JADE
A White Paper

F. Bellifemine, G. Caire, A. Poggi, G. Rimassa

ABSTRACT - This white paper gives an overview of the JADE platform, presents its architecture and main functionalities, and outlines the conceptual model underlying JADE. Two major aspects of the conceptual model are presented: distributed system topology with peer-to-peer networking, and software component architecture with agent paradigm. The network topology affects how the various components are linked together, whereas the component architecture specifies what the components are supposed to expect from one another. The relevance of standards for software interoperability, and in particular the compliance with FIPA, is also highlighted.

The paper tries also to address the vital technology transfer issue, which is crucial for a system such as JADE, moving right now from software research towards advanced business applications. Recognizing that technology transfer is largely a people centered process, the paper addresses the organization of the people using and developing the JADE core, its extensions and applications leveraging JADE infrastructure. The two main institutions dealing with JADE are outlined: the open source community and the JADE governing board.

Finally, the main feats of the JADE approach are summed up, with the intent of helping readers deciding whether JADE can fulfill their needs and in which application domains JADE can prove most useful.

JADE:
Java Agent
DEvelopment
Framework

Introduction

JADE (1,9) is an enabling technology, a ***middleware*** for the development and run-time execution of ***peer-to-peer*** applications which are based on the ***agents*** paradigm and which can seamless work and interoperate both in wired and wireless environment.

In order to understand this definition, the odd terms it includes, and the context, Section "Reference technologies" introduces some reference technologies. Two major aspects of the conceptual

fabio.bellifemine@telecomitalia.it

F. Bellifemine

Fabio L. Bellifemine is a senior project manager at the Department of Services and Multi-media of TILAB, Torino. He graduated in Computer Science from the University of Torino in 1988 and, prior to joining TILAB, until 1994 he held a researcher position at the Italian National Research Council. His research interests covered several aspects of multi-media, including signal processing and video compression, software architecture, system integration, applications, user modeling and software agents.
Since 1997, he is interested in the multi-agent system research and he is involved in the FIPA standardization body where he currently chairs the FIPA Architecture Board. In 1999 and 2002 he received a diploma from FIPA for its outstanding contribution to the activity. He participated to several research projects, and he leaded the FACTS Work Package that, in April 2000, received an award for achieving first place in the FIPA competition for agent-based applications. He is the leader of the JADE project and author of over 50 publications in proceedings of international conferences and journal papers. He is PC member of several workshops and conferences related to multi-agent system and user modeling research.

model are presented: distributed system topology with peer-to-peer networking, and software component architecture with agent paradigm. The network topology affects how the various components are linked together, whereas the component architecture specifies what the components are supposed to expect from one another. The section also provides some information about the concept of middleware and the advantages in application development, and about the profiling structure of the Java technology, which is the programming language of JADE and of the applications using JADE.
Section "What is JADE?" describes the JADE platform, its main functionalities, the architectural model, and some technical information.
JADE is an Open Source project around which a community of users and contributors has grown up, and recently also an International Governing Board has been formed. Section "The JADE Community" outlines this community, describes the open source project and how its organization is evolving through the creation of the Governing Board.
Finally, Section "Why using JADE?" presents some considerations to highlight the advantages of JADE and which application domains it can prove most useful as enabling technology.

REFERENCE TECHNOLOGIES

The peer-to-peer model

"Client-Server" (C/S) is the reference model, well-known and widely-diffused, for distributed applications. The model is based on a rigid distinction of roles between the client nodes (resource requester) and the server nodes (resource providers). The server nodes provide the services, more in general the capabilities of the distributed system, but they are not capable of taking any initiative as they are fully reactive and they can just wait for being invocated by the client nodes. Client nodes, as opposite, concentrate all the initiative of the system: they access and use the services, typically, but not necessarily, upon user requests, but they do never provide any capability.
Clients can appear and disappear at any time; generally, they have dynamic addresses, while servers must typically provide some guarantees of stability and generally listen to a well-known and static address.
Clients communicate with the servers, but they cannot communicate with other clients. On the other hand, server cannot communicate with their clients until the clients have taken the initiative and decided to activate a communication session with the server.
The web is a typical example of application based on the client-server model. The servers are the *sites/portals*, which own the entire application logic and information resources. The clients are the *browsers,* only a tool to manage the interface with the user and whose only task is to retrieve, upon explicit user request, information located on Internet sites and to present (render) it.
A large family of distributed applications exist, however, that are not well adapted to this model.

For instance, a simple “chat” application, as well as a distributed file sharing system (such as Napster or Gnutella) or a multiplayer game, require the active nodes on the users terminals to be capable of communicating each other. Even if implementing such an application by using the client-server model is still possible (and it is also quite often done), it will loose advantages of better software practices and architectures. In the case of the “chat” application, for instance, a user client should send messages to a central server from which they should be retrieved by the client of another user: the server is not necessary but it is there just as an implementation artefact while the “peer-to-peer” (P2P) model would have been more appropriate.

