“The idea of an agent originated with John McCarthy in the mid-1950’s, and the term was coined by Oliver G. Selfridge a few years later, when they were both at the Massachusetts Institute of Technology. They had in view a system that, when given a goal, could carry out the details of the appropriate computer operations and could ask for and receive advice, offered in human terms, when it was stuck. An agent would be a ‘soft robot’ living and doing its business within the computer’s world” [60].
This chapter starts with the justification for the selection of agents for migration. Next sections survey the FIPA standards and features of JADE, used for agent migration. This discussion is supported by the papers[45][47].

3.1 SELECTION OF TECHNOLOGY

Generally speaking, mobility can be classified into hardware and software mobility, as described in Fig. 3.1. Hardware mobility deals with mobile computing, such as with limitations on the connectivity of mobile computers and mobile IP. Passive data represents traditional-

![Taxonomy of Mobility](image)

**Fig. 3.1:** Taxonomy of Mobility

_was of transferring data between computers_; it has been employed ever since the first two computers were connected. Active data can be further classified into mobile code, Process Migration and mobile agents[82][83]. These three classes represent incremental evolution of state transfer. Mobile code, such as Java applets, transfers only code between nodes. Process Migration discussed in chapter 2, deals primarily with code and data transfer. It also deals with the transfer of authority, for instance access to a shared file system, but in a limited way, authority is under the control of a single administrative domain. Finally, mobile agents transfer code, data, and especially authority to act on the owner’s behalf on a wide scale, such as within the entire Internet. *It is because of this advantage, it has been decided to use mobile agents to migrate a process.*

In the following section the features and applications of agents are discussed, followed by mobile agents.

3.2 AGENTS

The research areas of multi-agent systems [74][75] and distributed systems coincide, and form the research area of _distributed agent computing._
Multi-agent systems are often distributed systems [69], and distributed systems are platforms to support multi-agent systems [24]. This is depicted in Fig.3.2.

**Fig 3.2:** Relationship between the research areas of multi-agent systems and distributed systems.

Agents are considered to be autonomous (i.e., independent, not-controllable), reactive (i.e., responding to events), pro-active (i.e., initiating actions of their own volition), and social (i.e., communicative). Sometimes a stronger notion is added (beliefs, desires or intentions) realizing intention notions for agents. Agents vary in their abilities; for e.g., they can be static or mobile, or may or may not be intelligent. Each agent may have its own task and/or role. Agents, and multi-agent systems are used as a metaphor to model complex distributed processes.

Both distributed systems and agents share the notion of 'distributedness'. The area of multi-agent systems addresses distributed tasks; distributed systems address supporting distributed information and processes. The area of distributed agent computing is the area in which both approaches intersect. Both can be synergized to further optimality.

Perhaps the most general way in which the term agent is used, is to denote a hardware or (more usually) software-based computer system that enjoys the following properties:

- **Autonomy:** agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;

- **Social ability:** agents interact with other agents and (possibly) humans via some kind of agent communication language;
- **Reactivity**: agents perceive their environment (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it. This may entail that an agent spends most of its time in a kind of sleep state from which it awakes if certain changes in its environment (like the arrival of new e-mail) give rise to it;

- **Proactivity**: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative;

- **Temporal continuity**: agents are continuously running processes (either running active in the foreground or sleeping/passive in the background), not once-only computations or scripts that map a single input to a single output and then terminate;

- **Goal orientedness**: an agent is capable of handling complex, high-level tasks. The decision how such a task is best split up in smaller sub-tasks, and in which order and in which way these sub-tasks are to be best performed, should be made by the agent itself.

Thus, a simple way of conceptualising an agent is as a kind of UNIX-like software process, that exhibits the properties listed above. A clear example of an agent that meets the weak notion of an agent is the so-called *softbot* (*software robot*). This is an agent that is active in a software environment (for instance, the previously mentioned UNIX operating system).

### 3.2.1 Application of Agents

In [43][47] eight application areas are identified where now (or in the near-future) agent technology is (or will be) used.

