you accept the
          terms and conditions of the Public License. You do not have to
          accept these terms and conditions, but unless you do so you have no
          rights to use the Programs.

   ************************************************************************ -->

<rdf:RDF  xml:lang="en"
       xmlns="http://www.nortelnetworks.com/fipa-os/schemas/profile#"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:ap="http://www.nortelnetworks.com/fipa-os/schemas/Objects#">

<ap:Profile rdf:about="default_platform_profile">

   <!-- Transport mechanism -->
 <ap:default_transport_mechanism>RMI</ap:default_transport_mechanism>

   <!-- Transport protocol -->
 <ap:default_transport_protocol>iiop</ap:default_transport_protocol>

   <!-- GUID of the AMS running on this platform DO NOT USE 127.0.0.1 in place
          of localhost when using Voyager transport-->
 <ap:ams>ams@iiop://vaio:9000/acc</ap:ams>

   <!-- where agent profiles on this platform should be saved. -->
 <ap:profile_directory>c:\MILLA\profiles</ap:profile_directory>

   <!-- port for the Naming Service FOR SUNIDL ENSURE THAT THE NAME SERVER IS
          RUNNING ON THE SAME PORT-->
   <!-- The SUNIDL Name Server Port is set in StartNameServer.bat -->
 <ap:naming_service_port>1098</ap:naming_service_port>

   <!--   default ports
      voyager3  1098
      sunidl   1198
      rmi   1100 -->


</ap:Profile>

</rdf:RDF>
```

MMS.PROFILE

```
<?xml version="1.0"?>

<!--    ************************************************************************
       Open Source Copyright Notice and License: FIPA-OS

       1. The programs and other works made available to you in these files
          ("the Programs") are Copyright (c) 1999 - 2000 Nortel Networks
          Corporation, 8200 Dixie Road, Suite 100, Brampton, Ontario, Canada
          L6R 5P6.
          All rights reserved.

       2. Your rights to copy, distribute and modify the Programs are as set
          out in the Nortel Networks FIPA-OS Public License, a copy of which
          can be found in file "FIPA_OS_Public_Licence.txt" and the latest
          version can also be found at http://www.nortelnetworks.com/fipa-os.
          By downloading the files containing the Programs you accept the
          terms and conditions of the Public License. You do not have to
          accept these terms and conditions, but unless you do so you have no
          rights to use the Programs.

   ************************************************************************ -->

<rdf:RDF  xml:lang="en"
       xmlns="http://www.nortelnetworks.com/fipa-os/schemas/profile#"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:ap="http://www.nortelnetworks.com/fipa-os/schemas/Objects#"
       xmlns:mad="MobileAgentDescription#">

<!-- Transport Bag object -->
<ap:TransportBag rdf:about="transports">
   <ap:transport_li rdf:resource="transport1"/>
</ap:TransportBag>

<!-- Transport object -->
<ap:Transport rdf:about="transport1">
   <ap:mechanism></ap:mechanism>
   <ap:protocol></ap:protocol>
   <ap:port_number>9025</ap:port_number>
</ap:Transport>

<!-- Database Bag object -->
<ap:DatabaseBag rdf:about="databases">
   <ap:database_li rdf:resource="database1"/>
</ap:DatabaseBag>

<!-- Database object -->
<ap:Database rdf:about="database1">
   <ap:database_type></ap:database_type>

</ap:Database>

<mad:MobileAgentDescription rdf:about="description">
   <mad:agent-mobility-protocol>simple-migration-protocol</mad:agent-mobility-
protocol>

   <mad:system-name>FIPA-OS</mad:system-name>
```

```
    <mad:system-major-version>1.03</mad:system-major-version>
    <mad:system-minor-version>1.03</mad:system-minor-version>
    <mad:system-dependencies>MobilityManagementSystem</mad:system-dependencies>

    <mad:language-name>Java</mad:language-name>
    <mad:language-major-version>1.2.2</mad:language-major-version>
    <mad:language-minor-version>1.2</mad:language-minor-version>
    <mad:language-format>bytecode</mad:language-format>

    <mad:os-hardware>Java</mad:os-hardware>
</mad:MobileAgentDescription>

</rdf:RDF>
```

SEARCHER.PROFILE

```
<?xml version="1.0"?>

<!--    ************************************************************************
       Open Source Copyright Notice and License: FIPA-OS

       1. The programs and other works made available to you in these files
          ("the Programs") are Copyright (c) 1999 - 2000 Nortel Networks
          Corporation, 8200 Dixie Road, Suite 100, Brampton, Ontario, Canada
          L6R 5P6.
          All rights reserved.

       2. Your rights to copy, distribute and modify the Programs are as set
          out in the Nortel Networks FIPA-OS Public License, a copy of which
          can be found in file "FIPA_OS_Public_Licence.txt" and the latest
          version can also be found at http://www.nortelnetworks.com/fipa-os.
          By downloading the files containing the Programs you accept the
          terms and conditions of the Public License. You do not have to
          accept these terms and conditions, but unless you do so you have no
          rights to use the Programs.

   ************************************************************************ -->

<rdf:RDF  xml:lang="en"
       xmlns="http://www.nortelnetworks.com/fipa-os/schemas/profile#"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:ap="http://www.nortelnetworks.com/fipa-os/schemas/Objects#"
       xmlns:sa="SearchAgent#">

<!-- Transport Bag object -->
<ap:TransportBag rdf:about="transports">
   <ap:transport_li rdf:resource="transport1"/>
</ap:TransportBag>

<!-- Transport object -->
<ap:Transport rdf:about="transport1">
   <ap:mechanism></ap:mechanism>
   <ap:protocol></ap:protocol>
   <ap:port_number>9000</ap:port_number>
</ap:Transport>

<!-- Database Bag object -->
<ap:DatabaseBag rdf:about="databases">
   <ap:database_li rdf:resource="database1"/>
</ap:DatabaseBag>

<!-- Database object -->
<ap:Database rdf:about="database1">
   <ap:database_type></ap:database_type>