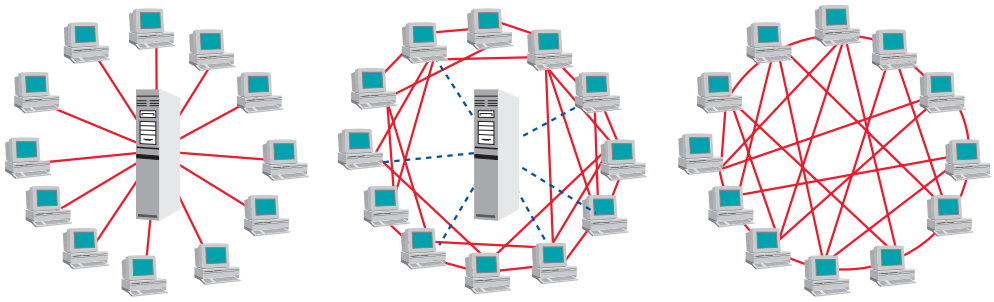
In the peer-to-peer model, in fact, there is no more any distinction of roles and each peer is capable of a mix of initiative and capability: each node can initiate the communication, be subject or object of a request, be proactive, provide capabilities; the application logics is no more concentrated on the server but distributed between all the peers of the network; each node is capable of discover each other, it can enter, join or leave the network anywhere anytime. The system is fully distributed as well as the value of the service is distributed across the network and new business models might be enabled.

An important consequence of the differences between the 2 models is the way the nodes can be discovered. In the C/S systems, clients must know their servers but they do not need to know other clients (of course, given that client-to-client communication is never expected to happen). In P2P systems, who-knows-whom is fully arbitrary and the system must provide proper services that allow peers to enter, join, or leave the network at any time as well as to search and discover other peers. These services are usually the white and yellow page mechanisms that allow publishing and discovering the features and the services offered by a peer. On the basis of the implementation of these mechanisms, two basic P2P network models can be identified (see figure 1): pure P2P networks (also called decentralized), and hybrid P2P networks (also called with central index).

A pure P2P network is fully decentralized and the peers are fully autonomous. The absence of any reference node makes more difficult to maintain the coherence of the network and the discovery of the peers, with a complexity and bandwidth that tends to grow exponentially with the number of nodes. Also security is quite demanding as each node is entitled to join the network without any control mechanism.

The hybrid architectures, instead, are based on a special node that provides a service that simplifies the look-up and discovery of the active peers, their list of capabilities, and their list of provided services. These types of networks, usually, generate less traffic and are more secure as they tend to require al-

Figure 1
Client/Server (left), pure P2P (right), hybrid P2P (centre)



a) Client/Server b) Hybrid P2P c) Pure P2P

so the registration and authentication of the peers. On the other hand, their functioning depends on the availability of the index nodes that might become a central point of failure and attack.

The agents paradigm

The agents paradigm applies concepts from artificial intelligence and speech act theory to the distributed object technology. The paradigm is based on the *agent* abstraction, a software component that is autonomous, proactive and social:

- *autonomous*: agents have a degree of control on their own actions, they own their thread of control and, under some circumstances, they are also able to take decisions;
- *proactive:* agents do not only react in response to external events (i.e. a remote method call) but they also exhibit a goal-directed behaviour and, where appropriate, are able to take initiative;
- *social:* agents are able to, and need to, interact with other agents in order to accomplish their task and achieve the complete goal of the system.

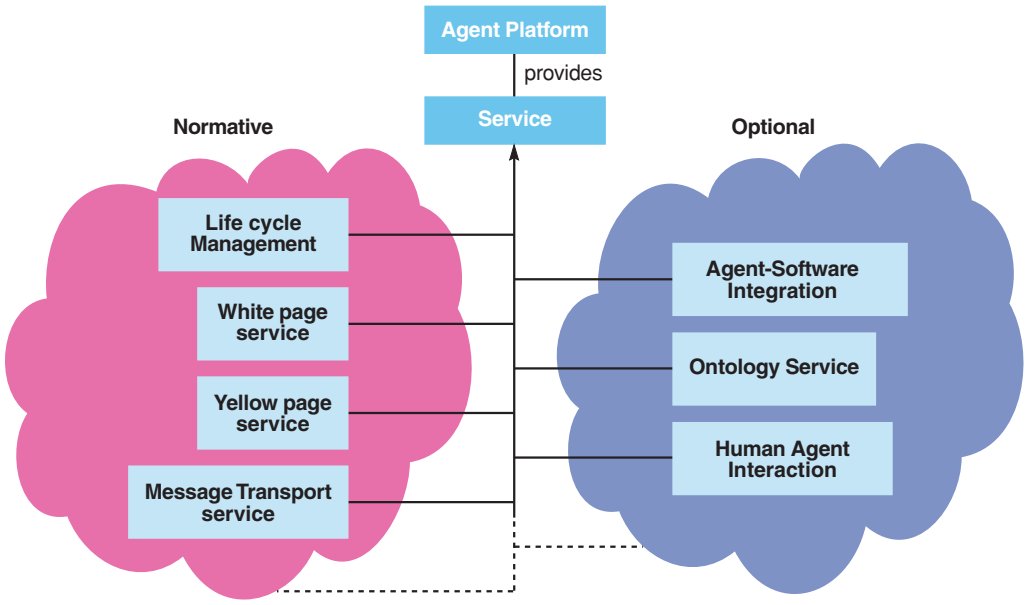
Agent-based systems are intrinsically peer-to-peer: each agent is a peer that potentially needs to initiate a communication with any other agent as well as it is capable of providing capabilities to the rest of the agents. The role of the communication is very important in an agent-based system, and its model is based on three main features:

1. *agents are active entities, they can say 'no', and they are loosely coupled*. This set of interrelated properties is the basis for the choice of message-based asynchronous communication between agents instead of remote procedure call: an agent wishing to communicate has just to send a message to a certain destination. This modality of communication, in fact, allows the receiver to select which messages to serve and which to discard, as well as which messages to serve first and which later in time. It also allows the sender to control its thread of execution and not to be blocked until the receiver reads and serves the message. Finally, it also removes any temporal dependency between the sender and the receiver: the receiver might not be available at the time the sender sends the message, or it might even not exist at that time, or, also, it might even be not known by the sender that, instead, defines the receiver intensionaly (e.g. all agents interested into 'football') or mediates the communication through a proxy (e.g. propagate this message to all agents in the domain X).
2. *agents perform actions and communication is just a type of action.* Making communication at the same level of actions allows an agent, for instance, to reason about a plan that includes both physical actions (e.g. turning on the left) and communicative actions (e.g. asking to open the door). In order to make communication plannable, effects and preconditions of each possible communication needs to be clearly defined.
3. *communication carries a semantics meaning*. When an agent is the object of a communicative action (i.e. when it receives a message), it must be able to properly understand the meaning of that action and, in particular, why that action has been performed (i.e. the communicative intention of the sender of the message). This property turns into the needs for a universal semantics and the need for a standard.

