CHAPTER 4

JADE PLATEFORM

JADE (Java Agent DEvelopment Framework) is a software development framework aimed at developing multi-agent systems and applications conforming to FIPA standards for intelligent agents. It includes two main products: a FIPA-compliant agent platform and a package to develop Java agents. JADE has been fully coded in Java. JADE is written in Java language and is made of various Java packages, giving application programmers both ready-made pieces of functionality and abstract interfaces for custom, application dependent tasks. Java is the programming language of choice because of its many attractive features, particularly geared towards object-oriented programming in distributed heterogeneous environments; some of these features are Object Serialization, Reflection API and Remote Method Invocation (RMI)[44].

4.1 BASIC PACKAGES

JADE is composed of the following main packages.

- **jade.core** implements the kernel of the system. It includes the Agent class that must be extended by application programmers.
- **jade.core.behaviours** sub-package contains Behaviour class hierarchy. Behaviours implement the tasks, or intentions, of an agent. They are logical activity units that can be composed in various ways to achieve complex execution patterns and that can be concurrently executed. Application programmers define agent operations writing behaviours and agent execution paths interconnecting them.
- **jade.lang.acl** sub-package is provided to process Agent Communication Language according to FIPA standard specifications.
- **jade.content** package contains a set of classes to support user-defined ontologies and content-languages. A separate tutorial describes how to use the JADE support to message content. In particular jade.content.lang.sl contains the SL codec, both the parser and the encoder.

- **jade.domain** package contains all those Java classes that represent the Agent Management entities defined by the FIPA standard, in particular the AMS and DF agents, that provide life-cycle, white and yellow page services. The subpackage jade.domain.FIPAAgentManagement contains the FIPA-Agent-Management Ontology and all the classes representing its concepts. The subpackage jade.domain.JADEAgentManagement contains, instead, the JADE extensions for AgentManagement (e.g. for sniffing messages, controlling the life-cycle of agents, …), including the Ontology and all the classes representing its concepts. The subpackage jade.domain.introspection contains the concepts used for the domain of discourse between the JADE tools (e.g. the Sniffer and the Introspector) and the JADE kernel. The subpackage jade.domain.mobility contains all concepts used to communicate about mobility.

- **jade.gui** package contains a set of generic classes useful to create GUIs to display and edit Agent-Iden-Identifiers, Agent Descriptions, ACLMessages.

- **jade.mtp** package contains a Java interface that every Message Transport Protocol should implement in order to be readily integrated with the JADE framework, and the implementation of a set of these protocols.

- **jade.proto** is the package that contains classes to model standard interaction protocols (i.e. fipa-request, fipa-query, fipa-contract-net, fipa-subscribe and soon others defined by FIPA), as well as classes to help application programmers to create protocols of their own.

- The FIPA package contains the IDL module defined by FIPA for IIOP-based message Finally, the jade.wrapper package provides wrappers of the JADE higher-level functionalities that allows the usage of JADE as a library, where external Java applications launch JADE agents and agent containers.
JADE comes bundled with some tools that simplify platform administration and application development. Each tool is contained in a separate sub-package of jade.tools. Currently, the following tools are available:

- Remote management and control. A first instance of an RMA can be started with a command line option ("-gui"), but then more than one GUI can be activated. JADE maintains coherence among multiple RMAs by simply multicasting events to all of them. Moreover, the RMA console is able to start other JADE tools.
- The Dummy Agent is a monitoring and debugging tool, made of a graphical user interface and an underlying JADE agent. Using the GUI it is possible to compose ACL messages and send them to other agents; it is also possible to display the list of all the ACL messages sent or received, completed with timestamp information in order to allow agent conversation recording and rehearsal.
- The Sniffer is an agent that can intercept ACL messages while they are in flight, and displays them graphically using a notation similar to UML sequence diagrams. It is useful for debugging your agent societies by observing how they exchange ACL messages.
- The Introspector is an agent that allows to monitor the life cycle of an agent, its exchanged ACL messages and the behaviours in execution.
- The DF GUI is a complete graphical user interface that is used by the default Directory Facilitator (DF) of JADE and that can also be used by every other DF that the user might need. In such a way, the user might create a complex network of domains and sub-domains of yellow pages. This GUI allows in a simple and intuitive way to control the knowledge base of a DF, to federate a DF with other DF's, and to remotely control (register/deregister/modify/search) the knowledge base of the parent DF's and also the children DF's (implementing the network of domains and subdomains).
- The LogManagerAgent is an agent that allows setting at runtime logging information, such as the log level, for both JADE and application specific classes that use Java Logging.
The *SocketProxyAgent* is a simple agent, acting as a bidirectional gateway between a JADE platform and an ordinary TCP/IP connection. ACL messages, travelling over JADE proprietary transport service, are converted to simple ASCII strings and sent over a socket connection. Vice versa, ACL messages can be tunneled via this TCP/IP connection into the JADE platform. This agent is useful, e.g., to handle network firewalls or to provide platform interactions with Java applets within a web browser.

### 4.2 JADE FEATURES

The following is the list of features that JADE offers to the agent programmer:

- Distributed agent platform. The agent platform can be split among several hosts. Only one Java application, and therefore only one Java Virtual Machine, is executed on each host. Agents are implemented as Java threads and live within *Agent Containers* that provide the runtime support to the agent execution.
- Graphical user interface to manage several agents and agent containers from a remote host.
- Debugging tools to help in developing multi-agents applications based on JADE.
- Intra-platform agent mobility, including transfer of both the state and the code (when necessary) of the agent.
- Support to the execution of multiple, parallel and concurrent agent activities via the behaviour model. JADE schedules the agent behaviours in a non-preemptive fashion.
- FIPA-compliant Agent Platform, which includes the *AMS* (*Agent Management System*) and the *DF* (*Directory Facilitator*). These components are automatically activated at the agent platform start-up.
- Many FIPA-compliant DFs can be started at run time in order to implement multi-domain applications, where a domain is a logical set of agents, whose services are advertised through a common facilitator. Each DF inherits a GUI and all the standard capabilities defined by FIPA (i.e., capability of registering, deregistering, modifying and searching for agent descriptions; and capability of federating within
a network of DF's).

- Efficient transport of ACL messages inside the same agent platform [45]. In fact, messages are transferred encoded as Java objects, rather than strings, in order to avoid marshalling and unmarshalling procedures. When crossing platform boundaries, the message is automatically converted to/from the FIPA compliant syntax, encoding, and transport protocol. This conversion is transparent to the agent implementers that only need to deal with Java objects.

- Library of FIPA interaction protocols ready to be used.

- Automatic registration and deregistration of agents with the AMS. - FIPA-compliant naming service: at start-up agents obtain their GUID (Globally Unique Identifier) from the platform.

- Support for application-defined content languages and ontologies.

- InProcess Interface to allow external applications to launch autonomous agents.

These features make the agent program to develop the agent application very easily. They can design the complex application in very easily & efficient manner.

### 4.3 CREATING AGENT WITH JADE

Before knowing how to create the agent, the programmer should be aware of agent platform. First let’s study the agent platform.

#### 4.3.1 Agent platform

The standard model of an agent platform, as defined by FIPA, is represented in the following figure.
The Agent Management System (AMS) is the agent who exerts supervisory control over access to and use of the Agent Platform. Only one AMS will exist in a single platform. The AMS provides white-page and life-cycle service, maintaining a directory of agent identifiers (AID) and agent state. Each agent must register with an AMS in order to get a valid AID.

The Directory Facilitator (DF) is the agent who provides the default yellow page service in the platform. The Message Transport System, also called Agent Communication Channel (ACC), is the software component controlling all the exchange of messages within the platform, including messages to/from remote platforms. JADE fully complies with this reference architecture and when a JADE platform is launched, the AMS and DF are immediately created. Furthermore the Messaging Service (implementing the ACC component) is always activated to allow message-based communication. The agent platform can be split on several hosts. Typically (but not
necessarily) only one Java application, and therefore only one Java Virtual Machine (JVM), is executed on each host. Each JVM is a basic container of agents that provides a complete run time environment for agent execution and allows several agents to concurrently execute on the same host. The main-container is the container where the AMS and DF lives. The other containers, instead, connect to the main container and provide a complete run-time environment for execution of any set of JADE agents.