</ap:Database>

<!-- MMS1 -->
<sa:MMS rdf:about="mms1">
   <sa:name>mms1</sa:name>
</sa:MMS>

<!-- MMS2 -->
<sa:MMS rdf:about="mms2">
   <sa:name>mms2</sa:name>
</sa:MMS>
```

```
</rdf:RDF>
```

APPENDIX J: TEST MESSAGES

1.0

(request: sender iotestagent@iiop://pc99:9000/acc :receiver mms1@iiop://pc99:9000/acc :content (testing) :protocol fipa-request :ontology fipa-mobile-agent-management :conversation-id test-123)

2.0

(request :sender searcher@iiop://pc99:9000/acc :receiver mms2@iiop://pc99:9000/acc :content (<?xml version="1.0"?> <!DOCTYPE action SYSTEM "description.dtd"> <action to="mms"> <move-state> <fipa-mobile-agent-description> <agent-name>searcher</agent-name> <address>iiop://pc99:9000/acc</address> <destination>pc119</destination> <agent-profile> <system> <name>Voyager</name> </system> <language> <name>Java</name> <major-version>1.2.2</major-version> <format>bytecode</format> </language> <os> <hardware>32 RAM</hardware> </os> </agent-profile> <agent-mobility-protocol>simple-migration-protocol</agent-mobility-protocol> <agent-code>mobility.SearchAgent</agent-code> <agent-data>1 c:\ txt pc99 0</agent-data> <signature>Milla</signature> </fipa-mobile-agent-description> </move-state> </action>) :language XML :ontology fipa-mobile-agent-management :protocol fipa-request :conversation-id searcher@iiop://pc99:9000/acc9547708051761)

2.1

(request :sender searcher@iiop://pc99:9000/acc :receiver mms2@iiop://pc99:9000/acc

:content (<?xml version="1.0"?> <!DOCTYPE action SYSTEM "description.dtd">
<action to="mms"> <move-state> <fipa-mobile-agent-description> <agent-
name>searcher</agent-name> <address>iiop://localhost:9000/acc</address>
<destination>vaio</destination> <agent-profile> <system> <name>FIPA-OS</name>
<major-version>1.01</major-version> <minor-version>1.01</minor-version>
<dependencies>MobilityManagementSystem</dependencies> </system> <language>
<name>Java</name> <major-version>1.2.2</major-version>
<format>bytecode</format> </language> <os> <hardware>32 RAM</hardware>
</os> </agent-profile> <agent-mobility-protocol>simple-migration-protocol</agent-
mobility-protocol> <agent-code>mobility.SearchAgent</agent-code> <agent-data>1 c:\
txt pc99 0</agent-data> <signature>Milla</signature> </fipa-mobile-agent-
description> </move-state> </action>) :language XML :ontology fipa-mobile-agent-
management :protocol fipa-request :conversation-id
searcher@iiop://pc99:9000/acc9547711656441)

2.2

(request :sender searcher@iiop://pc99:9000/acc :receiver mms2@iiop://pc99:9000/acc
:content (<?xml version="1.0"?> <!DOCTYPE action SYSTEM "description.dtd">
<action to="mms"> <move-state> <fipa-mobile-agent-description> <agent-
name>searcher</agent-name> <address>iiop://localhost:9000/acc</address>
<destination>pc119</destination> <agent-profile> <system> <name>FIPA-
OS</name> <major-version>1.03</major-version> <minor-version>1.03</minor-
version> <dependencies>Voyager</dependencies> </system> <language>
<name>Java</name> <major-version>1.2.2</major-version>

<format>bytecode</format> </language> <os> <hardware>32 RAM</hardware> </os> </agent-profile> <agent-mobility-protocol>simple-migration-protocol</agent-mobility-protocol> <agent-code>mobility.SearchAgent</agent-code> <agent-data>1 c:\ txt pc99 0</agent-data> <signature>Milla</signature> </fipa-mobile-agent-description> </move-state> </action>) :language XML :ontology fipa-mobile-agent-management :protocol fipa-request :conversation-id searcher@iiop://pc99:9000/acc9547713235111)

2.3

(request :sender searcher@iiop://pc99:9000/acc :receiver mms2@iiop://pc99:9000/acc :content (<?xml version="1.0"?> <!DOCTYPE action SYSTEM "description.dtd"> <action to="mms"> <move-state> <fipa-mobile-agent-description> <agent-name>searcher</agent-name> <address>iiop://localhost:9000/acc</address> <destination>pc119</destination> <agent-profile> <system> <name>FIPA-OS</name> <major-version>1.03</major-version> <minor-version>1.03</minor-version> <dependencies>MobilityManagementSystem</dependencies> </system> <language> <name>C++</name> <major-version></major-version> <minor-version></minor-version> <format></format> </language> <os> <hardware>32 RAM</hardware> </os> </agent-profile> <agent-mobility-protocol>simple-migration-protocol</agent-mobility-protocol> <agent-code>mobility.SearchAgent</agent-code> <agent-data>1 c:\ txt pc99 0</agent-data> <signature>Milla</signature> </fipa-mobile-agent-description> </move-state> </action>) :language XML :ontology fipa-mobile-agent-management :protocol fipa-request :conversation-id searcher@iiop://pc99:9000/acc9547714700721)

2.4

(request :sender searcher@iiop://pc99:9000/acc :receiver mms2@iiop://pc99:9000/acc :content (<?xml version="1.0"?> <!DOCTYPE action SYSTEM "description.dtd"> <action to="mms"> <move-state> <fipa-mobile-agent-description> <agent-name>searcher</agent-name> <address>iiop://pc99:9000/acc</address> <destination>pc28</destination> <agent-profile> <system> <name>FIPA-OS</name> <major-version>1.03</major-version> <minor-version>1.03</minor-version> <dependencies>MobilityManagementSystem</dependencies> </system> <language> <name>Java</name> <major-version>1.1.8</major-version> <format>bytecode</format> </language> <os> <hardware>32 RAM</hardware> </os> </agent-profile> <agent-mobility-protocol>simple-migration-protocol</agent-mobility-protocol> <agent-code>mobility.SearchAgent</agent-code> <agent-data>1 c:\txt pc99 0</agent-data> <signature>Milla</signature> </fipa-mobile-agent-description> </move-state> </action>) :language XML :ontology fipa-mobile-agent-management :protocol fipa-request :conversation-id searcher@iiop://pc99:9000/acc9548405360141)