Inspired by the vision that "agents will remain just a dream if end-to-end interoperability across different manufacturers and operators is not preserved" (2), in 1996 TILAB (formerly CSELT) promoted the creation of FIPA (Foundation for Intelligent Physical Agents) (3), an international non-profit association of companies and organizations sharing the goal and the effort to produce standard specifications for agent technology. TILAB, and in particular the JADE team, supported and leaded at several levels this initiative starting from the presidency of Leonardo Chiariglione, continuing with the editing of specifications and the leadership of Technical Committees and of the FIPA Architecture Board. JADE also participated with success to both FIPA Interoperability Tests, in 1999 and in 2001.

Based on the first set of specifications released in 1997, at the end of 2002 FIPA finally released the standard. The standard targets interoperability and, as a consequence, it focuses on the external behaviour of the system components, leaving

Figure 2
FIPA Standard: Services provided by a Platform



open the implementation details and the internal architectures. In fact, the internal architecture of JADE is unique even if it fully complies with FIPA. The FIPA standard fully embraces the agent paradigm and, in particular, it defines the reference model of an agent platform and a set of services that should be provided. The collection of these services, and their standard interfaces, represents the normative rules that allow a society of agents to exist, operate, and be managed.

Being agents social and needing to communicate, the Agent Communication Language is one of the main assets of the FIPA standard. The FIPA ACL is based on the speech act theory and on the assumptions and requirements of the agents paradigm described above. FIPA standardized an extensible library of 22 communicative acts that allow representation of different communicative intentions (such as requesting, proposing, informing, querying, calling for a proposal, refusing,...). FIPA also defined the structure of a message that allows to represent and convey information useful to identify sender and receivers, the content of the message and its properties (e.g. the encodings and the representation language), and, in particular, information useful to identify and follow threads of conversation between agents and to represent timeouts for the communication. Common patterns of conversations have been also defined by FIPA, the so-called interaction protocols, that provide agents with a library of patterns to achieve common tasks, such as delegating an action, calling for a proposal,... Figure 3 represents the model of communication defined by FIPA and the relationships between its composing elements. One of the main assets of these FIPA standards is their 'standard' status, defined and accepted by the agent community.

The middleware

The term *middleware* is meant to describe all those higher-level libraries that enable easier and more effective application development by providing generic services useful not just for a single application but rather for a variety of applications, for instance communication, data access, encodings, resource control. These same services are provided by operative systems, but the idea behind middleware is to provide better, OS independent APIs aggregating native facilities into

Figure 3
FIPA Standard: components of the communication model

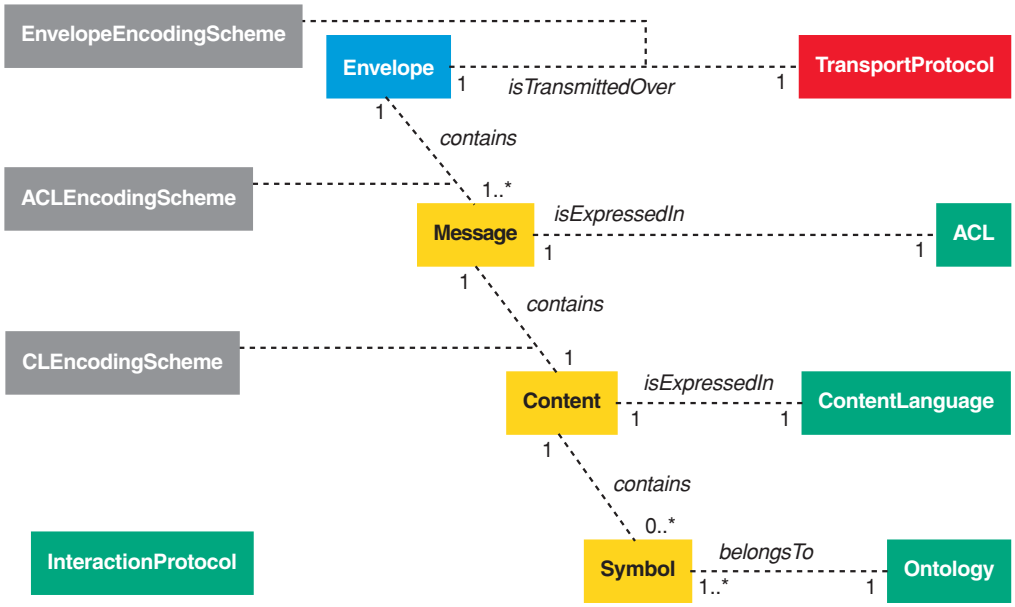
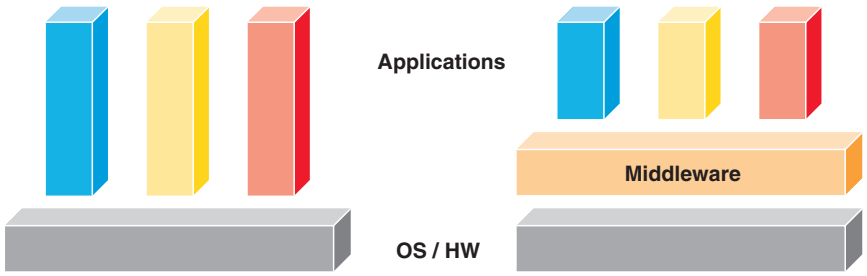


Figure 4
Role of the middleware: "vertical" approach (left) vs. "horizontal" approach (right)



simple-to-reuse building blocks. On the other hand, the implementation of these services often might require considerable time, if not the same, in respect to the development of the application logics. The capability of reusability across several application domains suggest the name of 'horizontal' approach as opposed to 'vertical' approach where an ad-hoc solution for a specific application should be provided. Middleware-based approaches allows to reduce footprint and development time of applications.

