Overview

- Standard compliance
- JADE
  - main features, architectural choices, sub-systems
  - how it tries to meet the requirements of agents, MAS, sw developers
- JADEpL (JADE powered by LEAP)
- JADE - what it is next
- Conclusions
**Open Standards**

- **No standards, No agents**
  - emphasis on semantics and interaction: protocols, negotiation, dynamic discovery, inter-organization, inter-platform

- **Open standard**
  - the spec is openly known
  - the evolution is controlled by an independent (not for profit) body

- **Proprietary standard**
  - the spec is normally not known until a late stage
  - the evolution is controlled by a (for profit) company or group of
    - “one of the ways that Microsoft leveraged its OS was to keep critical APIs evolving internally until fairly late in the cycle, giving its developers an automatic window of opportunity of several months to develop into a new niche before a competitor could.”

- **FIPA is an Open Standard and JADE decided to comply with FIPA specifications**
FIPA
Conceptual model of an Agent Platform
FIPA
conceptual model of agent communication
JADE is a middleware

- target users: agent programmers for MAS
- includes
  - a library of classes to create agents
  - a distributed runtime environment (i.e. an agent platform) that provides agent services
    - life-cycle, white-page, yellow-page, message transport, message encoding
  - a set of tools to support the debugging phase
- designed to support scalability
  - from debugging to deployment
  - from small scale to large scale to micro scale
- claims to comply with the FIPA specifications

- fully implemented in Java
- distributed in Open Source under LPGL
- some evolutions towards SDK just started
MAS are distributed systems

- host1.it
  - Application Agent
  - Application Agent
  - Application Agent

- host2.jp
  - Application Agent
  - Application Agent
  - Application Agent

- host3.us
  - Application Agent
  - Application Agent
  - Application Agent

JADE distributed Agent Platform

- JADE Main-container
  - JRE 1.2

- JADE Agent Container
  - JRE 1.2

- Network protocol stack
Communication is central to MAS

- Agents send/receive Java objects, that represent ACLMessages, within the scope of interaction protocols
  - JADE hides all message coding (encoding/parsing)
  - Envelope level
    - String-based, XML-based
  - Agent Communication Language level
    - String-based, XML-based, bit-efficient
  - Content Language level
    - FIPA SL + API to register user-defined content languages
    - support for Base64-encoded direct Java object serialization
  - Ontology level
    - FIPA-Agent Management; JADE Agent Management
    - API to register user-defined ontologies
  - the framework can be extended by users
    - all levels provide APIs to implement/register new codecs
    - integration with Protégé
  - JADE provides a library of common interaction protocols
    - users just need to implement the handle methods
    - users can compose agent tasks like states of FSM
controls the agent’s private queue of ACL messages

designed as a chameleon
  - the transport mechanism is selected according to the situation
    - to achieve the lowest cost for message passing
    - the overheads depend on the receiver’s location and the cache status

distributed Agent Communication Channel
  - the main container is not a bottle-neck, thanks to the distributed caches
  - Message Transport Protocols (MTP) can be activated/deactivated at run-time on any container via the GUI
    - IIOP based on the ORB implementation of Sun
    - IIOP based on the ORBacus implementation
      - allows to make persistent the object reference
      - allows a more friendly URL-format corbaloc::hostname:port/name
    - HTTP MTP provided by EPFL under LGPL
  - multiple ACL encodings have been implemented
    - String-based, XML-based (EPFL), bit-efficient (Sonera)
JADE Message Transport Service

MTP can be act/deact-ivated even at run-time, even remotely.
Agents are autonomous

- it completely controls its thread of execution
  - has a private proxy of the life-cycle manager
- decides itself when to read messages and which messages to read
  - the transport mechanism fills a private queue but it does not call the agent code (no automatic callback)
- other agents have no way to get the object reference
  - asynchronous message based communication
Agents engage multiple tasks

- different containers on the same platform
  - not necessarily 1 JVM per container

- different agents on the same container
  - run in a preemptive multi-threaded environment scheduled by the JAVA Virtual Machine

- different behaviours on the same agent
  - Java multi-thread or/and
  - JADE behaviours with cooperative scheduling
    - one thread-per-agent rather than one thread-per-task/conversation. scheduled cooperatively
    - every behaviour must release the control to allow the other behaviours to be executed
    - no stack to be saved, more effort to the programmer
    - JADE scheduler carries out a round-robin non-preemptive policy among all behaviours in the ready queue
  - Behaviours can be composed into a tree
    - every Behaviour is a Finite State Machine
      - one state per execution time slot
The JADE framework includes a library of interaction protocols and generic agent behaviours, that must be customized for the specific application needs in order to create the agent capabilities.
Sometimes Agents are Mobile

- **JADE supports intra-platform mobility and cloning**
  - A platform can be distributed across multiple hosts
    - each host is an agent container
  - Agents can migrate between containers
  - Agents can clone across containers
  - Self-initiated
    - `doMove(Location) / doClone(Location, String)`
    - `before/afterMove()` `before/afterClone()`
  - Requested to the platform (via the AMS)
    - Fipa-request interaction protocol
    - `jade.domain.MobilityOntology` defines all the concepts and actions needed to support agent mobility and cloning

- **JADE might also have provided inter-platform mobility, but …**
  - the support of a standard is a MUST or is just JADE2JADE mobility
MAS are complex to debug