2.5

(request :sender searcher@iiop://pc99:9000/acc :receiver mms2@iiop://pc99:9000/acc :content (<?xml version="1.0"?> <!DOCTYPE action SYSTEM "description.dtd"> <action to="mms"> <move-state> <fipa-mobile-agent-description> <agent-name>searcher</agent-name> <address>iiop://pc99:9000/acc</address> <destination>pc28</destination> <agent-profile> <system> <name>FIPA-OS</name> <major-version>1.03</major-version> <minor-version>1.03</minor-

version> <dependencies>MobilityManagementSystem</dependencies> </system> <language> <name>Java</name> <major-version>1.2.2</major-version> <format>source</format> </language> <os> <hardware>32 RAM</hardware> </os></agent-profile> <agent-mobility-protocol>simple-migration-protocol</agent-mobility-protocol> <agent-code>mobility.SearchAgent</agent-code> <agent-data>1 c:\ txt pc99 0</agent-data> <signature>Milla</signature> </fipa-mobile-agent-description> </move-state> </action>) :language XML :ontology fipa-mobile-agent-management :protocol fipa-request :conversation-id searcher@iiop://pc99:9000/acc9548408755121)

2.6

(request :sender searcher@iiop://pc99:9000/acc :receiver mms2@iiop://pc99:9000/acc :content (<?xml version="1.0"?> <!DOCTYPE action SYSTEM "description.dtd"> <action to="mms"> <move-state> <fipa-mobile-agent-description> <agent-name>searcher</agent-name> <address>iiop://pc99:9000/acc</address> <destination>pc28</destination> <agent-profile> <system> <name>FIPA-OS</name> <major-version>1.03</major-version> <minor-version>1.03</minor-version> <dependencies>MobilityManagementSystem</dependencies> </system> <language> <name>Java</name> <major-version>1.2.2</major-version> <format>bytecode</format> </language> <os> <hardware>32 RAM</hardware> </os> </agent-profile> <agent-mobility-protocol>full-migration-protocol</agent-mobility-protocol> <agent-code>mobility.SearchAgent</agent-code> <agent-data>1 c:\ txt pc99 0</agent-data> <signature>Milla</signature> </fipa-mobile-agent-description> </move-state> </action>) :language XML :ontology fipa-mobile-agent-

management :protocol fipa-request :conversation-id
searcher@iiop://pc99:9000/acc9548412831491)

2.7

(request :sender searcher@iiop://pc99:9000/acc :receiver mms2@iiop://pc99:9000/acc
:content (<?xml version="1.0"?> <!DOCTYPE action SYSTEM "description.dtd">
<action to="mms"> <move-state> <fipa-mobile-agent-description> <agent-
name>searcher</agent-name> <address>iiop://pc99:9000/acc</address>
<destination>pc28</destination> <agent-profile> <system> <name>FIPA-
OS</name> <major-version>1.03</major-version> <minor-version>1.03</minor-
version> <dependencies>MobilityManagementSystem</dependencies> </system>
<language> <name>Java</name> <major-version>1.2.2</major-version>
<format>bytecode</format> </language> <os> <hardware>32 RAM</hardware>
</os> </agent-profile> <agent-mobility-protocol>simple-migration-protocol</agent-
mobility-protocol> <agent-code>aw.platform.DirectoryFacilitator</agent-code> <agent-
data>1 c:\ txt pc99 0</agent-data> <signature>Milla</signature> </fipa-mobile-agent-
description> </move-state> </action>) :language XML :ontology fipa-mobile-agent-
management :protocol fipa-request :conversation-id
searcher@iiop://pc99:9000/acc9548414495581)

3.0

(request :sender searcher@iiop://pc99:9000/acc :receiver mms2@iiop://pc99:9000/acc
:content (<?xml version="1.0"?> <!DOCTYPE action SYSTEM "description.dtd">
<action to="mms"> <move-state> <fipa-mobile-agent-description> <agent-

name>searcher</agent-name> <address>iiop://pc99:9000/acc</address>
<destination>pc28</destination> <agent-profile> <system> <name>FIPA-
OS</name> <major-version>1.03</major-version> <minor-version>1.03</minor-
version> <dependencies>MobilityManagementSystem</dependencies> </system>
<language> <name>Java</name> <major-version>1.2.2</major-version>
<format>bytecode</format> </language> </agent-profile> <agent-mobility-
protocol>simple-migration-protocol</agent-mobility-protocol> <agent-
code>mobility.SearchAgent</agent-code> <agent-data>1 c:\ txt pc99 0</agent-data>
<signature>Milla</signature> </fipa-mobile-agent-description> </move-state>
</action>) :language XML :ontology fipa-mobile-agent-management :protocol fipa-
request :conversation-id searcher@iiop://pc99:9000/acc9548417012301)

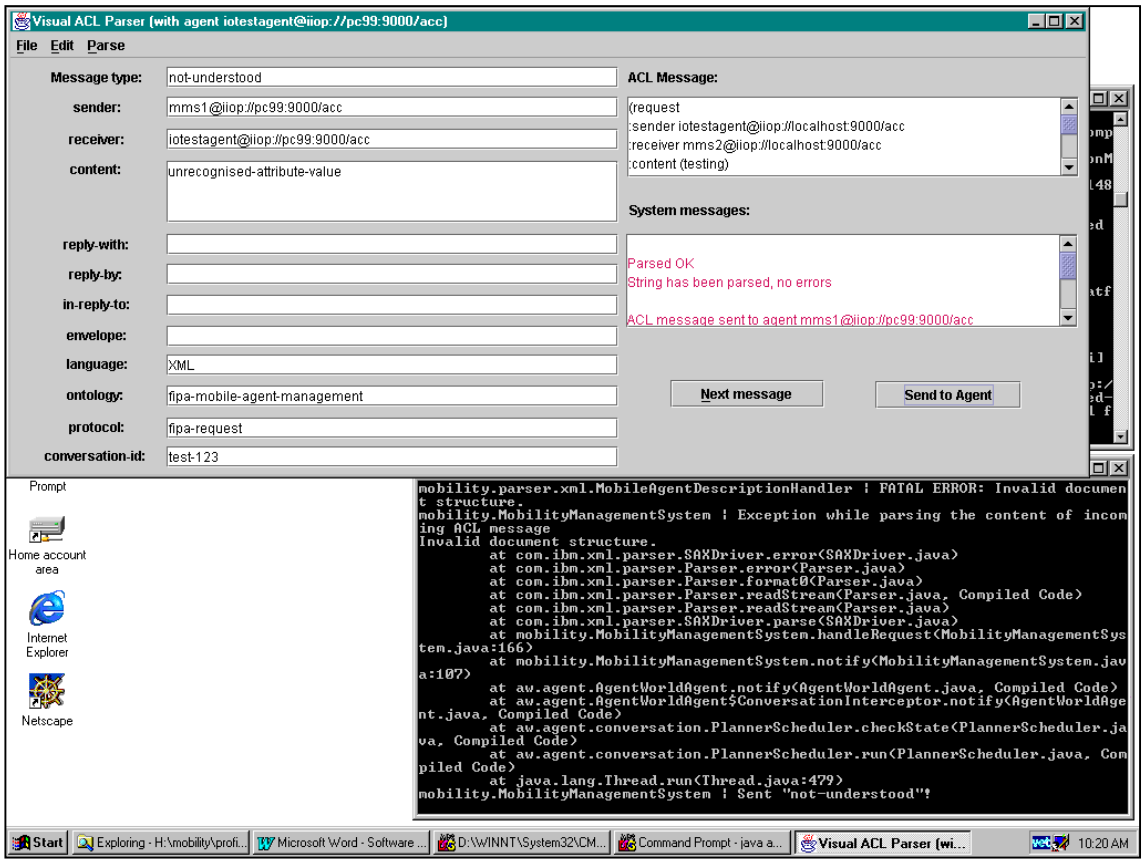
3.1

(request :sender searcher@iiop://pc99:9000/acc :receiver mms2@iiop://pc99:9000/acc
:content (<?xml version="1.0"?> <!DOCTYPE action SYSTEM "description.dtd">
<action to="mms"> <move-state> <fipa-mobile-agent-description> <agent-
name>searcher</agent-name> <address>iiop://localhost:9000/acc</address>
<destination>pc119</destination> <agent-profile> <system> <name>FIPA-
OS</name> <major-version>1.03</major-version> <minor-version>1.03</minor-