The Java technology

An overview of the Java technology is out of the scope of this paper but, in order to better understand some of the features of JADE and its relationships with the Java world, it is important to remember that the Java technology is structured into 4 *editions* (as named by Sun itself) according to the target device and the expected supported functionalities: server-based applications (J2EE), desktop-type applications (J2SE), portable and mobile-phone devices (J2ME), SIM/smart-card devices (Java Card). JADE has been implemented fully in Java language and, at the time of writing this paper, it can be seamless executed on every type of Java Virtual Machine with exception of the Java Card.

WHAT IS JADE?

JADE (1,9) is the middleware developed by TILAB for the development of distributed multi-agent applications based on the peer-to-peer communication architecture. Both the intelligence, the initiative, the information, the resources and the control can be fully distributed on mobile terminals as well as on computers in the fixed network. The environment can evolve dynamically with *peers*, that in JADE are called *agents*, that appear and disappear in the system according to the needs and the requirements of the application environment. Communication between the peers, regardless of whether they are running in the wireless or wireline network, is completely symmetric with each peer being able to play both the initiator and the responder role.

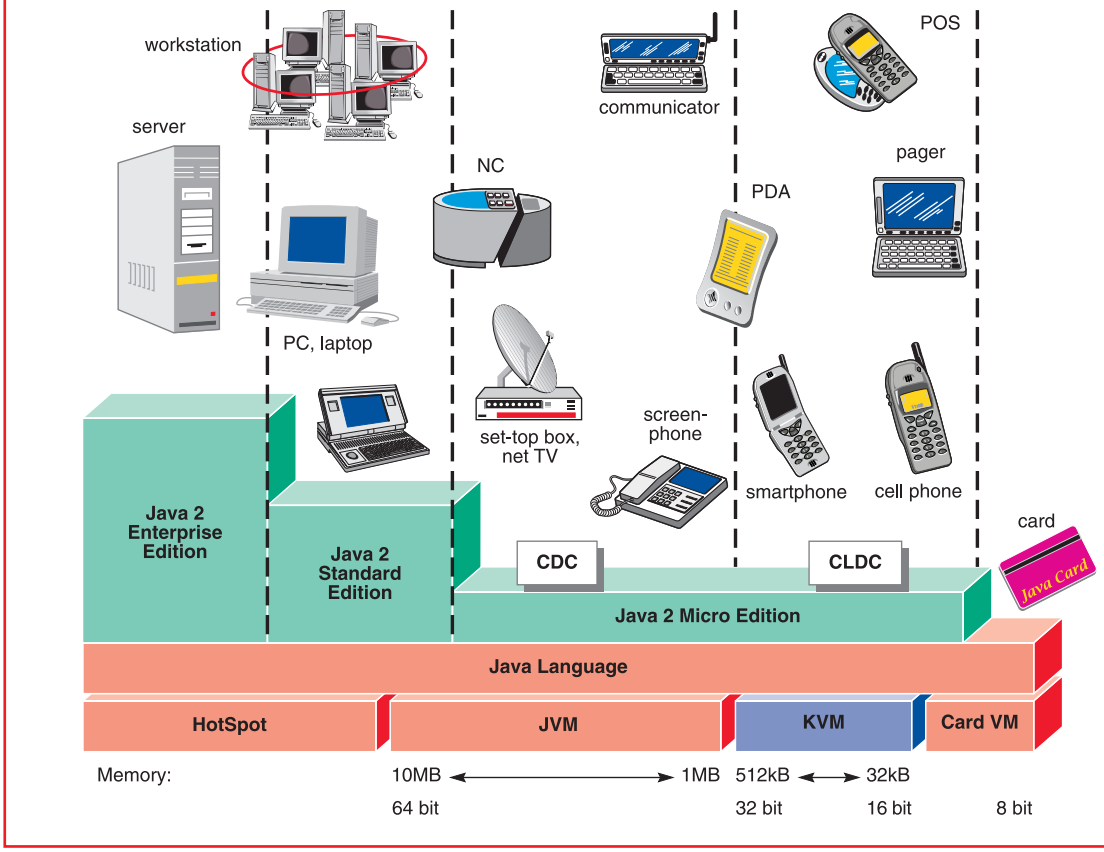
JADE is fully developed in **Java** and is based of the following driving principles:

- **Interoperability** – JADE is compliant with the FIPA specifications (3). As a consequence, JADE agents can interoperate with other agents, provided that they comply with the same standard.
- **Uniformity and portability** – JADE provides a homogeneous set of APIs that are independent from the underlying network and Java version. More in details, the JADE run-time provides the same APIs both for the J2EE, J2SE and J2ME environment. In theory, application developers could decide the Java run-time environment at deploy-time.
- **Easy to use** – The complexity of the middleware is hidden behind a simple and intuitive set of APIs.
- **Pay-as-you-go philosophy** – Programmers do not need to use all the features provided by the middleware. Features that are not used do not require programmers to know anything about them, neither add any computational overhead.