![Figure 4.2 - JADE Agent Platform distributed over several containers](image)

### 4.3.2 JADE architecture

A JADE platform is composed of agent containers that can be distributed over the network. Agents live in containers which are the Java process that provides the JADE run-time and all the services needed for hosting and executing agents. There is a special container, called the main container, which represents the bootstrap point of a platform: it is the first container to be launched and all other containers must join to a main container
by registering with it. The UML diagram in Figure schematizes the relationships between
the main architectural elements of JADE.

![Figure 4.3 Relationship between the main architectural elements](image)

The programmer identifies containers by simply using a logical name; by default
the main container is named ‘Main Container’ while the others are named ‘Container-1’,
‘Container-2’, etc. Command-line options are available to override default names. As a
bootstrap point, the main container has the following special responsibilities:

- Managing the container table (CT), which is the registry of the object references
  and transport addresses of all container nodes composing the platform;
Managing the global agent descriptor table (GADT), which is the registry of all agents present in the platform, including their current status and location;

Hosting the AMS and the DF, the two special agents that provide the agent management and white page service, and the default yellow page service of the platform, respectively.

4.4 COMPILING SOFTWARE & LAUNCHING PLATFORM

All the software relating to JADE can be downloaded from the JADE website at http://jade.tilab.com. JADE-related software is divided into two sections: the main distribution and the add-ons. The add-ons in particular include self-contained modules that implement specific extended features such as codecs for given languages. In many cases these have not been developed by the JADE team directly, but by members of the open source community who decided to return their achievements to the community itself.

The main distribution is composed of five primary archive files with the following content:

- `jadeBin.zip` contains only the pre-compiled JADE Java archive (jar) files in a ready to use state.
- `jadeDoc.zip` contains the documentation, including the Administrator and Programmer guides. This documentation is also available online from the website.
- `jadeExamples.zip` contains the source code of various examples.
- `jadeSrc.zip` contains all the sources of JADE.
- `jadeAll.zip` contains all of the four files listed above.

If the above zip files are downloaded and unzipped, the directory structure should be as shown in Figure 4.4 (only the most relevant files and directories are actually shown). Some of the important files/folders include:

- License, the open source licence that regulates all use of the software.
The file jade/doc/index.html is a good starting point for beginners containing links to a variety of thematic tutorials, the Programmer and Administrator guide, javadoc documentation of all the sources, plus several other support documents.

The jade/lib folder contains all the *.jar files which must be included in the Java CLASSPATH in order to run JADE. It includes the lib/commons-codec subdirectory where an external Base64 codec is distributed that should also be included in the Java CLASSPATH.

![JADE directory structure](image)

Figure 4.4 JADE directory structure

### 4.4.1 Compiling JADE

The main-container can now be launched with the JADE GUI using the command:

```
prompt> java jade.Boot -gui
```
This figure show first JADE disclaimers & then it initialize the JADE service. In the output it also shows the MTS address & name of container. The JADE is running on the port 7778. Finally Main-Container is ready to provide the services.

4.4.2 JADE user interface

It is known as JADE Remote Agent Management GUI. It provide following options:
> Start new agent
> Kill agent
> Suspend agent
> Resume agent
> Send Message
> Migrate agent
> Clone agent
> Load agent
> Save agent
> Freeze agent
> Thaw agent
> Start Sniffer
> Start DummyAgent
> Start LogManagerAgent
> Start IntrospectorAgent

Basic interface is given below:

Figure 4.6 GUI of the JADE RMA
Now that the main-container has been initialized, any number of other containers can be launched on the various hosts composing the platform. If, for example, the host name of the machine where the main container was launched is ‘jaglan’, the command will launch a non-main container on the current host and attach it to the main-container running on the host specified by host.