version> <dependencies>MobilityManagementSystem</dependencies> </system>
<language> <name>Java</name> <major-version>1.2.2</major-version>
<format>bytecode</format> </language> <os> <hardware>32 RAM</hardware>
</os> </agent-profile> <agent-mobility-protocol>simple-migration-protocol</agent-
mobility-protocol> <agent-code>mobility.SearchAgent</agent-code> <agent-data>1

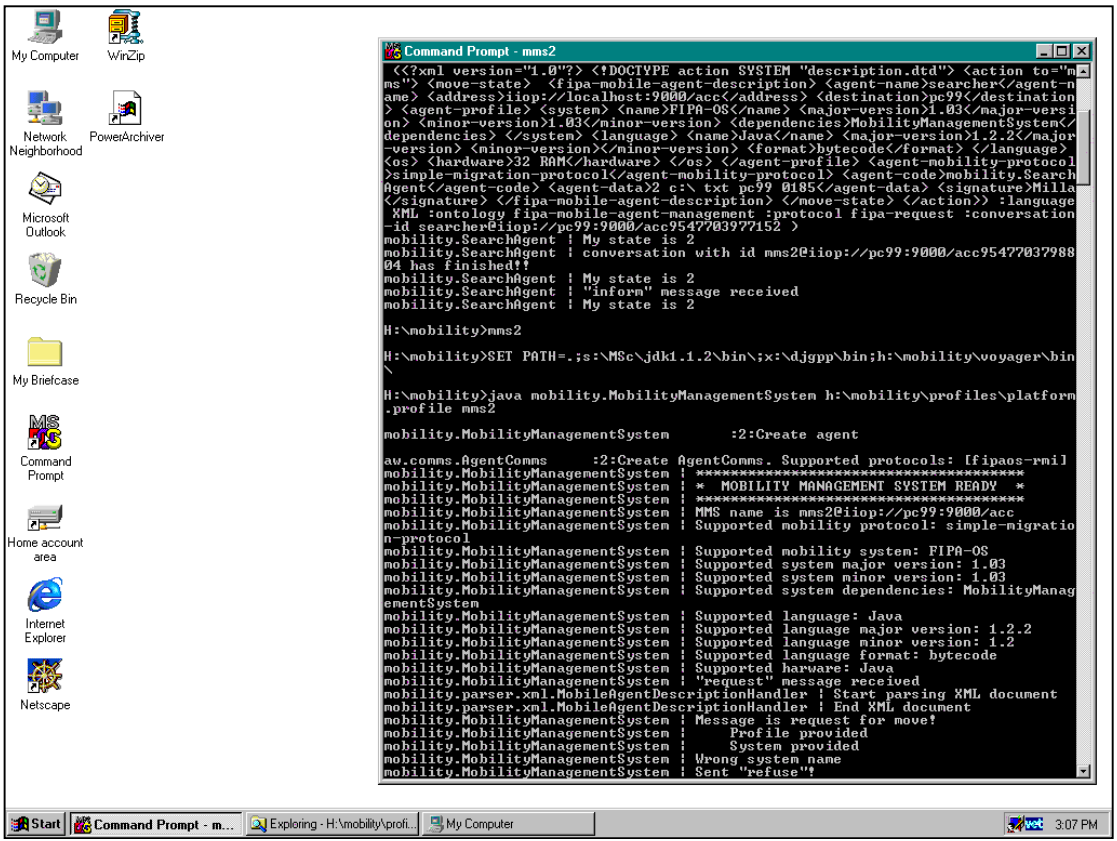
c:\ txt pc99 0</agent-data> <signature>Milla</signature> </fipa-mobile-agent-description> </move-state> </action>) :language XML :ontology fipa-mobile-agent-management :protocol fipa-request :conversation-id searcher@iiop://pc99:9000/acc9547715978351)

APPENDIX K: TEST SCREENSHOTS

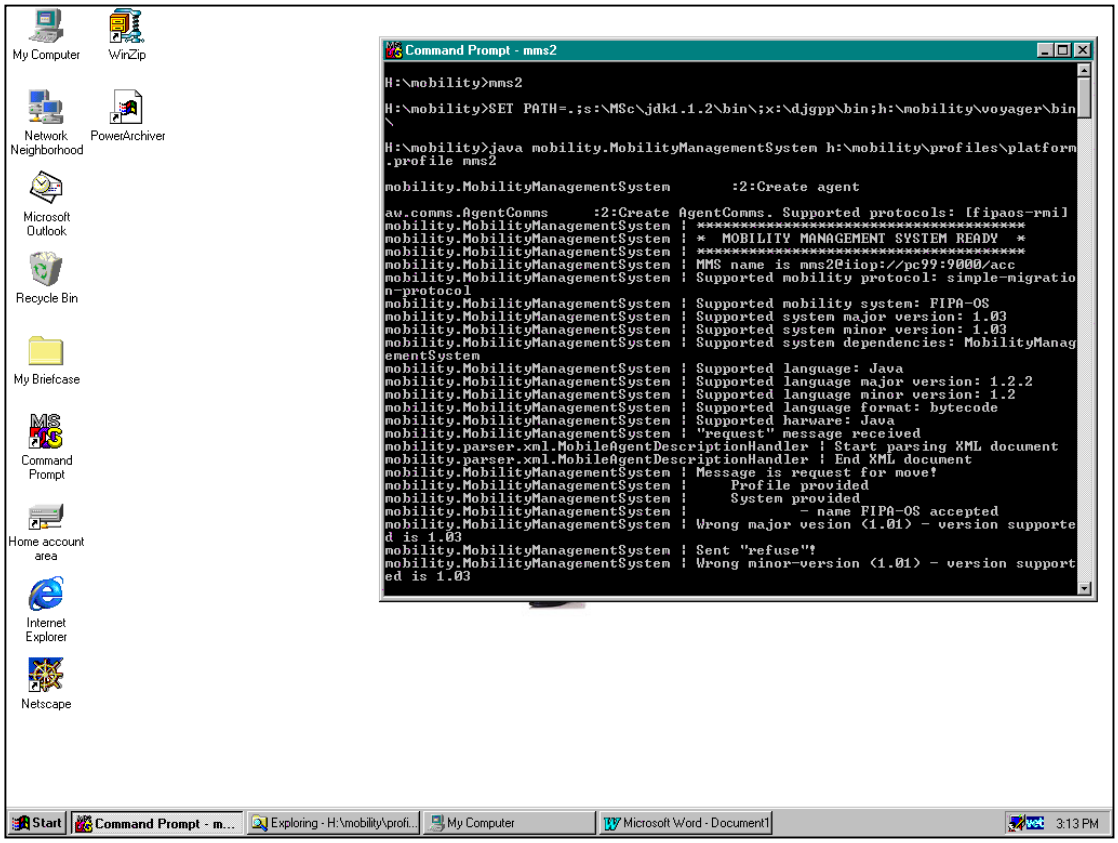
1.0



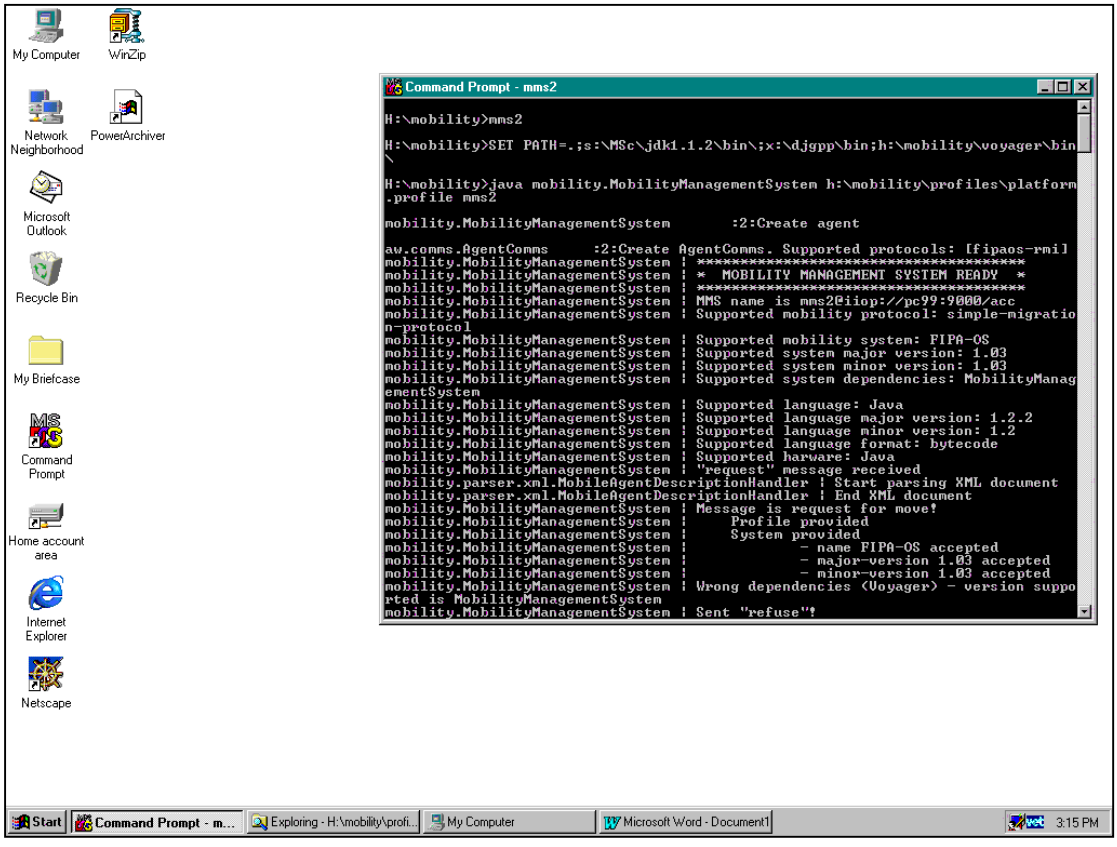
2.0



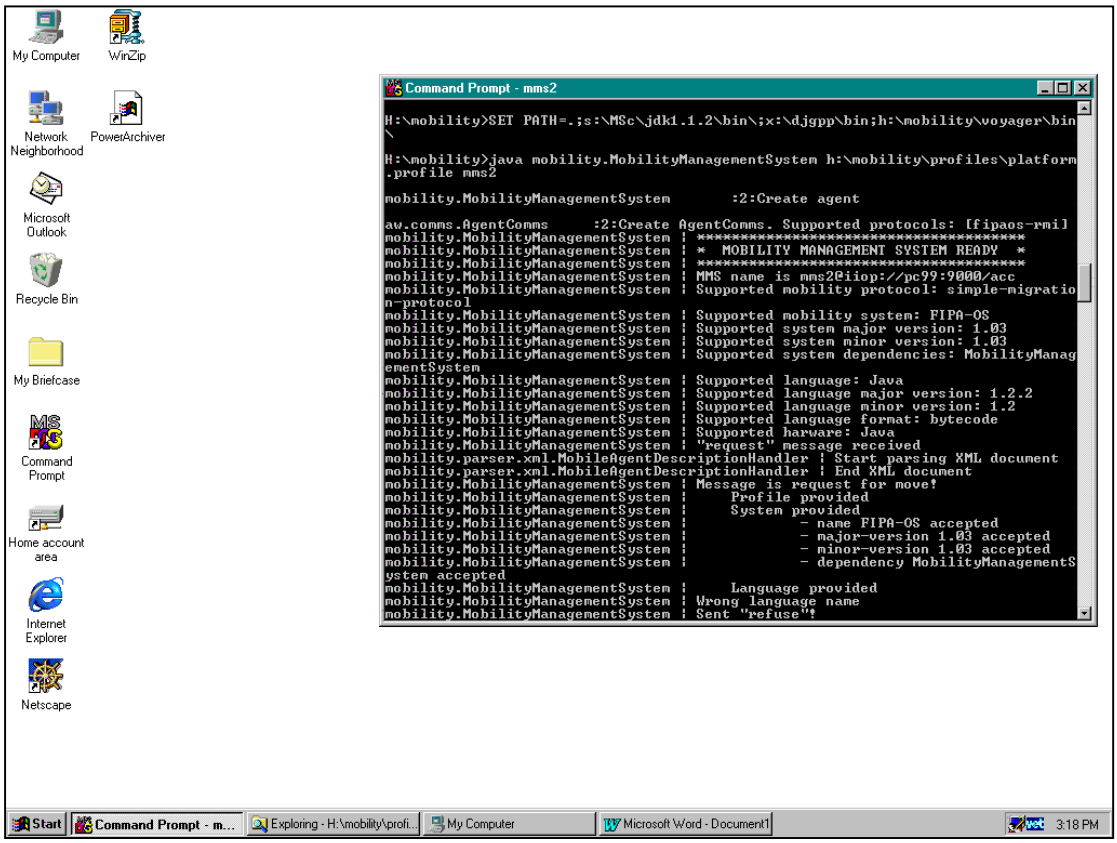
2.1


My Computer
WinZip
Network Neighborhood
PowerArchiver
Microsoft Outlook
Recycle Bin
My Briefcase
Command Prompt
Home account area
Internet Explorer
Netscape
Command Prompt - mms2
H:\mobility>mms2
H:\mobility>SET PATH=.;s:\MSc\jdk1.2\bin\;x:\djgpp\bin;h:\mobility\voyager\bin
\
H:\mobility>java mobility.MobilityManagementSystem h:\mobility\profiles\platform
.profile mms2
mobility.MobilityManagementSystem :2:Create agent
aw.comms.AgentComms :2:Create AgentComms. Supported protocols: [fipaos-rmi]
mobility.MobilityManagementSystem ! **********
mobility.MobilityManagementSystem ! * MOBILITY MANAGEMENT SYSTEM READY *
mobility.MobilityManagementSystem ! **********
mobility.MobilityManagementSystem ! MMS name is mms2@iiop://pc99:9000/acc
mobility.MobilityManagementSystem ! Supported mobility protocol: simple-migratio
n-protocol
mobility.MobilityManagementSystem ! Supported mobility system: FIPA-OS
mobility.MobilityManagementSystem ! Supported system major version: 1.03
mobility.MobilityManagementSystem ! Supported system minor version: 1.03
mobility.MobilityManagementSystem ! Supported system dependencies: MobilityManag
ementSystem
mobility.MobilityManagementSystem ! Supported language: Java
mobility.MobilityManagementSystem ! Supported language major version: 1.2.2
mobility.MobilityManagementSystem ! Supported language minor version: 1.2
mobility.MobilityManagementSystem ! Supported language format: bytecode
mobility.MobilityManagementSystem ! Supported harware: Java
mobility.MobilityManagementSystem ! "request" message received
mobility.parser.xml.MobileAgentDescriptionHandler ! Start parsing XML document
mobility.parser.xml.MobileAgentDescriptionHandler ! End XML document
mobility.MobilityManagementSystem ! Message is request for move!
mobility.MobilityManagementSystem ! Profile provided
mobility.MobilityManagementSystem ! System provided
mobility.MobilityManagementSystem ! - name FIPA-OS accepted
mobility.MobilityManagementSystem ! Wrong major vesion (1.01) - version supporte
d is 1.03
mobility.MobilityManagementSystem ! Sent "refuse"!
mobility.MobilityManagementSystem ! Wrong minor-version (1.01) - version support
ed is 1.03
Start
Command Prompt - m...
Exploring - H:\mobility\profi...
My Computer
Microsoft Word - Document1
3:13 PM

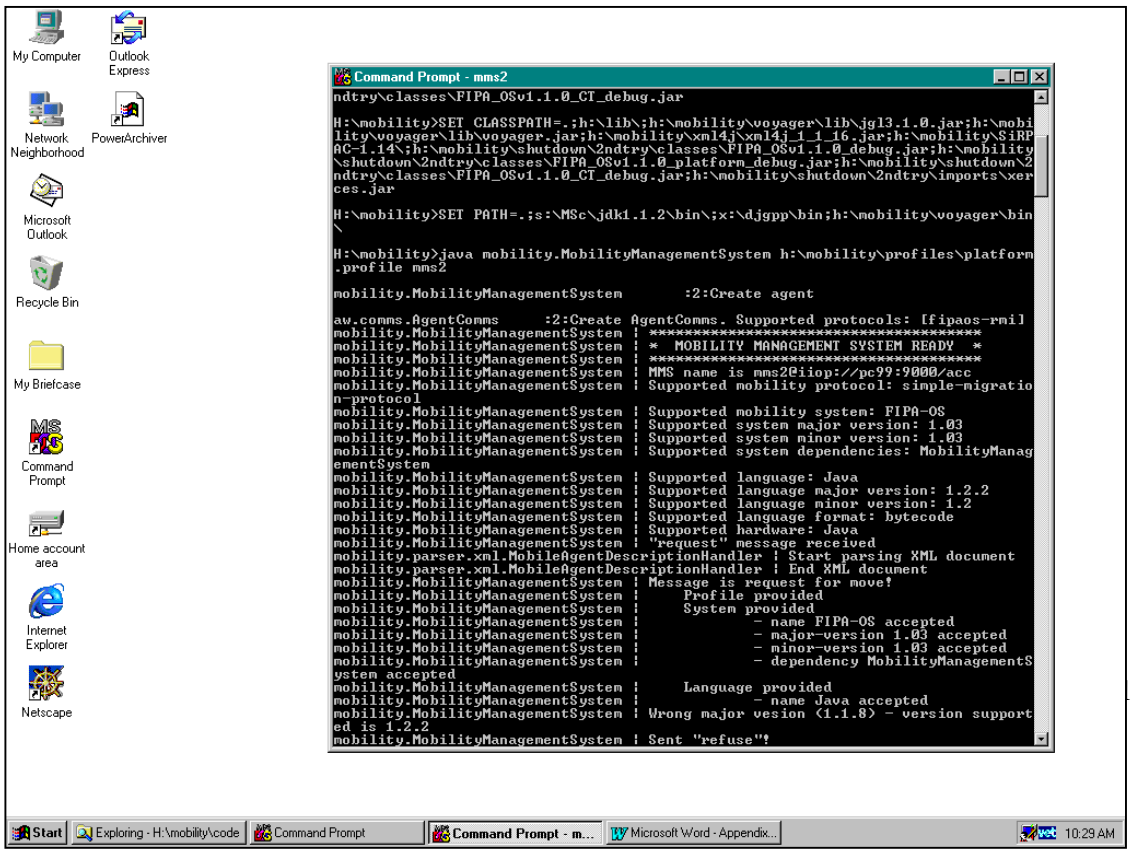
2.2


My Computer
WinZip
Network Neighborhood
PowerArchiver
Microsoft Outlook
Recycle Bin
My Briefcase
Command Prompt
Home account area
Internet Explorer
Netscape
Command Prompt - mms2
H:\mobility>mms2
H:\mobility>SET PATH=.;s:\MSc\jdk1.2\bin\;x:\djgpp\bin;h:\mobility\voyager\bin
\
H:\mobility>java mobility.MobilityManagementSystem h:\mobility\profiles\platform
.profile mms2
mobility.MobilityManagementSystem :2:Create agent
aw.comms.AgentComms :2:Create AgentComms. Supported protocols: [fipaos-rmi]
mobility.MobilityManagementSystem ! **********
mobility.MobilityManagementSystem ! * MOBILITY MANAGEMENT SYSTEM READY *
mobility.MobilityManagementSystem ! **********
mobility.MobilityManagementSystem ! MMS name is mms2@iiop://pc99:9000/acc
mobility.MobilityManagementSystem ! Supported mobility protocol: simple-migratio
n-protocol
mobility.MobilityManagementSystem ! Supported mobility system: FIPA-OS
mobility.MobilityManagementSystem ! Supported system major version: 1.03
mobility.MobilityManagementSystem ! Supported system minor version: 1.03
mobility.MobilityManagementSystem ! Supported system dependencies: MobilityManag
ementSystem
mobility.MobilityManagementSystem ! Supported language: Java
mobility.MobilityManagementSystem ! Supported language major version: 1.2.2
mobility.MobilityManagementSystem ! Supported language minor version: 1.2
mobility.MobilityManagementSystem ! Supported language format: bytecode
mobility.MobilityManagementSystem ! Supported harware: Java
mobility.MobilityManagementSystem ! "request" message received
mobility.parser.xml.MobileAgentDescriptionHandler ! Start parsing XML document
mobility.parser.xml.MobileAgentDescriptionHandler ! End XML document
mobility.MobilityManagementSystem ! Message is request for move!
mobility.MobilityManagementSystem ! Profile provided
mobility.MobilityManagementSystem ! System provided
mobility.MobilityManagementSystem ! - name FIPA-OS accepted
mobility.MobilityManagementSystem ! - major-version 1.03 accepted
mobility.MobilityManagementSystem ! - minor-version 1.03 accepted