The Architectural model

JADE includes both the libraries (i.e. the Java classes) required to develop application agents and the run-time environment that provides the basic

Figure 5
The Java technology



services and that must be active on the device before agents can be executed. Each instance of the JADE run-time is called *container* (since it "contains" agents). The set of all containers is called platform and provides a homogeneous layer that hides to agents (and to application developers also) the complexity and the diversity of the underlying tires (hardware, operating systems, types of network, JVM).

As depicted in figure 6, JADE is compatible with the J2ME CLDC/MIDP1.0 environment. It has already been tested on the fields over the GPRS network with different mobile terminals among which: Nokia 3650, Motorola Accompli008, Siemens SX45, PalmVx, Compaq iPaq, Psion5MX, HP Jornada 560. The JADE run-time memory footprint, in a MIDP1.0 environment, is around 100 KB, but can be further reduced until 50 KB using the ROMizing technique (4), i.e. compiling JADE together with the JVM. JADE is extremely versatile and therefore, not only it fits the constraints of environments with limited resources, but it has already been integrated into complex architectures such as.NET or J2EE (5) where JADE becomes a service to execute multi-party proactive applications. The **limited memory footprint** allows installing JADE on all mobile phones provided that they are Java-enabled.

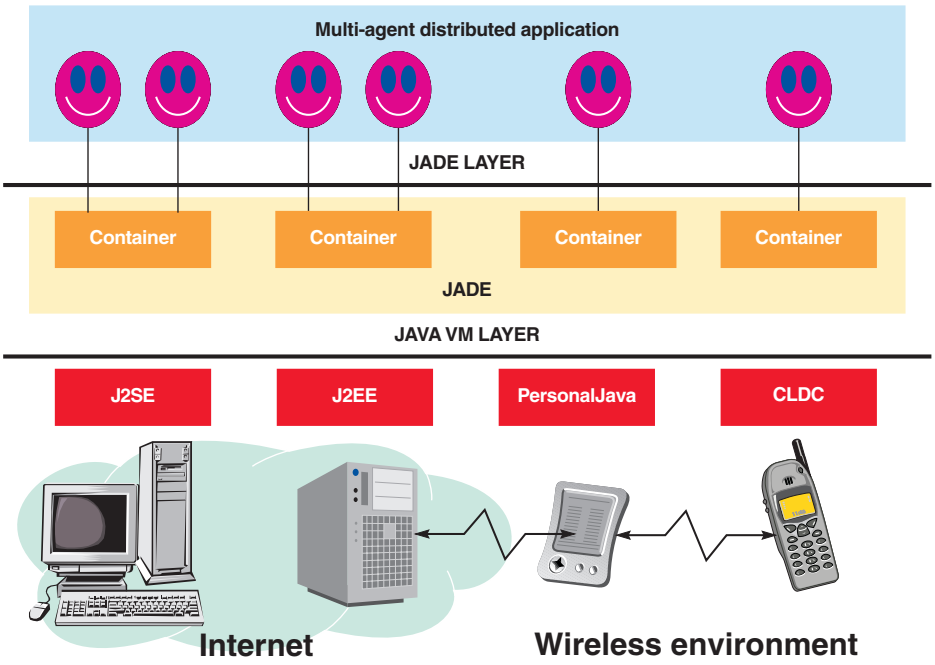
An analysis and a **benchmark of Scalability and Performance** of the JADE Message Transport System is reported in (10).

The Functional model

From the functional point of view, JADE provides the basic services necessary to distributed peer-to-peer applications in the fixed and mobile environment. JADE allows each agent to **dynamically discover** other agents and to **communicate** with them according to the peer-to-peer paradigm. From the application point of view, each agent is identified by a unique name and provides a set of services. It can register and modify its services and/or search for agents providing given services, it can control its life cycle and, in particular, communicate with all other peers.

Agents communicate by exchanging asynchronous messages, a communication model almost universally accepted for **distributed and loosely-**

Figure 6
The JADE architecture



coupled communications[1], i.e. between heterogeneous entities that do not know anything about each other. In order to communicate, an agent just sends a message to a destination. Agents are identified by a name (no need for the destination object reference to send a message) and, as a consequence, there is no temporal dependency between communicating agents. The sender and the receiver could not be available at the same time. The receiver may not even exist (or not yet exist) or could not be directly known by the sender that can specify a property (e.g. "all agents interested in football") as a destination. Because agents identifies each other by their name, hot change of their object reference are transparent to applications.

Despite this type of communication, **security** is preserved, since, for applications that require it, JADE provides proper mechanisms to authenticate and verify "rights" assigned to agents. When needed, therefore, an application can verify the identity of the sender of a message and prevent actions not allowed to perform (for instance an agent may be allowed to receive messages from the agent representing the boss, but not to send messages to it). All messages exchanged between agents are carried out within an *envelope* including only the information required by the transport layer. That allows, among others, to encrypt the content of a message separately from the envelope.

The structure of a message complies with the ACL language defined by FIPA (3) and includes fields, such as variables indicating the context a message refers-to and timeout that can be waited before an answer is received, aimed at supporting complex interactions and multiple parallel conversations. To further support the implementation of **complex conversations**, JADE provides a set of skeletons of typical interaction patterns to perform specific tasks, such as negotiations, auctions and task delegation. By using these skeletons (implemented as Java abstract classes), programmers can get rid of the burden of dealing with synchronization issues, timeouts, error conditions and, in general, all those aspects that are not strictly related to the application logic.