These areas are:

1. **Systems and Network Management**

   Systems and network management is one of the earliest application areas to be enhanced using intelligent agent technology. The movement to client/server computing has intensified the complexity of systems being managed, especially in the area of LANs, and as network centric computing becomes more prevalent, this complexity further escalates. Users in this area
(primarily operators and system administrators) need greatly simplified management, in the face of rising complexity.

Agent architectures have existed in the systems and network management area for some time, but these agents are generally "fixed function" rather than intelligent agents. However, intelligent agents can be used to enhance systems management software. For example, they can help filter and take automatic actions at a higher level of abstraction, and can even be used to detect and react to patterns in system behaviour. Further, they can be used to manage large configurations dynamically.

ii. Mobile Access / Management

As computing becomes more pervasive and network centric, focus shifts from the desktop to the network, users want to be more mobile. Not only do they want to access network resources from any location, they want to access those resources despite bandwidth limitations of mobile technology such as wireless communication, and despite network volatility.

Intelligent agents which (in this case) reside in the network rather than on the users’ personal computers, can address these needs by persistently carrying out user requests despite network disturbances. In addition, agents can process data at its source and ship only compressed answers to the user, rather than overwhelming the network with large amounts of unprocessed data.

iii. Mail and Messaging

Messaging software (such a software for e-mail) has existed for some time, and is also an area where intelligent agent function is currently being used. Users today want the ability to automatically prioritise and organise their e-mail, and in the future, they would like to do even more automatically, such as addressing mail by organisational function rather than by person.

Intelligent agents can facilitate all these functions by allowing mail handling rules to be specified ahead of time, and letting intelligent agents operate on behalf of the user according to those rules. Usually it is also possible (or at least it will be) to have agents deduce these rules by observing a user's behaviour and trying to find patterns in it.

iv. Information Access and Management
Information access and management is an area of great activity, given the rise in popularity of the Internet and the explosion of data available to users.

Here, intelligent agents are helping users not only with search and filtering, but also with categorisation, prioritisation, selective dissemination, annotation, and (collaborative) sharing of information and documents.

**v. Collaboration**

Collaboration is a fast-growing area in which users work together on shared documents, using personal video-conferencing, or sharing additional resources through the network. One common denominator is shared resources; another is teamwork. Both of these are driven and supported by the move to network centric computing.

Not only do users in this area need an infrastructure that will allow robust, scalable sharing of data and computing resources, they also need other functions to help them actually build and manage collaborative teams of people, and manage their work products.

One of the most popular and most heard-of examples of such an application is the *groupware* packet called *Lotus Notes*.

**vi. Workflow and Administrative Management**

Administrative management includes both workflow management and areas such as computer/telephony integration, where processes are defined and then automated. Much as in the messaging area (application area iii in this list), intelligent agents can be used to ascertain, then automate user wishes or business processes.

**vii. Electronic Commerce**

Electronic commerce is a growing area fuelled by the popularity of the Internet. Buyers need to find sellers of products and services, they need to find product information (including technical specifications, viable configurations, etc.) that solve their problem, and they need to obtain expert advice both prior to the purchase and for service and support afterwards. Sellers need to find buyers and they need to provide expert advice about
their product or service as well as customer service and support. Both buyers and sellers need to automate handling of their "electronic financial affairs".

Intelligent agents can assist in electronic commerce in a number of ways. Agents can "go shopping" for a user, taking specifications and returning with recommendations of purchases which meet those specifications. They can act as "salespeople" for sellers by providing product or service sales advice, and they can help troubleshoot customer problems.

**viii. Adaptive User Interfaces**

Although the user interface was transformed by the advent of graphical user interfaces (GUIs), for many, computers remain difficult to learn and use. As capabilities and applications of computers improve, the user interface needs to accommodate the increase in complexity. As user populations grow and diversify, computer interfaces need to learn user habits and preferences and adapt to individuals.

Intelligent agents (called *interface agents*) can help with both these problems. Intelligent agent technology allows systems to monitor the user's actions, develop models of user abilities, and automatically help out when problems arise. When combined with speech technology, intelligent agents enable computer interfaces to become more human or more "social" when interacting with human users.