mobility.MobilityManagementSystem ! Wrong dependencies (Voyager) - version suppo
rted is MobilityManagementSystem
mobility.MobilityManagementSystem ! Sent "refuse"!
Start
Command Prompt - m...
Exploring - H:\mobility\profi...
My Computer
Microsoft Word - Document1
3:15 PM

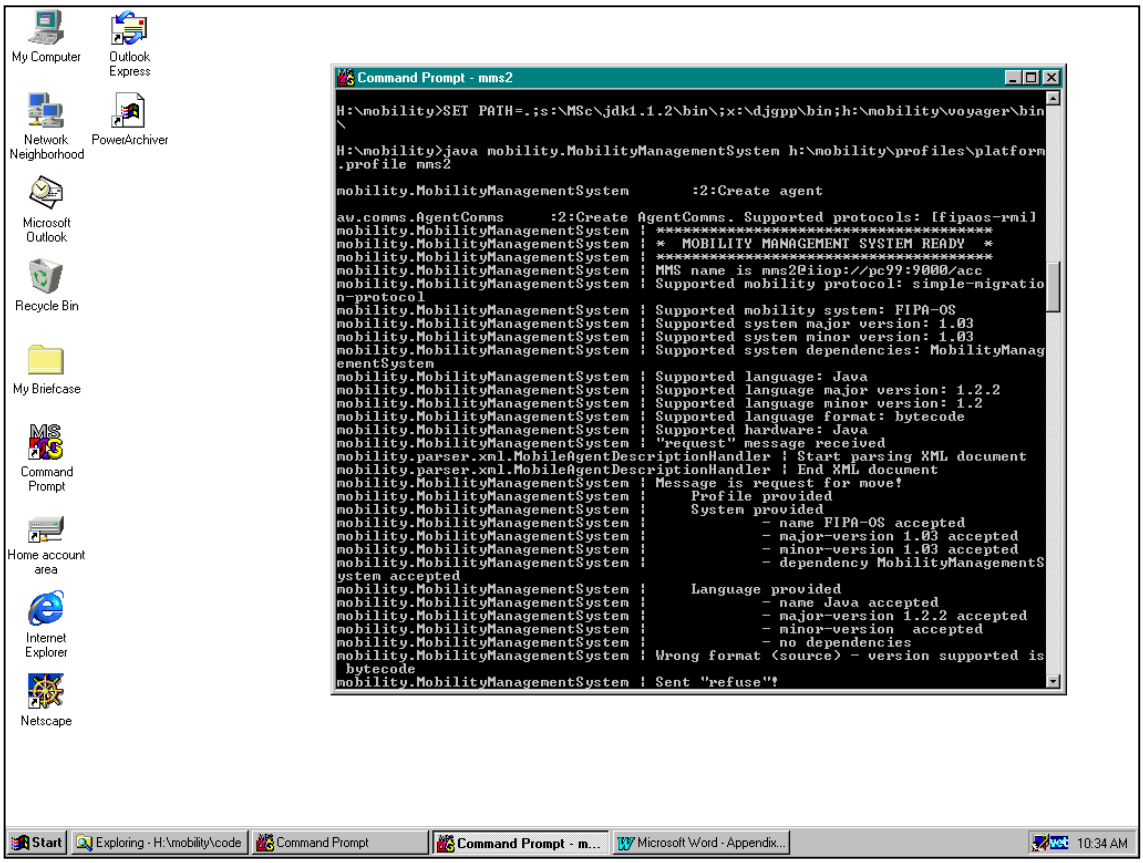
2.3


My Computer
WinZip
Network Neighborhood
PowerArchiver
Microsoft Outlook
Recycle Bin
My Briefcase
Command Prompt
Home account area
Internet Explorer
Netscape
Command Prompt - mms2
H:\mobility>SET PATH=.;s:\MSc\jdk1.2\bin\;x:\djgpp\bin;h:\mobility\voyager\bin
\
H:\mobility>java mobility.MobilityManagementSystem h:\mobility\profiles\platform
.profile mms2
mobility.MobilityManagementSystem :2:Create agent
aw.comms.AgentComms :2:Create AgentComms. Supported protocols: [fipaos-rmi]
mobility.MobilityManagementSystem | **********
mobility.MobilityManagementSystem | * MOBILITY MANAGEMENT SYSTEM READY *
mobility.MobilityManagementSystem | **********
mobility.MobilityManagementSystem | MMS name is mms2@iiop://pc99:9000/acc
mobility.MobilityManagementSystem | Supported mobility protocol: simple-migratio
n-protocol
mobility.MobilityManagementSystem | Supported mobility system: FIPA-OS
mobility.MobilityManagementSystem | Supported system major version: 1.03
mobility.MobilityManagementSystem | Supported system minor version: 1.03
mobility.MobilityManagementSystem | Supported system dependencies: MobilityManag
ementSystem
mobility.MobilityManagementSystem | Supported language: Java
mobility.MobilityManagementSystem | Supported language major version: 1.2.2
mobility.MobilityManagementSystem | Supported language minor version: 1.2
mobility.MobilityManagementSystem | Supported language format: bytecode
mobility.MobilityManagementSystem | Supported harware: Java
mobility.MobilityManagementSystem | "request" message received
mobility.parser.xml.MobileAgentDescriptionHandler | Start parsing XML document
mobility.parser.xml.MobileAgentDescriptionHandler | End XML document
mobility.MobilityManagementSystem | Message is request for move!
mobility.MobilityManagementSystem | Profile provided
mobility.MobilityManagementSystem | System provided
mobility.MobilityManagementSystem | - name FIPA-OS accepted
mobility.MobilityManagementSystem | - major-version 1.03 accepted
mobility.MobilityManagementSystem | - minor-version 1.03 accepted
mobility.MobilityManagementSystem | - dependency MobilityManagementS
ystem accepted
mobility.MobilityManagementSystem | Language provided
mobility.MobilityManagementSystem | Wrong language name
mobility.MobilityManagementSystem | Sent "refuse"!
Start
Command Prompt - m...
Exploring - H:\mobility\profi...
My Computer
Microsoft Word - Document1
3:18 PM

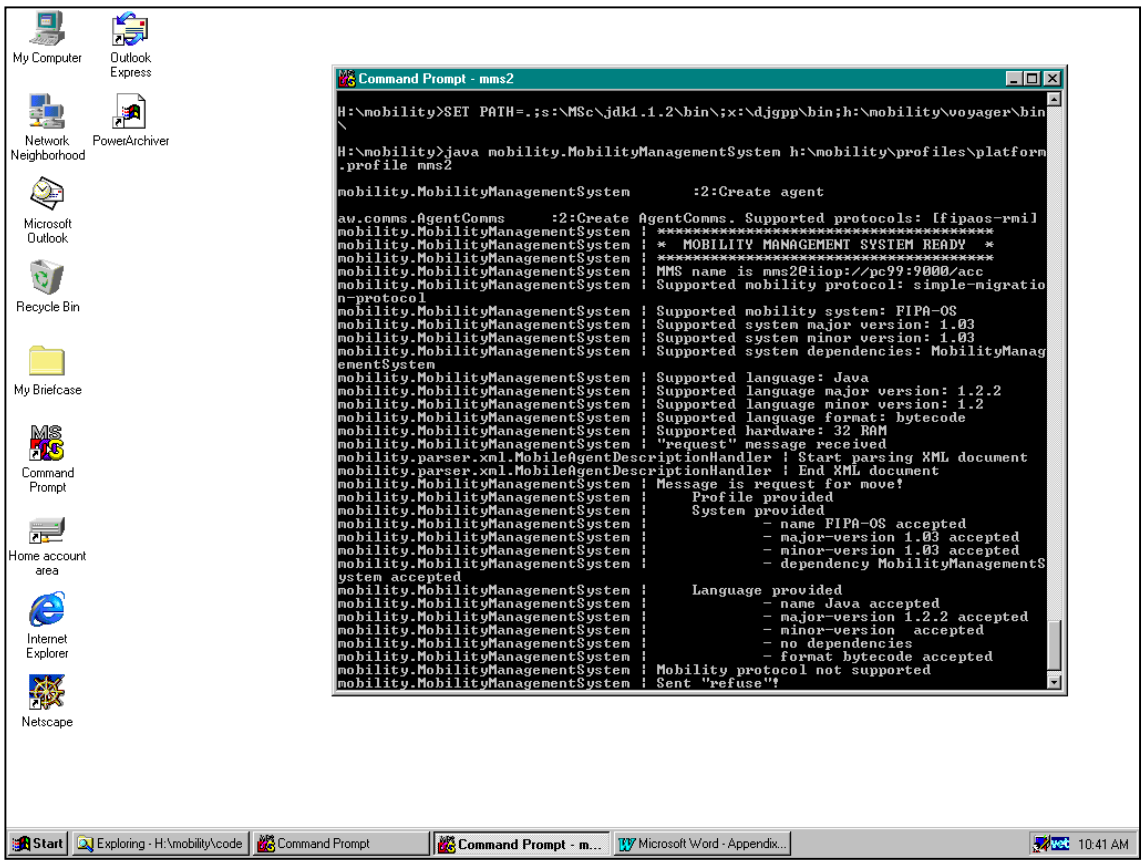
2.4


My Computer
Outlook Express
Network Neighborhood
PowerArchiver
Microsoft Outlook
Recycle Bin
My Briefcase
Command Prompt
Home account area
Internet Explorer
Netscape
Command Prompt - mms2
ndtry\classes\FIPA_OSv1.1.0_CT_debug.jar
H:\mobility>SET CLASSPATH=.;h:\lib\;h:\mobility\voyager\lib\jgl3.1.0.jar;h:\mobi
lity\voyager\lib\voyager.jar;h:\mobility\xml4j\xml4j_1_1_16.jar;h:\mobility\SiRP
AC-1.14\;h:\mobility\shutdown\2ndtry\classes\FIPA_OSv1.1.0_debug.jar;h:\mobility
\shutdown\2ndtry\classes\FIPA_OSv1.1.0_platform_debug.jar;h:\mobility\shutdown\2
ndtry\classes\FIPA_OSv1.1.0_CT_debug.jar;h:\mobility\shutdown\2ndtry\imports\xer
ces.jar
H:\mobility>SET PATH=.;s:\MSc\jdk1.2\bin\;x:\djgpp\bin;h:\mobility\voyager\bin
\
H:\mobility>java mobility.MobilityManagementSystem h:\mobility\profiles\platform
.profile mms2
mobility.MobilityManagementSystem :2:Create agent
aw.comms.AgentComms :2:Create AgentComms. Supported protocols: [fipaos-rmi]
mobility.MobilityManagementSystem | **********
mobility.MobilityManagementSystem | * MOBILITY MANAGEMENT SYSTEM READY *
mobility.MobilityManagementSystem | **********
mobility.MobilityManagementSystem | MMS name is mms2@iiop://pc99:9000/acc
mobility.MobilityManagementSystem | Supported mobility protocol: simple-migratio
n-protocol
mobility.MobilityManagementSystem | Supported mobility system: FIPA-OS
mobility.MobilityManagementSystem | Supported system major version: 1.03
mobility.MobilityManagementSystem | Supported system minor version: 1.03
mobility.MobilityManagementSystem | Supported system dependencies: MobilityManag
ementSystem
mobility.MobilityManagementSystem | Supported language: Java
mobility.MobilityManagementSystem | Supported language major version: 1.2.2
mobility.MobilityManagementSystem | Supported language minor version: 1.2
mobility.MobilityManagementSystem | Supported language format: bytecode
mobility.MobilityManagementSystem | Supported hardware: Java
mobility.MobilityManagementSystem | "request" message received
mobility.parser.xml.MobileAgentDescriptionHandler | Start parsing XML document
mobility.parser.xml.MobileAgentDescriptionHandler | End XML document
mobility.MobilityManagementSystem | Message is request for move!
mobility.MobilityManagementSystem | Profile provided
mobility.MobilityManagementSystem | System provided
mobility.MobilityManagementSystem | - name FIPA-OS accepted
mobility.MobilityManagementSystem | - major-version 1.03 accepted