To facilitate the creation and handling of messages **content**, JADE provides support for automatically converting back and forth between the format suitable for content exchange, including XML and RDF, and the format suitable for content manipulation (i.e. Java objects). This support is integrated with some ontology creation tools, e.g. Protégé, allowing programmers to graphically create their ontology.

JADE is opaque to the underlying inference engine system, if inferences are needed for a specific application, and it allows programmers to reuse their preferred system. It has been already integrated and tested with JESS and Prolog.

To increase **scalability** or also to meet the constraints of environments with limited resources, JADE provides the opportunity of executing multiple parallel tasks within the same Java thread. Several elementary tasks, such as communication, may then be combined to form more complex tasks structured as concurrent Finite States Machines.

In the J2SE and Personal Java environments, JADE supports **mobility of code and of execution state**. That is, an agent can stop running on a host, migrate on a different remote host (without the need to have the agent code already installed on that host), and restart its execution from the point it was interrupted (actually, JADE implements a form of not-so-weak mobility because the stack and the program counter cannot be saved in Java). This functionality allows, for example, distributing computational load at runtime by moving agents to less loaded machines without any impact on the application.

The platform also includes a naming service (ensuring each agent has a unique name) and a **yellow pages** service that can be distributed across multiple hosts. Federation graphs can be created in order to define structured domains of agent services.

Another very important feature consists in the availability of a rich suite of graphical tools supporting both the **debugging** and **management/monitoring** phases of application life cycle. By means of these tools, it is possible to remotely control agents, even if already deployed and running: agent conversations can be emulated, exchanged messages can be sniffed, tasks can be monitored, agent life-cycle can be controlled.

1 A Gartner technical note [8] foresees that MOM (Message-Oriented-Middleware) will be the predominant form of communication middleware for mobile applications in the business market by 2004.

The described pieces of functionality, and particularly the possibility of remotely activating (both from code and from console), even on mobile terminals, tasks, conversations and new peers, makes JADE very well suited to support the development and execution of distributed, machine-to-machine, multi-party, intelligent and proactive applications.

JADE in the mobile environment

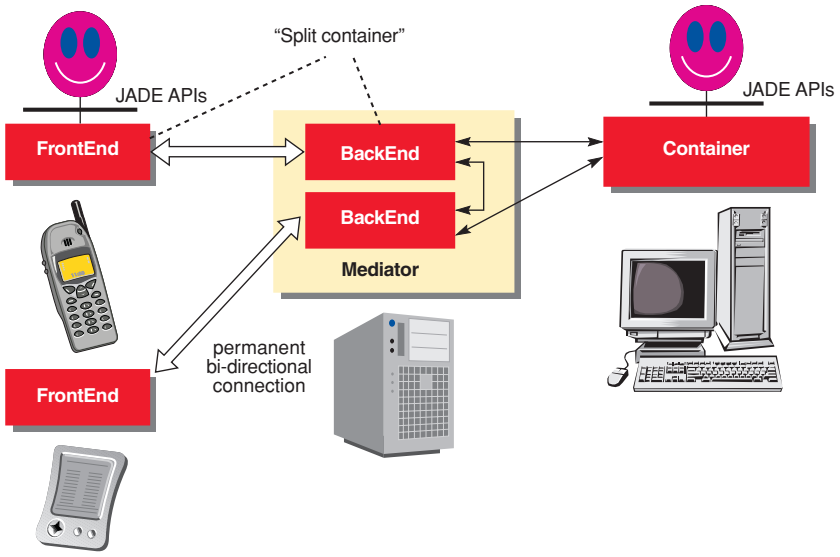
As already mentioned, the JADE run-time can be executed on a wide class of devices ranging from servers to cell phones, for the latter the only requirement being Java MIDP1.0 (or higher versions). In order to properly address the memory and processing power limitations of mobile devices and the characteristics of wireless networks (GPRS in particular) in terms of bandwidth, latency, intermittent connectivity and IP addresses variability, and at the same time in order to be efficient when executed on fixed network hosts, JADE can be configured to adapt to the characteristics of the deployment environment. JADE architecture, in facts, is completely modular and, by activating certain modules instead of others, it is possible to meet different requirements in terms of connectivity, memory and processing power.

More in details, a module called LEAP allows optimising all communication mechanisms when dealing with devices with limited resources and connected through wireless networks. By activating this module, a JADE container is "split", as depicted in figure 7 into a *front-end*, actually running on the mobile terminal, and a *back-end* running in the fixed network. A proper architectural element, called *mediator*, must be already active and is in charge of instantiating and holding the back-ends (that basically are entries in the mediator itself). To face work-load problems it is possible to deploy several mediators each one holding several back-ends. Each front-end is linked to its corresponding back-end by means of a permanent bi-directional connection. It is important to note that there is no difference at all for application developers depending on whether an agent is deployed on a normal container or on the front-end of a split container, since both the available functionality and the APIs to access them are exactly the same.

The approach has a number of advantages:

- Part of the functionality of a container is delegated to the back-end, thus making the front-end extremely lightweight in terms of required memory and processing power.
- The back-end masks, to other containers, the actual IP address assigned to the wireless device and, among the others, allows hiding to the rest of the multi-agent system a possible change of IP address.

Figure 7
JADE architecture in the wireless environment



- The front-end is able to detect a loss of connection with the back-end (for instance due to an out of coverage condition) and re-establish it as soon as possible.
- Both the front-end and the back-end implement a store-and-forward mechanism: messages that cannot be transmitted due to a temporary disconnection are buffered and delivered as soon as the connection is re-established.
- Several information that containers exchange (for instance to retrieve the container where an agent is currently running) are handled only by the back-end. This approach, together with a bit-efficient encoding of communications between the front-end and the back-end, allows optimising the usage of the wireless link.

