JADE: Java Agent Development Framework

Overview

Marco Fabbri
(after Stefano Mariani)
marco.fabbri28@unibo.it

Dipartimento di Informatica: Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM—Università di Bologna

DS 2013/2014
Distributed Systems – 6th of March 2014
DISCLAIMER

All the material presented in these slides is rearranged by the author from a collection of documents kindly made available by the JADE team.

Then, credits for all the stuff (text & images) goes to the JADE team, in particular to Giovanni Caire.

Credits for all the mistakes goes to the author.
OUTLINE

1. What is JADE?

2. JADE Architecture
   - JADE & FIPA
   - JADE Agents
   - FIPA ACC

3. JADE Tools
JADE “DEFINITION”

JADE (Java Agent DEvelopment Framework)\textsuperscript{a} is a Java-based framework to develop agent-based applications in compliance with the FIPA\textsuperscript{b} specifications for interoperable, intelligent, multi-agent systems.

\textsuperscript{a} \url{http://jade.tilab.com}
\textsuperscript{b} \url{http://www.fipa.org}
JADE in Brief (II)

JADE GOALS

As an agent-oriented middleware, JADE pursues the twofold goal of being:

- a full-fledged FIPA-compliant agent platform. Hence, it takes in charge all those application-independent aspects – such as agent lifecycle management, communications, distribution transparency, etc. – needed to implement a MAS

- a simple yet comprehensive agent development framework. Therefore, it provides Java developers a set of APIs to build their own customizations
JADE Main Ingredients

JAVA

Being fully implemented in Java, JADE is a notable example of a distributed, object-based, agent-oriented infrastructure, hence an interesting example about how to face a design/programming paradigm shift.

FIPA

Being compliant to FIPA standards, JADE is a complete and coherent agent platform providing all the necessary facilities to deploy MASs.
JADE Main Features

JADE offers (among many)

• a distributed agent platform, where “distributed” means that a single (logical) JADE system can be split among different networked hosts

• transparent, distributed message passing interface & service

• transparent, distributed naming service

• white pages & yellow pages discovering facilities

• intra-platform agent mobility (code & context, to some extent)

• debugging & monitoring graphical tools

• …much more (we will find out later)…
OUTLINE

1. What is JADE?

2. JADE Architecture
   • JADE & FIPA
   • JADE Agents
   • FIPA ACC

3. JADE Tools
JADE Architecture Overview

(JADE System Overview)
OUTLINE

1. What is JADE?

2. JADE Architecture
   - JADE & FIPA
   - JADE Agents
   - FIPA ACC

3. JADE Tools
According to FIPA, the agent platform can be split on several hosts given that:

- each host acts as a \textit{container} of agents, that is, provides a complete \textit{runtime environment} for JADE agents execution—lifecycle management, message passing facilities, etc.

- (at least) one of these containers is the \textit{main container} (actually, the first started), responsible to maintain a registry of all other containers in the same JADE platform—through which agents can discover each other

- hence, promotes a P2P interpretation of a MAS
AMS

For a given JADE platform, a single Agent Management System (AMS) exists, which:

• keeps track of all other agents in the same JADE platform—even those “living” in remote containers

• should be contacted by JADE agents prior to any other action (they “do not exist” until registered by the AMS)

• hence, provides the white pages service—that is, a location-transparent naming service
A singleton *Directory Facilitator* (DF) exists for each JADE platform, that:

- keeps track of all advertised services provided by all the agents in the same JADE platform
- should be contacted by JADE agents who wish to publish their capabilities
- hence, provides the default *yellow pages* service—*publish/subscribe* paradigm
For a given JADE platform, a distributed message passing system exists—which is called *Agent Communication Channel*: 

- it controls all exchange of messages within the JADE platform, be them local or remote

- it implements all the needed facilities to provide *asynchronicity* of communications

- it manages all aspects regarding *FIPA ACL* (Agent Communication Language) message format, such as serialization and deserialization
FIPA Architecture (V)
(FIPA required services)
OUTLINE

1. What is JADE?

2. JADE Architecture
   - JADE & FIPA
   - JADE Agents
   - FIPA ACC

3. JADE Tools
What Is an Agent in JADE (I)

An agent is a Java object (and thread)

Being an object-based middleware, JADE agents are first of all Java objects:

• user-defined agents must extend jade.core.Agent class, thus inheriting some ready-to-use methods

• a JADE agent is executed by a single Java thread (there is an exception, though)
What Is an Agent in JADE (II)

An agent is NOT ONLY a Java object (and thread)

Jade agents have a wide range of features enabling their autonomy—despite being still Java objects

- all Jade agents must have a *globally unique name (aid)*, which is (by default) the concatenation – by symbol ‘@’ – of their local name and of the Jade platform name

- agents *business logic* must be expressed in terms of *behaviours*

- Jade agents can communicate by exchanging *FIPA ACL messages*
According to FIPA, a JADE agent can be in one of several states during its lifetime:

- **Initiated** – the agent object has been built, but cannot do anything since it is not registered to the AMS yet—it has no aid even

- **Active** – the agent is registered to the AMS and can access all JADE features—in particular, it is executing its behaviour(s)

- **Waiting** – the agent is blocked, waiting for something to happen (and to react to)—typically, an ACL message

- **Suspended** – the agent is stopped, therefore none of its behaviours are being executed

- **Transit** – the agent has started a migration process—it will stay in this state until migration ends

- **Unknown** – the agent is “dead”—it has been deregistered to the AMS
JADE Agents Lifecycle (II)

(JADE Agent Platform Life Cycle)
Agent Behaviours (I)

Why behaviours?

• By definition, agents are *autonomous* entities, therefore they should act independently and in parallel with each other.

• The need for *efficiency* drives toward the execution of JADE agents as a single Java thread each.

! However, agents need to perform complex activities, possibly composed by multiple tasks—even concurrently.

*How to conciliate this contrasting needs?*
Agent Behaviours (II)

What are behaviours?

• A behaviour can be seen as “an activity to perform with the goal of completing a task”

• A behaviour can represent both a proactive activity – started by the agent on its own – as well as a reactive activity—performed in response to some event (timeouts, messages, etc.)

Jade implements behaviours as Java objects, which are executed concurrently (still by a single Java thread) using a non-preemptive, round-robin scheduler (internal to the agent class but hidden to the programmer)
Agent Behaviours (III)

(Jade multi-tasking, non-preemptive scheduling policy)
OUTLINE

1. What is JADE?

2. JADE Architecture
   - JADE & FIPA
   - JADE Agents
   - FIPA ACC

3. JADE Tools
The Agent Communication Channel (I)

JADE messaging runtime

According to the FIPA specification, JADE agents communicate via asynchronous message passing:

- each agent has a message queue (a sort of mailbox) where the JADE ACC delivers ACL messages sent by other agents
- whenever a new entry is added to the mailbox, the receiving agent is notified—it does not need to block nor to poll either

! *if* and when the agent actually process a message is up to the agent itself (or the programmer)—for the sake of agents autonomy
The Agent Communication Channel (II)

ACL-compliant messages

To understand each other, it is crucial that agents agree on the format and semantics of the messages they exchange.

Hence, an ACL message contains:

- **:sender** – who sends the message—automatically set
- **:receiver** – who the message targets—may be many
- **performative** – the name of the communication act the agents want to carry out—constrained by a FIPA ontology
- **:content** – the actual information conveyed by the message
- **:language** – the syntax used to encode the :content
- **:ontology** – the semantics upon which the :content relies
- ...
- **others** field (we will see them maybe)
The Agent Communication Channel (III)
(FIPA communication model abstractions)
The Agent Communication Channel (IV)

JADE communication primitives

To interact, Jade agents have a couple of ready-to-use methods:

- **send** – to send a message to a (implicitly specified) recipient agent
- **receive** – to *asynchronously* retrieve the first message in the mailbox (if any)
- **timed receive** – to perform a timed, *synchronously* receive on the mailbox—timeout causes agent to “wake up”
- **selective receive** – to retrieve a message from the mailbox which matches a given message template—message queue order is bypassed

All these methods are *distribution-transparent*, that is they choose the proper address and transport mechanism based upon sender and receiver locations.
OUTLINE

1. What is JADE?

2. JADE Architecture
   - JADE & FIPA
   - JADE Agents
   - FIPA ACC

3. JADE Tools
Jade Management Tools (I)

RMA

The Remote Monitoring Agent (RMA) allows controlling the life cycle of the agent platform and of all the registered (possibly, remote) agents.

RMA Features

RMA allows to:

• start, stop, kill agents
• send them messages
• clone and/or migrate agents
• add, remove, shutdown (remote) platforms ...much more
AgentPlatforms

"192.168.1.148:1099/JADE"

Main-Container

ams@192.168.1.148:1099/JADE
df@192.168.1.148:1099/JADE
rma@192.168.1.148:1099/JADE

<table>
<thead>
<tr>
<th>name</th>
<th>addresses</th>
<th>state</th>
<th>owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>rma@192.168.1.148:1099/JADE</td>
<td></td>
<td>active</td>
<td>NONE</td>
</tr>
</tbody>
</table>
Jade Management Tools (II)

Dummy Agent

The Dummy Agent allows a human user to interact with Jade agents by sending, inspecting, recording custom ACL messages.
Jade Management Tools (II)

Sniffer Agent

The Sniffer Agent allows a user to sniff an agent or a group of agents, which means that every message directed to/from that agent / agent group is tracked and displayed.
Introspector Agent

The Introspector Agent allows to monitor and control both the queue of sent and received messages as well as the queue of behaviours—including executing them step-by-step.