mobility.MobilityManagementSystem | - minor-version 1.03 accepted
mobility.MobilityManagementSystem | - dependency MobilityManagementS
ystem accepted
mobility.MobilityManagementSystem | Language provided
mobility.MobilityManagementSystem | - name Java accepted
mobility.MobilityManagementSystem | Wrong major vesion (1.1.8) - version support
ed is 1.2.2
mobility.MobilityManagementSystem | Sent "refuse"!
Start
Exploring - H:\mobility\code
Command Prompt
Command Prompt - m...
Microsoft Word - Appendix...
10:29 AM

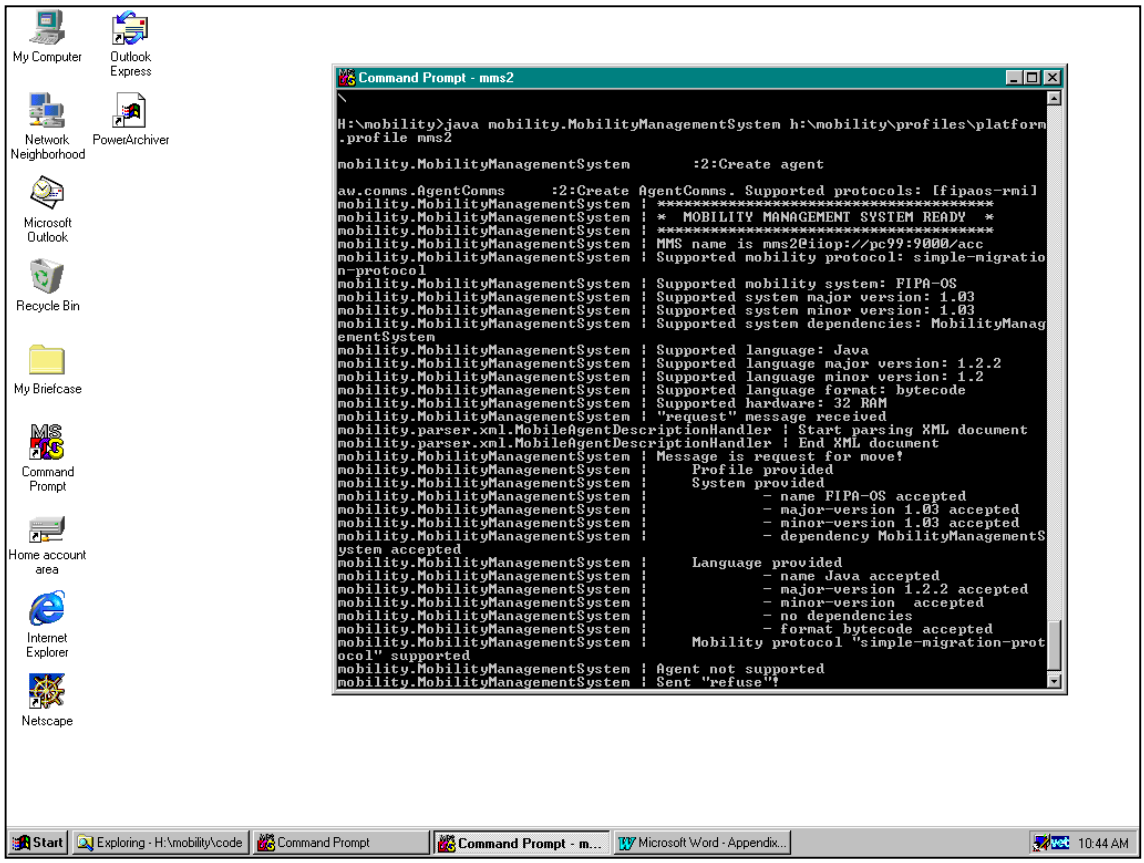
2.5


My Computer
Outlook Express
Network Neighborhood
PowerArchiver
Microsoft Outlook
Recycle Bin
My Briefcase
Command Prompt
Home account area
Internet Explorer
Netscape
Command Prompt - mms2
H:\mobility>SET PATH=.;s:\MSc\jdk1.2\bin\;x:\djgpp\bin;h:\mobility\voyager\bin
\
H:\mobility>java mobility.MobilityManagementSystem h:\mobility\profiles\platform
.profile mms2
mobility.MobilityManagementSystem :2:Create agent
aw.comms.AgentComms :2:Create AgentComms. Supported protocols: [fipaos-rmi]
mobility.MobilityManagementSystem | **
mobility.MobilityManagementSystem | * MOBILITY MANAGEMENT SYSTEM READY *
mobility.MobilityManagementSystem | **
mobility.MobilityManagementSystem | MMS name is mms2@iiop://pc99:9000/acc
mobility.MobilityManagementSystem | Supported mobility protocol: simple-migratio
n-protocol
mobility.MobilityManagementSystem | Supported mobility system: FIPA-OS
mobility.MobilityManagementSystem | Supported system major version: 1.03
mobility.MobilityManagementSystem | Supported system minor version: 1.03
mobility.MobilityManagementSystem | Supported system dependencies: MobilityManag
ementSystem
mobility.MobilityManagementSystem | Supported language: Java
mobility.MobilityManagementSystem | Supported language major version: 1.2.2
mobility.MobilityManagementSystem | Supported language minor version: 1.2
mobility.MobilityManagementSystem | Supported language format: bytecode
mobility.MobilityManagementSystem | Supported hardware: Java
mobility.MobilityManagementSystem | "request" message received
mobility.parser.xml.MobileAgentDescriptionHandler | Start parsing XML document
mobility.parser.xml.MobileAgentDescriptionHandler | End XML document
mobility.MobilityManagementSystem | Message is request for move!
mobility.MobilityManagementSystem | Profile provided
mobility.MobilityManagementSystem | System provided
mobility.MobilityManagementSystem | - name FIPA-OS accepted
mobility.MobilityManagementSystem | - major-version 1.03 accepted
mobility.MobilityManagementSystem | - minor-version 1.03 accepted
mobility.MobilityManagementSystem | - dependency MobilityManagementS
ystem accepted
mobility.MobilityManagementSystem | Language provided
mobility.MobilityManagementSystem | - name Java accepted
mobility.MobilityManagementSystem | - major-version 1.2.2 accepted
mobility.MobilityManagementSystem | - minor-version accepted
mobility.MobilityManagementSystem | - no dependencies
mobility.MobilityManagementSystem | Wrong format (source) - version supported is
bytecode
mobility.MobilityManagementSystem | Sent "refuse"!
Start
Exploring - H:\mobility\code
Command Prompt
Command Prompt - m...
Microsoft Word - Appendix...
10:34 AM

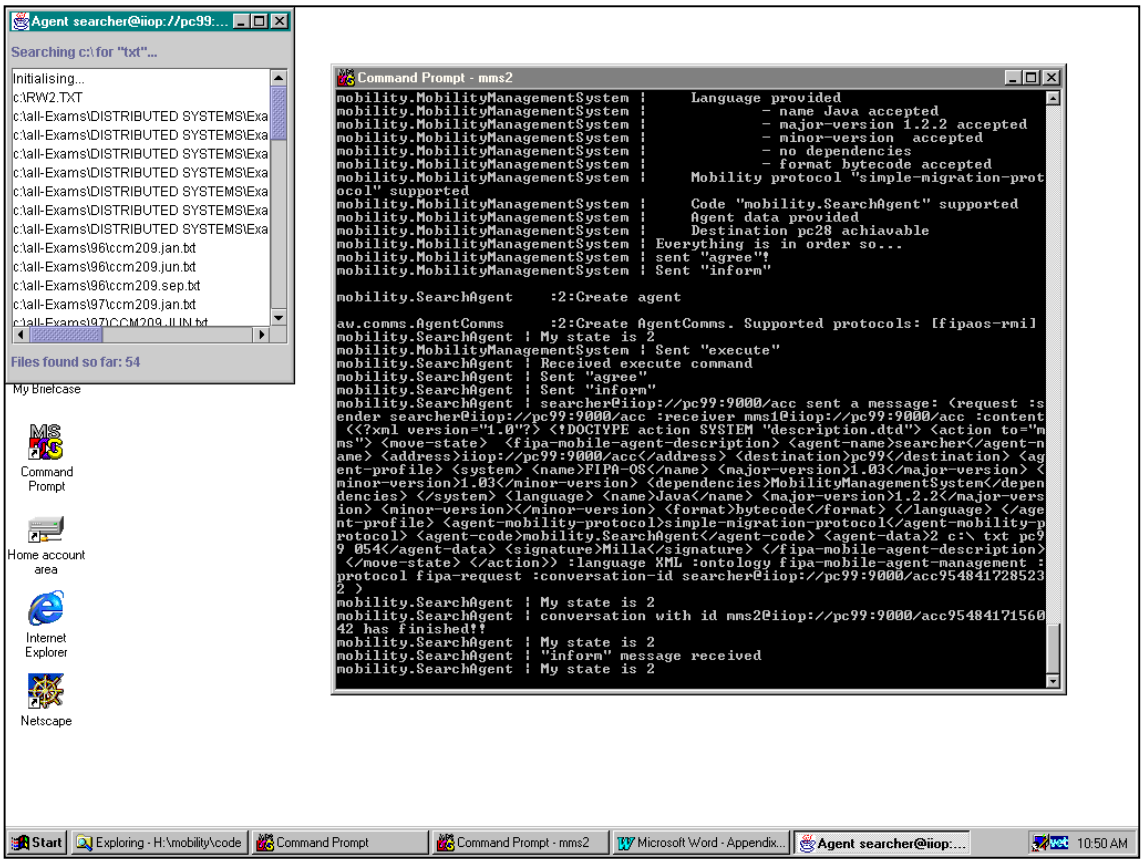
2.6

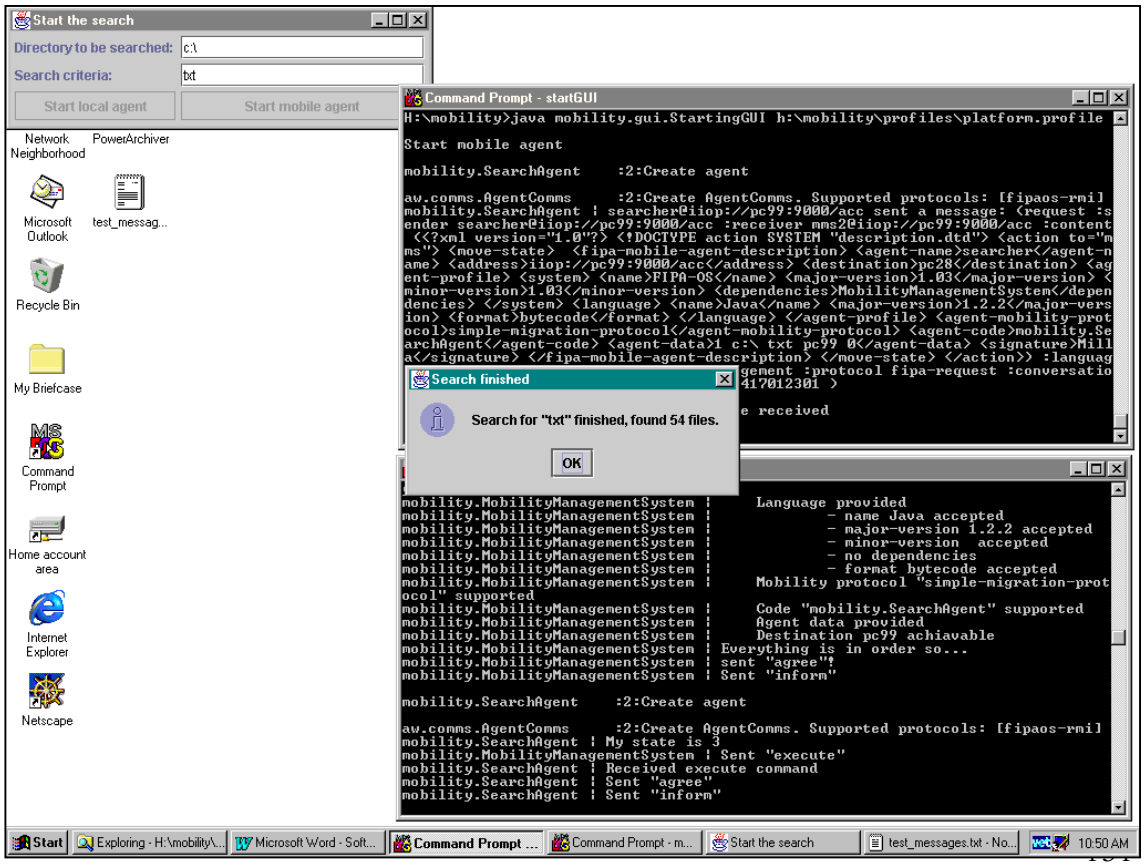

My Computer
Outlook Express
Network Neighborhood
PowerArchiver
Microsoft Outlook
Recycle Bin
My Briefcase
Command Prompt
Home account area
Internet Explorer
Netscape
Command Prompt - mms2
H:\mobility>SET PATH=.;s:\MSc\jdk1.2\bin\;x:\djgpp\bin;h:\mobility\voyager\bin
\
H:\mobility>java mobility.MobilityManagementSystem h:\mobility\profiles\platform
.profile mms2
mobility.MobilityManagementSystem :2:Create agent
aw.comms.AgentComms :2:Create AgentComms. Supported protocols: [fipaos-rmi]
mobility.MobilityManagementSystem | **
mobility.MobilityManagementSystem | * MOBILITY MANAGEMENT SYSTEM READY *
mobility.MobilityManagementSystem | **
mobility.MobilityManagementSystem | MMS name is mms2@iiop://pc99:9000/acc
mobility.MobilityManagementSystem | Supported mobility protocol: simple-migratio
n-protocol
mobility.MobilityManagementSystem | Supported mobility system: FIPA-OS
mobility.MobilityManagementSystem | Supported system major version: 1.03
mobility.MobilityManagementSystem | Supported system minor version: 1.03
mobility.MobilityManagementSystem | Supported system dependencies: MobilityManag
ementSystem
mobility.MobilityManagementSystem | Supported language: Java
mobility.MobilityManagementSystem | Supported language major version: 1.2.2
mobility.MobilityManagementSystem | Supported language minor version: 1.2
mobility.MobilityManagementSystem | Supported language format: bytecode
mobility.MobilityManagementSystem | Supported hardware: 32 BAM
mobility.MobilityManagementSystem | "request" message received
mobility.parser.xml.MobileAgentDescriptionHandler | Start parsing XML document
mobility.parser.xml.MobileAgentDescriptionHandler | End XML document
mobility.MobilityManagementSystem | Message is request for move!
mobility.MobilityManagementSystem | Profile provided
mobility.MobilityManagementSystem | System provided
mobility.MobilityManagementSystem | - name FIPA-OS accepted