Technical details

The following table summarizes the JADE main characteristics.

Name	JADE - Java Agent Development Framework
Provider	TILAB
Web site	http://jade.tilab.com/
Contact point	Fabio Bellifemine, email: fabio.bellifemine@tilab.com
Language	Java: J2EE, J2SE, J2ME CLDC/MIDP1.0 platforms
Availability	Open Source, LGPL license If needed, commercial licenses for specific purposes or consultancy frameworks can be properly negotiated.
Technical/functional characteristics	Distributed, multi-party application with peer-to-peer communication. Compliance with the FIPA standard. Agent life cycle management. White pages and yellow pages services with the opportunity of creating federation graphs at run-time. Graphical tools supporting the debugging, management and monitoring phases. Support for agent code and execution state migration. Support for complex interaction protocols (e.g. contract-net). Support for messages content creation and management including XML and RDF. Support for integration in JSP pages by means of a tag library. Support for application level security (currently only in J2SE). Transport protocols selectable at run-time. Currently available: JAVA-RMI, JICP (JADE proprietary protocol), HTTP and IIOP.
Network environment	Already tested in the fields over Bluetooth, GPRS, W-LAN and the Internet.
Terminals	All terminals supporting Java MIDP1.0 or Personal Java or J2SE. Already tested on Nokia 3650, Motorola Accompli008, Siemens SX45, PalmVx, Compaq iPaq, Psion5MX, HP Joranda 560.

Table 1
Summary of JADE main characteristics

THE JADE COMMUNITY

Though TILAB is the originator of the JADE project, there is an ever-growing community that participates to the whole development process of the platform. This community revolves around two major gathering points: the open source project and the government board.

The open source project

The whole JADE source code is distributed under an open source policy, the Lesser GNU Public License (LGPL for short) (6). LGPL enables full exploitation of JADE, even in a business environment, while enforcing the constraint that any modification of JADE source code and any derivative work be returned to the community under the LGPL license itself. No restrictions, instead, are put on applications and other categories of software that *uses* JADE.

A large user base, counting more than a thousand members, gathered around this project; many among them are contact points within their company or university, bridging internal JADE users with the worldwide community. Community subscribers come partly from academic environments (JADE is very popular as a teaching support environment in distributed AI courses), partly from R&D centers of world leading companies such as Motorola, HP, Siemens and Rockwell Automation, and partly from small start-ups such as Mobile Tribe and Acklin, looking at JADE as an enabling technology for their businesses. Outstanding contributions of Motorola, Siemens, and Broadcom have to be acknowledged because, within the framework of the LEAP IST project (7), they strongly contributed to port the JADE platform to the J2ME/MIDP environment.

The JADE project is supported by a web site (1) where users can download code and documentation, report possible bugs and browse a collection of useful links maintained by the JADE team. Moreover, two mailing lists are available to developers for discussing technical issues or just for staying tuned about the project, e.g. to be informed about new releases. Due to such an active user base, hundreds of JADE downloads were registered in peak days and the project counts now more than 40,000 downloads in total.

JADE welcomes contributions of the Open Source Community that can be given under different forms: simply making publicly known the usage of JADE, reporting and, better, fixing bugs and documentation, replying and giving support to less-experienced users on the mailing list, contributing with new add-ons and software modules.

The JADE Governing Board

In May 2003 TILAB and Motorola launched a new initiative, the JADE Governing Board, a not-for-profit organization, with the intent of promoting the evolution and the adoption of JADE by the mobile telecommunications industries as a java-based de-facto standard middleware for agent-based applications in the mobile personal communication sector.

The mission of the JADE Governing Board is the industrial affirmation of JADE through the establishment of consensus and the contribution of key players in the mobile sector in order to expand consumer options and interest through new wireless agent applications. The JADE-board paves the way for mobile VAS services, where peer-to-peer communication and services on PCs, PDA's and phones will enable tailored solutions for the mobile users and mobile teams to meet the increasing demand for intelligent mobile lifestyles.

The Board intends to leverage, continue and consolidate the Open-source tradition through the continuous support and involvement of the JADE Open-source Community. The Board has been formed as a contractual consortium among the Members, it is open and it welcomes all those companies and organizations that have a concrete business interest in the extension of JADE and that commit to contribute to its development and promotion. The JADE Web site provides information on how to join the Board.

WHY USING JADE?

JADE is a middleware that simplifies the development of applications. Several companies are already using it for very different application sectors including supply chain management, holonic manufacturing, rescue management, fleet management, auctions, tourism, etc. Some papers of this special issue of the EXP journal already give evidence of the types of usage, while this section tries to describe which application features best benefit from JADE.

Distributed applications composed of autonomous entities

First of all, JADE simplifies the development of distributed applications composed of autonomous entities that need to communicate and collaborate in order to achieve the working of the entire system. A software framework that hides all complexity of the distributed architecture is made available to application developers, who can focus their software development just on the logic of the application rather than on middleware issues, such as discovering and contacting the entities of the system.

This type of distributed applications enabled by JADE, in particular when applied to the mobile environment, ignite a new trend of evolution that we like to name smart-device smart-interconnection: the software on each device is equipped with autonomy, intelligence, and capability of collaboration and the value of the system is given by the capabilities of the devices and by their interaction and collaboration capabilities. This is quite different from the ubiquitous access trend where the value of the system is given by the content and the capability of accessing the content from anywhere.

Negotiation and Coordination

JADE simplifies the development of applications that require negotiation and coordination among a set of agents, where the resources and the control logics are distributed in the environment. In fact, easy-to-use software libraries to implement peer-to-peer communication and interaction protocols (i.e. patterns of interaction between autonomous entities) are provided by JADE to developers.