- **RMA (Remote Monitoring Agent)**
  - to browse the white-page service
  - to control the agent life-cycle (e.g. remote creation, agent migration, …)
  - to activate/deactivate MTPs on containers
  - to browse white-page services of remote agent platforms

- **DF GUI**
  - to browse the yellow-page service
  - to make DF federations and browse remote DF’s

- **DummyAgent**
  - send/receive store/save ACLMessages

- **Sniffer Agent**
  - to sniff, debug, save to file, multi-agent conversations

- **Introspector Agent**
  - to debug an agent: queue of sent/received messages, queue of behaviours, …
MAS must scales up and down

- Configuration of a platform
  - from one MAS on a single host
    - single-host platform
  - to one agent on a single host
    - agent platform on a cluster of hosts
  - configuration can be changed at run-time
    - hot restarting is possible thanks to the local caches
      - agent is referred by name => no need to get new reference

- Is the main-container a bottle-neck?
  - the Agent Communication Channel is distributed
  - the main container is involved only when strictly necessary

- JADE can scale down to a CLDC-device through the LEAP libraries
  - homogeneous API, no change needed to the sources
Lightweight Extensible Agent Platform

LEAP

http://leap.crm-paris.com/

Motorola
ADAC
Broadcom
British Telecommunications
Telecom Italia Lab
University of Parma
Siemens

Project number IST-1999-10211
JADE and LEAP

Library

Util j2me

Runtime

LEAP-IMTP

JADEpL j2me

Build tool
JADE and LEAP

JADE
- Administration tools
- Library
  - Util j2se
- Runtime Environment
- RMI-IMTP

LEAP Library
- Util j2me
- LEAP-IMTP

LEAP Built tool

JADE_LEAP for J2SE
- Administration tools
- Library
  - Util j2se
- Runtime Environment
- LEAP-IMTP

JADE_LEAP for PJava
- Library
  - Util j2me
- Runtime Environment
- LEAP-IMTP

JADE_LEAP for MIDP
- Library
  - Util j2me
- Runtime Environment
- LEAP-IMTP
JADEpL - Some figures

- Memory footprint: ~100 Kbyte
- Successfully tested over
  - Motorola Accompli008 (CLDC MIDP, GSM/GPRS)
  - Siemens SX45 Emulator (CLDC MIDP)
  - PalmVx (CLDC, Serial)
  - Compaq iPaq (PersonalJava, GSM/WirelessLAN/Serial)
  - Psion5MX (PersonalJava, GSM/Serial)
- Provision for disconnected devices
- Already tested over GPRS network
JADEpL – Ubiquitous deployment
Machines talking to machines

Distributed agent application

HOMOGENEOUS LAYER

Should not be the scope of FIPA?

Container
Main Container
Container
Light Container

JADEpL – JADE powered by LEAP

J2SE
J2SE
PersonalJava
CLDC

Internet

Wireless environment
The JADE license

open source does not just mean access to the source code (O’Reilly)

- JADE is OSS since Feb. 2000 (version 1.3) under LGPL
  - keep all contributors to the same level relative to each other
  - **Assure right to**
    - make and distribute copies of JADE
    - have access to the software’s source code
    - make improvements to the program
    - incorporate JADE into a proprietary program
    - continue the JADE experience even if we stopped it!!
      - *which will not happen, because we will not stop JADE so easily*
  - **Mandate duty to**
    - not keep modifications private
    - not change the license of JADE and its modifications
How much of FIPA is hidden by JADE to the programmer?

- no need to implement the Agent Platform
  - AMS, DF, and ACC automatically launched at start-up

- no need to implement agent-management ontology and functionalities
  - an agent is registered with the AP by the Java constructor itself
    - it is given a name and an address
  - the Agent class provides a simplified interface to access the services of the DF (registration, searching, …)

- no need to implement Message Transport and Parsing
  - automatically (and possibly efficiently) done by the framework when sending/receiving messages

- no need to implement Interaction Protocols
  - they must only be extended via handle methods
JADE – what is next

- JADE 2.5 released on 5th Feb.
  - Protégé plug-in for creating JADE ontologies (Chris Van Aart)
  - support for full SL, corbaloc and corbaname, ...

- in-progress activities
  - security and multi-users support
    - integrate a user-centric model with the code-centric security model of Java
      - authority and certificates to authenticate users and agents
      - permissions, policies, and delegations of permissions
      - secure channels
  - improving integration with Web technologies
  - improving support for persistence, restartability, and fault-tolerance
  - adding plans, i.e. JADE behaviours as production rule systems

- SDK
  - BlueJADE and Smart Agent - HP Labs
  - Protégé bean generator – Chris Van Aart
  - integration with Jess and Protégé – Oliver Hoffman
Conclusions

- JADE simplifies the development of MAS  
  - pay-as-you-go philosophy
  - JADEpL (JADE Powered By LEAP)

- LGPL license
  - respect and protect both the users and the authors
  - encourages to act as a community, possibly not an anarchy

- OPEN Standards and Software tools, Agents?
  - FIPA still misses some specs: security, mobility, internal APIs, …
    - is FIPA still for platform developers?
    - Are agent developers well represented in FIPA?
  - standard and sw tools are enabling components but
    - the technology needs a scope (agents to do what?)
    - identification of a killer application of mainstream technologies
    - methodology
      - pragmatic view of the theory (see OO theory vs. OO usage)

- AgentCities: from multi-agent platforms to multi-agent applications