mobility.MobilityManagementSystem | - major-version 1.03 accepted
mobility.MobilityManagementSystem | - minor-version 1.03 accepted
mobility.MobilityManagementSystem | - dependency MobilityManagementS
ystem accepted
mobility.MobilityManagementSystem | Language provided
mobility.MobilityManagementSystem | - name Java accepted
mobility.MobilityManagementSystem | - major-version 1.2.2 accepted
mobility.MobilityManagementSystem | - minor-version accepted
mobility.MobilityManagementSystem | - no dependencies
mobility.MobilityManagementSystem | - format bytecode accepted
mobility.MobilityManagementSystem | Mobility protocol not supported
mobility.MobilityManagementSystem | Sent "refuse"!
Start
Exploring - H:\mobility\code
Command Prompt
Command Prompt - m...
Microsoft Word - Appendix...
10:41 AM

2.7

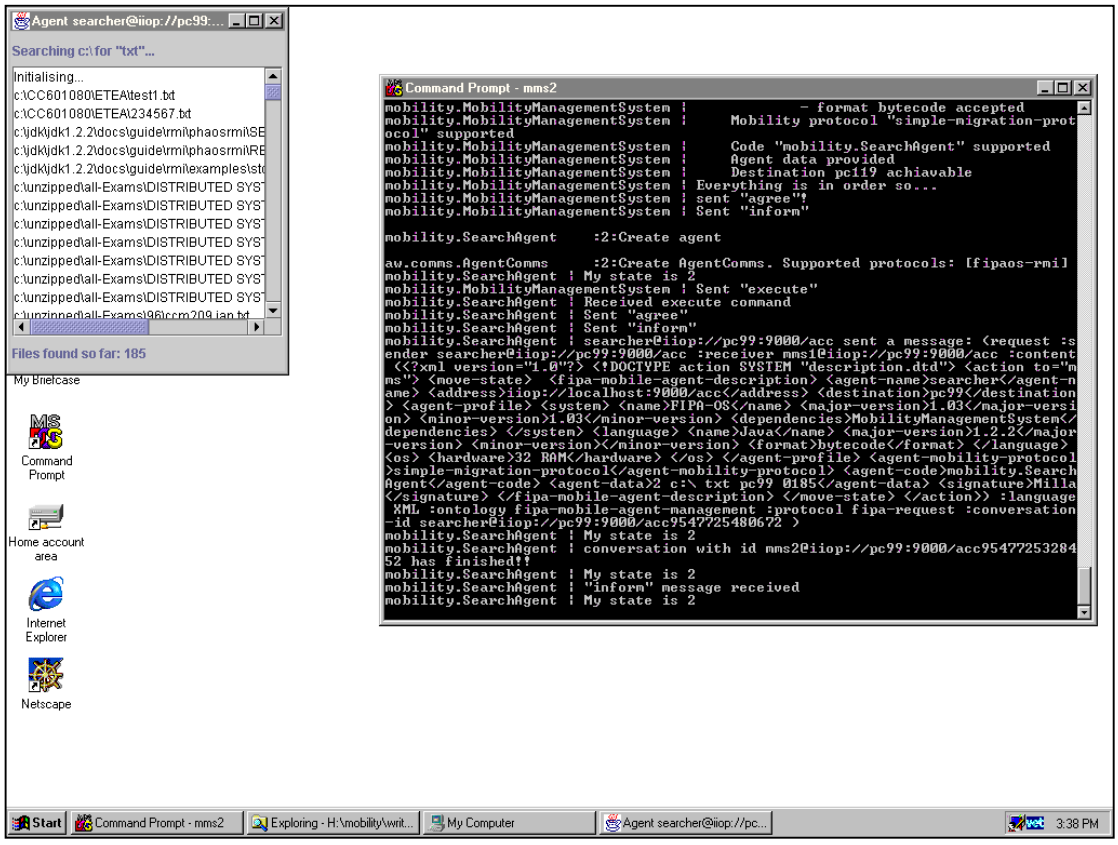

My Computer
Outlook Express
Network Neighborhood
PowerArchiver
Microsoft Outlook
Recycle Bin
My Briefcase
Command Prompt
Home account area
Internet Explorer
Netscape
Command Prompt - mms2
H:\mobility>java mobility.MobilityManagementSystem h:\mobility\profiles\platform.profile mms2
mobility.MobilityManagementSystem :2:Create agent
aw.comms.AgentComms :2:Create AgentComms. Supported protocols: [fipaos-rmi]
mobility.MobilityManagementSystem : **
mobility.MobilityManagementSystem : * MOBILITY MANAGEMENT SYSTEM READY *
mobility.MobilityManagementSystem : **
mobility.MobilityManagementSystem : MMS name is mms2@iiop://pc99:9000/acc
mobility.MobilityManagementSystem : Supported mobility protocol: simple-migration-protocol
mobility.MobilityManagementSystem : Supported mobility system: FIPA-OS
mobility.MobilityManagementSystem : Supported system major version: 1.03
mobility.MobilityManagementSystem : Supported system minor version: 1.03
mobility.MobilityManagementSystem : Supported system dependencies: MobilityManagementSystem
mobility.MobilityManagementSystem : Supported language: Java
mobility.MobilityManagementSystem : Supported language major version: 1.2.2
mobility.MobilityManagementSystem : Supported language minor version: 1.2
mobility.MobilityManagementSystem : Supported language format: bytecode
mobility.MobilityManagementSystem : Supported hardware: 32 RAM
mobility.MobilityManagementSystem : "request" message received
mobility.parser.xml.MobileAgentDescriptionHandler : Start parsing XML document
mobility.parser.xml.MobileAgentDescriptionHandler : End XML document
mobility.MobilityManagementSystem : Message is request for move!
mobility.MobilityManagementSystem : Profile provided
mobility.MobilityManagementSystem : System provided
mobility.MobilityManagementSystem : - name FIPA-OS accepted
mobility.MobilityManagementSystem : - major-version 1.03 accepted
mobility.MobilityManagementSystem : - minor-version 1.03 accepted
mobility.MobilityManagementSystem : - dependency MobilityManagementSystem accepted
mobility.MobilityManagementSystem : Language provided
mobility.MobilityManagementSystem : - name Java accepted
mobility.MobilityManagementSystem : - major-version 1.2.2 accepted
mobility.MobilityManagementSystem : - minor-version accepted
mobility.MobilityManagementSystem : - no dependencies
mobility.MobilityManagementSystem : - format bytecode accepted
mobility.MobilityManagementSystem : Mobility protocol "simple-migration-protocol" supported
mobility.MobilityManagementSystem : Agent not supported
mobility.MobilityManagementSystem : Sent "refuse"!
Start
Exploring - H:\mobility\code
Command Prompt
Command Prompt - m...
Microsoft Word - Appendix...
10:44 AM

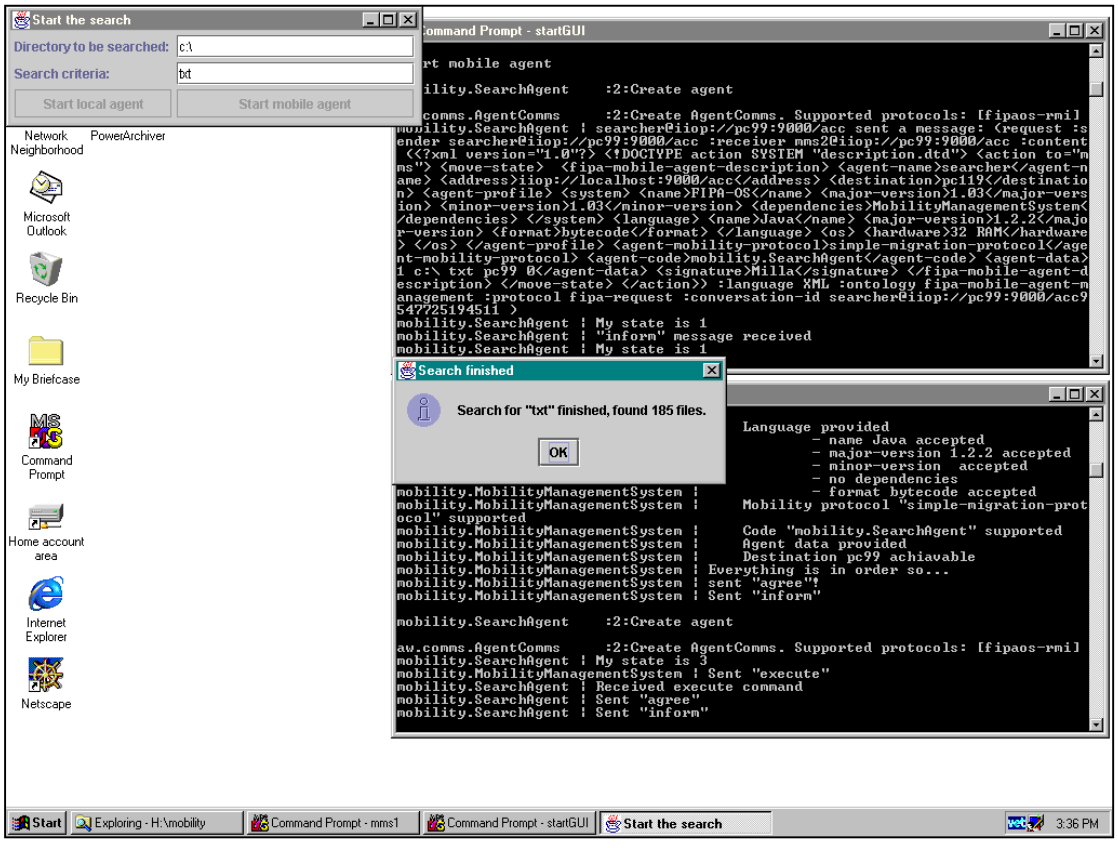
3.0


Agent searcher@iiop://pc99:...
Searching c:\ for "txt"...
Initialising...
c:\RW2.TXT
c:\all-Exams\DISTRIBUTED SYSTEMS\Exa
c:\all-Exams\DISTRIBUTED SYSTEMS\Exa
c:\all-Exams\DISTRIBUTED SYSTEMS\Exa
c:\all-Exams\DISTRIBUTED SYSTEMS\Exa
c:\all-Exams\DISTRIBUTED SYSTEMS\Exa
c:\all-Exams\DISTRIBUTED SYSTEMS\Exa
c:\all-Exams\DISTRIBUTED SYSTEMS\Exa
c:\all-Exams\96\ccm209.jan.txt
c:\all-Exams\96\ccm209.jun.txt
c:\all-Exams\96\ccm209.sep.txt
c:\all-Exams\97\ccm209.jan.txt
Files found so far: 54
Command Prompt - mms2
mobility.MobilityManagementSystem ! Language provided
- name Java accepted
- major-version 1.2.2 accepted
- minor-version accepted
- no dependencies
- format bytecode accepted
Mobility protocol "simple-migration-protocol" supported
Code "mobility.SearchAgent" supported
Agent data provided
Destination pc28 achievable
Everything is in order so...
sent "agree"!
Sent "inform"
mobility.SearchAgent :2:Create agent
aw.comms.AgentComms :2:Create AgentComms. Supported protocols: [fipaos-rmi]
mobility.SearchAgent ! My state is 2
mobility.MobilityManagementSystem ! Sent "execute"
mobility.SearchAgent ! Received execute command
mobility.SearchAgent ! Sent "agree"
mobility.SearchAgent ! Sent "inform"
mobility.SearchAgent ! searcher@iiop://pc99:9000/acc sent a message: (request :sender searcher@iiop://pc99:9000/acc :receiver mms1@iiop://pc99:9000/acc :content