Pro-activity

JADE agents control their own thread of execution and, therefore, they can be easily programmed to initiate the execution of actions without human intervention just on the basis of a goal and state changes. This feature, that is usually called pro-activity, makes JADE a suitable environment for the realization of machine-to-machine (m2m) applications, for example, for industrial plant automation, traffic control and communication network management.

Multi-party applications

Peer-to-peer architectures are more efficient than client-server architectures for developing multi-party applications, as the server might become the bottleneck and the point of failure of the entire system. Because JADE agents can both provide and consume services, they remove any need to distinguish between clients and servers. JADE agents allow *clients* to communicate each-other without the intervention of a central server. Moreover, the fact that intelligence, information and control are distributed, allows the realization of applications where the ownership is distributed among the peers (agents) given that each peer may be able, and authorized to perform, just a subset of the actions of the application.

Interoperability

JADE complies with the FIPA specifications that enable end-to-end interoperability between agents of different agent platforms. All applications where inter-organization communication is needed can benefit from interoperability, including machine-to-machine and holonic manufacturing.

Openness

JADE is an open-source project that involve the contributions and collaborations of the user community. This user-driven approach allows both users and developers to contribute with suggestions and new code, which guarantees openness and usefulness of the APIs. Of course, anarchy must be avoided and the JADE Governing Board is the actor that formally controls the evolution of JADE in terms of new APIs and functionalities.

Versatility

JADE provides a homogeneous set of APIs that are independent from the underlying network and Java version. It in fact provides the same APIs both for the J2EE, J2SE and J2ME environment. This feature allows application developers to reuse the same application code both for a PC, a PDA or a Javaphone, it allows to postpone this choice as late as possible, in theory, until the deploy-time.

Easy of use and mobile applications

JADE API's are easy to learn and use. JADE has been designed to simplify the management of communication and message transport by making transparent to the developer the management of the different communication layers used to send a message from an agent to another agent, and so allowing her/him to concentrate on the logic of the application. Of course, the effect of this feature is to make faster the development of applications. JADE reduces the application development time in respect to the time necessary to develop the same application by using only Java standard packages. In particular when developing distributed applications for mobile terminals, JADE APIs and ready-to-use functionalities allow to strongly reduce the application development time and costs (some estimations have been given that indicates reduction of development time up to 30%).

GLOSSARY

ACL	Agent Communication Language
AMS	Agent Management System
API	Application Program Interface
C/S	Client-Server
CLDC	Connected, Limited Device Configuration
DF	Directory Facilitator
FIPA	Foundation for Intelligent Physical Agents
GPRS	General Packet radio System
HTTP	Hyper Text Transfer Protocol
HW	Hardware
IIOP	Internet Inter-ORB Protocol
IP	Internet Protocol
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JESS	Java Expert System Shell
JICP	JADE Internal Communication Protocol
JVM	Java Virtual Machine
LGPL	Lesser GNU Public License
M2M	Machine to Machine
MIDP	Mobile Internet Device Profile
MOM	Message Oriented Middleware
OS	Operating System
P2P	Peer to Peer
PC	Personal Computer
PDA	Personal Digital Assistant
R&D	Research and Development
RDF	Resource Description Framework
RPC	Remote Procedure Call
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SSL	Secure Socket layer
TCP	Transmission Control Protocol
URI	Uniform Resource Locator
VAS	Value Added Service
W3C	World Wide Web Consortium
W-LAN	Wireless Local Area Network
WSDL	Web Services Description Language
XML	eXtensible Markup Language

REFERENCES

[1] JADE Web Site, http://jade.tilab.com/

[2] Leonardo Chiariglione's Web site, http://www.chiariglione.org

[3] FIPA Web Site, http://www.fipa.org

[4] Michael Berger, Steffen Rusitschka, Dmitri Toropov, Michael Watzke, Marc Schlichte, Porting Agents to Small Mobile Devices –The Development of the Lightweight Extensible Agent Platform, this number of EXP

[5] BlueJADE, http://sourceforge.net/projects/bluejade

[6] LGPL license, http://www.opensource.org/licenses/lgpl-license.php

[7] LEAP Web Site, http://leap.crm-paris.com/

[8] M. Pezzini - Do MOM, ORBs and Data Access Middleware Suit Mobile? Gartner Research Note Number: T-14-3936, 20 September 2001

[9] F. Bellifemine, A. Poggi, G. Rimassa. Developing multi agent systems with a FIPA-compliant agent framework. in Software - Practice & Experience, John Wiley & Sons, Ltd. vol no. 31, 2001, pagg. 103-128

[10] Pavel Vrba, E.Cortese, F. Quarta, G. Vitaglione, Scalability and Performance of the JADE Message Transport System. Analysis of suitability for Holonic Manufacturing Systems. this number of EXP.

CONTACTS

Fabio Bellifemine
Telecom Italia Lab
Tel.: +39 011 228 6175 - Fax +39 011 228 6299
fabio.bellifemine@telecomitalia.it

Giovanni Caire
Telecom Italia Lab
Tel.: +39 011 228 6107 - Fax +39 011 228 6299
giovanni.caire@telecomitalia.it

Agostino Poggi
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma
Parco Area delle Scienze, 181A
I-43100 Parma – Italy
Tel.: +39 0521 90 5728 - Fax +39 0521 90 5723
poggi@ce.unipr.it

Giovanni Rimassa
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma
Parco Area delle Scienze, 181A
I-43100 Parma – Italy
Tel.: +39 0521 90 5728 - Fax +39 0521 90 5723
rimassa@ce.unipr.it