Currently available agent-systems and agent-enabled applications are of a rather basic and ad hoc nature. However, more complex and elaborated systems are in the making.

### 3.3 MOBILE AGENTS

A Mobile Agent[39], namely, is a type of software agent, with the feature of autonomy, social ability, learning, and most importantly, mobility.

More specifically, a *mobile agent* is a process that can transport its state from one environment to another, with its data intact, and be capable of performing appropriately in the new environment. Mobile agents decide when and where to move. Movement is often evolved from RPC methods. Just as a user directs an Internet browser to "visit" a website (the browser merely downloads a copy of the site or one version of it in the case of dynamic web sites), similarly, a mobile agent accomplishes a move through
data duplication. When a mobile agent decides to move, it **saves its own state**, transports this saved state to the new host, and resumes execution from the saved state.

A mobile agent is a specific form of **mobile code**. However, in contrast to the [Remote evaluation](#) and [Code on demand](#) programming paradigms, mobile agents are active in that they can *choose* to migrate between computers at any time during their execution. This makes them a powerful tool for implementing **distributed applications** in a computer network.

An open multi-agent systems (MAS) is a system in which agents, that are owned by a variety of stakeholders, continuously enter and leave the system.

Mobile agents have been the focus of much speculation and hype in recent years. The appeal of mobile agents is quite alluring - mobile agents roaming the Internet could search for information, find us great deals on goods and services, and interact with other agents that also roam networks (and meet in a gathering place) or remain bound to a particular machine. Significant research and development into mobile agency has been conducted in recent years [40], and there are many mobile agent architectures available today[33]. However, mobile agency has failed to become a sweeping force of change, and now faces competition in the form of message passing and remote procedure call (RPC) technologies.

### 3.3.1 The Technology behind Process Migration

In general, the following things are required to allow agents to migrate across a network

- Common execution language
- Process persistence
- Communication mechanism between agent hosts
- Security[81] to protect agents and agent hosts

**Common execution language**

If a process is to migrate from one host to another, then both hosts must share a common execution language. In a homogenous networking environment, it's conceivable that assembly language or machine code could be sent across the network for execution. However, such a system would be extremely limited, and not very future proof.
A more likely scenario for mobile agency is a heterogeneous environment, where many different system architectures are connected. In this case, an interpreted scripting language or emulation of a system that is capable of executing machine code\[43\] solves the problem of a common execution language.

**Process persistence**

For processes to migrate to remote machines, they must be capable of saving their execution state, or spawning a new process whose execution state will be saved. This property is called persistence. Persistence involves converting the object's state (variables, stack, and possibly even the point of execution) and converting it into a data form suitable for transmission over a network. Agents should not have to be responsible for achieving this themselves, and process persistence would likely be built into the mobile agent language or architecture.

**Communication mechanism between agent hosts**

Some communication mechanism must exist to transfer agents across networks. An agent might be transferred using TCP/IP, or by using a higher level of communication such as RMI, IIOP, SMTP or even HTTP. Mobile agent architectures may even use a variety of transport mechanisms, giving greater flexibility.

An agent's executable code must be transferred, which may consume a large amount of network bandwidth, unless shared code is located at the agent host. Techniques such as shared libraries of code, or caching, may be of benefit. In addition, the persistent state of the agent must be transferred.

### 3.4 FIPA

FIPA[36] is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies.

FIPA, the standards organization for agents and multi-agent systems was officially accepted by the IEEE as its eleventh standards committee on 8 June 2005.

FIPA was originally formed as a Swiss based organization in 1996 to produce software standards specifications for heterogeneous and interacting agents and agent based systems. Since its foundations, FIPA has played a
crucial role in the development of agents standards and has promoted a number of initiatives and events that contributed to the development and uptake of agent technology. Furthermore, many of the ideas originated and developed in FIPA are now coming into sharp focus in new generations of Web/Internet technology and related specifications.