(<?xml version="1.0"?> (!DOCTYPE action SYSTEM "description.dtd") (action to="mms") (move-state) (fipa-mobile-agent-description) (agent-name)searcher(/agent-name) (address)iiop://pc99:9000/acc(/address) (destination)pc99(/destination) (agent-profile) (system) (name)FIPA-OS(/name) (major-version)1.03(/major-version) (minor-version)1.03(/minor-version) (dependencies)MobilityManagementSystem(/dependencies) (/system) (language) (name)Java(/name) (major-version)1.2.2(/major-version) (minor-version)(/minor-version) (format)bytecode(/format) (/language) (/agent-profile) (agent-mobility-protocol)simple-migration-protocol(/agent-mobility-protocol) (agent-code)mobility.SearchAgent(/agent-code) (agent-data)2 c:\ txt pc99 054(/agent-data) (signature)Milla(/signature) (/fipa-mobile-agent-description) (/move-state) (/action)) :language XML :ontology fipa-mobile-agent-management :protocol fipa-request :conversation-id searcher@iiop://pc99:9000/acc954841728523 2)
mobility.SearchAgent ! My state is 2
mobility.SearchAgent ! conversation with id mms2@iiop://pc99:9000/acc95484171560 42 has finished!!
mobility.SearchAgent ! My state is 2
mobility.SearchAgent ! "inform" message received
mobility.SearchAgent ! My state is 2
My Briefcase
Command Prompt
Home account area
Internet Explorer
Netscape
Start
Exploring - H:\mobility\code
Command Prompt
Command Prompt - mms2
Microsoft Word - Appendix...
Agent searcher@iiop:...
10:50 AM


Start the search
Directory to be searched: c:\
Search criteria: txt
Start local agent
Start mobile agent
Command Prompt - startGUI
H:\mobility>java mobility.gui.StartingGUI h:\mobility\profiles\platform.profile
Start mobile agent
mobility.SearchAgent :2:Create agent
aw.comms.AgentComms :2:Create AgentComms. Supported protocols: [fipaos-rmi]
mobility.SearchAgent ! searcher@iiop://pc99:9000/acc sent a message: (request :sender searcher@iiop://pc99:9000/acc :receiver mms2@iiop://pc99:9000/acc :content
(<?xml version="1.0"?> (!DOCTYPE action SYSTEM "description.dtd") (action to="mms") (move-state) (fipa-mobile-agent-description) (agent-name)searcher(/agent-name) (address)iiop://pc99:9000/acc(/address) (destination)pc28(/destination) (agent-profile) (system) (name)FIPA-OS(/name) (major-version)1.03(/major-version) (minor-version)1.03(/minor-version) (dependencies)MobilityManagementSystem(/dependencies) (/system) (language) (name)Java(/name) (major-version)1.2.2(/major-version) (format)bytecode(/format) (/language) (/agent-profile) (agent-mobility-protocol)simple-migration-protocol(/agent-mobility-protocol) (agent-code)mobility.SearchAgent(/agent-code) (agent-data)1 c:\ txt pc99 0(/agent-data) (signature)Milla(/signature) (/fipa-mobile-agent-description) (/move-state) (/action)) :language
gement :protocol fipa-request :conversation
417012301)
e received
Search finished
Search for "txt" finished, found 54 files.
OK
mobility.MobilityManagementSystem ! Language provided
- name Java accepted
- major-version 1.2.2 accepted
- minor-version accepted
- no dependencies
- format bytecode accepted
Mobility protocol "simple-migration-protocol" supported
Code "mobility.SearchAgent" supported
Agent data provided
Destination pc99 achievable
Everything is in order so...
sent "agree"!
Sent "inform"
mobility.SearchAgent :2:Create agent
aw.comms.AgentComms :2:Create AgentComms. Supported protocols: [fipaos-rmi]
mobility.SearchAgent ! My state is 3
mobility.MobilityManagementSystem ! Sent "execute"
mobility.SearchAgent ! Received execute command
mobility.SearchAgent ! Sent "agree"
mobility.SearchAgent ! Sent "inform"
Network Neighborhood
PowerArchiver
Microsoft Outlook
test_messag...
Recycle Bin
My Briefcase
Command Prompt
Home account area
Internet Explorer
Netscape
Start
Exploring - H:\mobility\...
Microsoft Word - Soft...
Command Prompt ...
Command Prompt - m...
Start the search
test_messages.txt - No...
10:50 AM

3.1


Agent searcher@iiop://pc99:...
Searching c:\ for "txt"...
Initialising...
c:\CC601080\ETEA\test1.txt
c:\CC601080\ETEA\234567.txt
Files found so far: 185
Command Prompt - mms2
mobility.MobilityManagementSystem ! Everything is in order so...
mobility.MobilityManagementSystem ! sent "agree"!
mobility.MobilityManagementSystem ! Sent "inform"
mobility.SearchAgent :2:Create agent
mobility.SearchAgent ! My state is 2
mobility.MobilityManagementSystem ! Sent "execute"
mobility.SearchAgent ! Received execute command
mobility.SearchAgent ! Sent "agree"
mobility.SearchAgent ! Sent "inform"
mobility.SearchAgent ! My state is 2
mobility.SearchAgent ! conversation with id mms2@iiop://pc99:9000/acc954772538452 has finished!!
mobility.SearchAgent ! My state is 2
mobility.SearchAgent ! "inform" message received
mobility.SearchAgent ! My state is 2
My Briefcase
Command Prompt
Home account area
Internet Explorer
Netscape
Start
Command Prompt - mms2
Exploring - H:\mobility\writ...
My Computer
Agent searcher@iiop://pc...
3:38 PM


Start the search
Directory to be searched: c:\
Search criteria: txt
Start local agent
Start mobile agent
Command Prompt - startGUI
Search finished
Search for "txt" finished, found 185 files.
OK
mobility.SearchAgent ! My state is 1
mobility.SearchAgent ! "inform" message received
mobility.SearchAgent ! My state is 1
Network Neighborhood
PowerArchiver
Microsoft Outlook
Recycle Bin
My Briefcase
Command Prompt
Home account area
Internet Explorer
Netscape
Start
Exploring - H:\mobility
Command Prompt - mms1
Command Prompt - startGUI
Start the search
3:36 PM