JADE: Java Agent DEvelopment Framework
Overview
Multiagent Systems LM
Sistemi Multiagente LM

Stefano Mariani
*revised by Andrea Omicini*
s.mariani@unibo.it, andrea.omicini@unibo.it

Dipartimento di Informatica: Scienza e Ingegneria (DISI)
*Alma Mater Studiorum*—Università di Bologna a Cesena

Academic Year 2012/2013
1. What is JADE?

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

3. JADE Tools
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)$^a$ is a Java-based framework to develop agent-based applications in compliance with the FIPA$^b$ specifications for interoperable, intelligent, multi-agent systems.

$a$http://jade.tilab.com/

$b$http://www.fipa.org/
**JADE Goals**

As an agent-oriented middleware, **JADE** pursues the twofold goal of providing programmers & developers with

- a full-fledged FIPA-compliant *agent platform* — hence, it takes care of 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 with a set of APIs to customise their MAS
**Java**

Being fully Java-based, **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 MAS.
JADE Main Features

JADE offers

- 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
Outline

1. What is JADE?

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

3. JADE Tools
FIPA Architecture I

According to the FIPA specification, the agent platform can be split on several hosts given that:

Containers

- each host acts as a container of agents, that is, it provides a complete runtime environment for JADE agents execution—lifecycle management, message passing facilities, etc.
- (at least) one of these containers is the 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
- this promotes a peer-to-peer interpretation of a MAS
Agent Management System (AMS)

A Jade platform contains one Agent Management System (AMS), 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 they are registered by the AMS

→ provides the white pages service—that is, a location-transparent naming service
A JADE platform contains one Directory Facilitator (DF) exists, which

- 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

→ provides the default yellow pages service—publish/subscribe paradigm
A JADE platform contains a distributed message passing system – called Agent Communication Channel (ACC) –, which

- controls every message exchange within the JADE platform, be them local or remote
- implements the facilities required for asynchronous communication
- manages all aspects concerning FIPA ACL (Agent Communication Language) message format, such as serialisation and deserialisation
FIPA Required Services

- Life cycle Management
- White page service
- Yellow page service
- Message Transport service

- Agent-Software Integration
- Ontology Service
- Human Agent Interaction
Outline

1. What is JADE?

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

3. JADE Tools
An agent is *first of all* a Java object.

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 method
- a JADE agent is executed by a single Java *thread*—with an exception, though
What is an Agent in JADE II

An agent is *not just* a Java object & thread

Despite their “objectness”, and beyond their “threadness”, JADE agents have features gearing towards *autonomy*

- all JADE agents have a *globally unique name* (aid), which is (by default) the concatenation – via symbol ‘@’ – of their *local name* and of the JADE platform name
- agents *business logic* is expressed in terms of *behaviours*
- JADE agents communicate by exchanging FIPA *ACL* messages
Agent platform life cycle

According to FIPA, a JADE agent can be in one of the following states during its lifetime:

- **Initiated**: agent object built, yet not registered to the AMS
- **Active**: agent registered to the AMS (got aid)—it can access all JADE features, such as behaviour execution
- **Waiting**: agent blocked, waiting for something to happen (and to react to)—typically, an ACL message
- **Suspended**: agent stopped—none of its behaviours executing
- **Transit**: agent started a migration process—in this state until migration ends
- **Unknown**: agent “dead”—deregistered from the AMS
FIPA Agents Lifecycle II
**Agent Behaviours I**

**Why behaviours?**
- By definition, agents are *autonomous* entities, therefore they should act independently and concurrently.
- The need for *efficiency* gears toward executing each JADE agent as a single Java thread.
- However, agents need to perform complex activities, possibly composed by multiple tasks—even concurrently.

How to get the two things together?
What are behaviours?

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

→ **JADE** is *task-oriented*

- A behavior 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)
JADE Multi-tasking, Non-preemptive Scheduling Policy

- Initializations
- Addition of initial behaviours

- Agent “life” (execution of behaviours)

- Clean-up operations

Highlighted in red the methods that programmers have to/can implement

setup()

Agent has been killed (doDelete() method called)?

NO

Get the next behaviour from the pool of active behaviours

b.action()

NO

b.done()?

YES

Remove currentBehaviour from the pool of active behaviours

takeDown()
Outline

1. What is JADE?

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

3. JADE Tools
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 processes a message is up to the agent itself (or the programmer)—for the sake of agents autonomy.
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 exchange messages, **JADE** agents have some 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*, synchronous reception 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 the above methods are distribution-transparent, that is, they select 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
Remote Monitoring Agent (RMA)

The Remote Monitoring Agent (RMA) makes it possible to control the life cycle of the agent platform and of all the registered (even remote) agents.

RMA features

RMA makes it possible to
- start, stop, kill agents
- send them messages
- clone and/or migrate agents
- add, remove, shutdown (remote) platforms
- ... and much more
JADE Management Tools II

![JADE Remote Agent Management GUI]

- **AgentPlatforms**
  - Main-Container
    - RMA@TestPlatform
    - ams@TestPlatform
    - df@TestPlatform
  - Container-1
    - test-suite@TestPlatform
    - tester@TestPlatform

- **Table**
  - | Name      | Addresses | State  | Owner |
    |-----------|-----------|--------|-------|
    | df@TestPl... | active    | none   |       |
The Dummy Agent allows a human user to interact with JADE agents by sending, inspecting, recording custom ACL messages.
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.
The Introspector Agent allows to monitor and control both the queue of sent and received messages as well as the queue of behaviours—also allowing for step-by-step execution.
JADE: Java Agent DEvelopment Framework
Overview

Multiagent Systems LM
Sistemi Multiagente LM

Stefano Mariani
revised by Andrea Omicini
s.mariani@unibo.it, andrea.omicini@unibo.it

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

Academic Year 2012/2013