In March 2005, the FIPA Board of Directors presented this opportunity to the entire FIPA membership, who unanimously voted to join the IEEE computer Society. Now, it is time to move standards for agents and agent-based systems into the wider context of software development. In short, agent technology needs to work and integrate with non-agent technologies. To this end, the IEEE Computer Society has formally accepted FIPA to become part of its family of standards committees.

FIPA specifications represent a collection of standards which are intended to promote the interoperation of heterogeneous agents and the services that they can represent.

The life cycle of specifications details what stages a specification can attain while it is part of the FIPA standards process. Each specification is assigned a specification identifier as it enters the FIPA specification life cycle. The specifications themselves can be found in the Repository.

The Foundation of Intelligent Physical Agents (FIPA) is now the eleventh Standards Committee of the IEEE Computer Society.

As discussed in section 3.1, we have decided to use agents for Process Migration. After examining various technologies like Aglets, Agentscape and IBM [21]Agent Systems[22] we have zeroed on Jade as it supports all the features that are required for the implementation of Process Migration(more elaborately discussed in section 2.1.1.4). In this section we discuss this technology.

3.5 JADE - AN OVERVIEW

JADE[31][51] 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[44].

3.5.1 The Java Technology
An overview of the Java technology is out of the scope of this thesis 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 thesis, it can be seamlessly executed on every type of Java Virtual Machine with exception of the Java Card.

3.5.2 What is JADE?

JADE 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 on the following driving principles:

- **Interoperability** - JADE is compliant with the FIPA specifications[36]. 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, and also do not add any computational overhead.

### 3.5.3 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 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 Fig.3.3, 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[71], i.e. compiling JADE together with the JVM[54]. 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[11] where JADE becomes service to execute multi-party proactive applications. The limited memory footprint allows installing JADE on all mobile phones provided that they are Java-enabled[76].
3.5.4 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-coupled communications, 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 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.

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.

3.5.5 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 fact, 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-end of a split container, since both the available functionality dealing with devices with limited resources and connected through wireless networks. By activating this module, a JADE container is "split" 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 the APIs to access them are exactly the same.

3.5.6 Technical Details

The following table summarizes the JADE main characteristics.

<table>
<thead>
<tr>
<th>Name</th>
<th>JADE - Java Agent Development Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider</td>
<td>TILAB</td>
</tr>
<tr>
<td>Web site</td>
<td><a href="http://jade.tilab.com/">http://jade.tilab.com/</a></td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Contact point</td>
<td>Fabio Bellifemine,</td>
</tr>
<tr>
<td></td>
<td>email: <a href="mailto:fabio.bellifemine@tilab.com">fabio.bellifemine@tilab.com</a></td>
</tr>
<tr>
<td>Language</td>
<td>Java: J2EE, J2SE, J2ME CLDC/MIDP1.0 platforms</td>
</tr>
<tr>
<td>Availability</td>
<td>Open Source, LGPL license If needed, commercial licenses for specific purposes or consultancy frameworks can be properly negotiated.</td>
</tr>
<tr>
<td>Technical/functional</td>
<td>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.</td>
</tr>
</tbody>
</table>
| characteristics              | 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, WLAN 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 3.1:** Summary of JADE main characteristics

### 3.5.7 The JADE Community

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

### 3.5.8 The Open Source Project

The whole JADE source code is distributed under an open source policy, the Lesser GNU Public License (LGPL for short)[68]. LGPL enables full exploitation of JADE, even in a business environment, while enforcing the constraint that any modification of JADE source code and any derivativte 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[66], they strongly contributed to port the JADE platform to the J2ME/MIDP environment.

**3.5.9 Why use 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.

**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.

**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 Java-phone, it allows to postpone this choice as late as possible, in theory, until the deploy-time.

**Easy of use and mobile applications:**

JADE APIs 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%).

### 3.6 SUMMARY

This chapter starts with the justification for using agents for migration, followed by the detailed survey of the topics: agents, mobile agents, FIFA and JADE. To understand the design and implementation of DDS(discussed in chapter 4 and 5), these basics are required.