FIPA and FIPA-OS

Stefan Poslad
Multimedia, Intelligent Systems & Applications Group
Dept. Electronic Engineering

email: stefan.poslad@elec.qmul.ac.uk
web: http://www2.elec.qmul.ac.uk/~stefan
Tutorial Objectives

- Develop a mind-set for how (FIPA) MAS type agents operate
- Understand how to develop a simple (FIPA) agent service
- Understand how FIPA-OS can be used to develop such agent services
Outline

- The essence of FIPA
- The FIPA Specifications
- Using FIPA ~ using the FIPA-OS Implementation
The FIPA type of agent

- Wooldridge & Jennings (1995) weak notion of agents:
  - **Social ability**: agents can communicate & collaborate
  - **Autonomy**: agents can say no (can also be commanded)
  - Reactive: agents perceive the environment & respond in a timely fashion
  - Pro-active: agents are goal-directed, they can take the initiative.

- W & J Stronger (mentalistic) notion of agents
  - supported by mentalistic models of communication
  - In practice require mobility and nomadicity etc
FIPA focuses on speech act protocols, dialogues & ontologies

- Buyer
- Stock Supply
- Store Assistant
- Store Manager

- query
- inform
- subscribe
- inform
- query
- inform
- query
- failure
Agent standards: a driver for scaleable agencies

- Many incompatible, proprietary agent systems exist
  - leads to closed systems and cluster of agents that are unable to communicate with each other
  - unlikely to scale (e.g., across the Internet), kills the market

- Interoperability and Openness as driving forces
  - customers strive for simplicity and universality when accessing multiple services
  - service providers can act in unison to attain a critical mass for a sustainable customer-base
The leading Agent Standard: FIPA

Foundation for Intelligent Physical Agents

16 implementations
5 open source implementations
JCP called JAS
70 + members
Several related European projects

Agent Platform

Yellow pages
White pages

Agent Communication Channel

Mobile Agent

Agent Platform

Yellow pages
White pages

Agent Communication Channel

Internet, Wireless connection, etc.
FIPA: What’s in a Name?

- **Foundation for Intelligent Physical Agents**
- **Key focuses:**
  - *software* agents but initial vision was physical agents (robotics)
  - specifying **communication** and **interoperability** between agents
  - specifies **external behaviour** not internal behaviour - don’t specify how agents process and reason about the information they receive.
- Use in **heterogeneous** environments
- **Foundation for InteroPerable Agents**
What is standardized?
What is standardized? (2)

- **Communication**
  - Dialogues, communication primitives or speech acts, content (actions), ontologies

- **Communication roles**
  - (Set by the choice of speech act & dialogue)
  - P2p, client-server, manager-contractor

- **Communication Support Services**
  - Core: Transport (encodings), Directory, Naming
  - Other: ontology, mobility, nomadicity, etc

- **Organisation & architecture**
For interoperability, it is not enough to have a de facto standard
- Standard needs to be verifiable
- Conformance to the standard needs to be verifiable

FIPA Agent Specifications consist of:
- Formal Models (design)
  - can be verified using logic proofs
  - but can’t easily verify complexity of implementation
- Descriptive Models
  - Well-established mapping of design to implementation
  - Verify implementation at specified points
Test Suite Agents

- TestSuite
- Tester x
- Forwarder
- Hacker
- AMS
- DF
- Ping
- AMS
- DF
Example

platformAddress = http://liasun17.epfl.ch:8080/acc

# list of test identifiers
tests =
(test11;test12;test13;test14;test21;test22;test30;test31;test32;test33;test34;test35;test36;test41;test42;
test43;test44;test45;test46;test11;test12;test13;test14;test21;test22;test23;test24)

# tests details
# Test 1 (MTS)
# parameters
# T: target (exiting target)
# X: non-existing target
# F: forwarder
# P: protocol used (include in the message a wrong address)

test11 = leap.testsuite.tests.agentcities.TestMTS1(T:acl_ping)
test12 = leap.testsuite.tests.agentcities.TestMTS2(T:acl_ping;F:forwarder)
test13 = leap.testsuite.tests.agentcities.TestMTS3(X:nemo)
Outline

- The essence of FIPA
- Specifications
- Using FIPA ~ using the FIPA-OS Implementation
FIPA site view of specifications

- Applications
- Abstract Architecture
  - ACL
  - Agent Management
  - Agent Message Transport
Another view of the specifications

Communication: ACL

Core Communication support

Other Communication support

Content language

Communicative acts

Interaction protocols

Naming

Transport

Directory

Agent management

Nomadic application support

Mobility Support

Configuration management

Ontology Service

Applications

PTA, PA,

Audio-Visual Entertainment

Network Management

Abstract Arch.
Speech or Communicative Act based Agent Communication

1. (Request ... it is raining, in Montreal, today (?)...)
2. (Agree (to the request))
3. (Inform (it is raining))

(interaction) protocol or conversation
dialogue = Request
communicative act = Request, agree (or refuse or failure), inform
content = It is raining
(content or ontology) language = English
ontology = weather | general conversation
### Speech Act based Agent Communication (2)

<table>
<thead>
<tr>
<th>Action</th>
<th>Accept-proposal</th>
<th>Agree</th>
<th>Cancel</th>
<th>Cfp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Disconfirm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Failure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inform</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inform-if</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inform-ref</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Not-understood</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Propose</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query-if</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Query-ref</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Refuse</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reject-proposal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Request-when</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Request-whenever</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Subscribe</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Agent Communication using the FIPA ACL

**Envelope**

- **ACL**
  - Comm. Act
  - Protocol
- **Content**
- **Ontology**
- **Language**

**Transport encoding**

```xml
<!-- Document Type: XML DTD -->
<!ELEMENT envelope ( params+ )>
<!ELEMENT params ( to?,from?,comments?, acl-representation?,payload-length?,payload-encoding?,
date?,encrypted?,intended-receiver?,
received? )> ......
```

**Message**

```xml
<!-- Document Type: XML DTD -->
......
<!ENTITY % communicative-acts "agree|confirm|failure|inform|not-understood|refuse||request|...”
<!ELEMENT content (#PCDATA)> ..
<!ELEMENT language (#PCDATA)> ..
<!ELEMENT ontology (#PCDATA)>
<!ELEMENT protocol (#PCDATA)>
```
Communicative acts & dialogues use 2 layers of protocols

FIPA communicative acts are basic types:
- **Assertives**: inform, refuse, failure, etc.
- **Directives**: request, query-reg, etc.
- **Commissives** (promises): agree
- **Expressives** (wishes?): subscribe

Messages occur in patterns called conversations or dialogues
e.g., fipa-request, fipa-cfp
Descriptive models of interaction in AUML
Content is defined using a (ontology) language & a (domain) ontology

ACL

:content
  (action
    (agent-identifier
      :name ams@foo.com
      :addresses (sequence iiop://foo.com/acc))
    (register
      (ams-agent-description
        ...
    ))

Domain Ontology defines terms, relationships between terms, operations for the content, for a particular domain

Ontology or content language defines the generic constraints, rules and operations for inferencing from the content

Content

Ontology

Language
A frame-based ontology example: a FIPA management ontology (part)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Presence</th>
<th>Type</th>
<th>Reserved Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The identifier of the agent.</td>
<td>Mandatory</td>
<td>agent-identifier</td>
<td></td>
</tr>
<tr>
<td>services</td>
<td>A list of services supported by this agent.</td>
<td>Optional</td>
<td>service-description</td>
<td>See [FIPA00025]</td>
</tr>
<tr>
<td>protocol</td>
<td>A list of interaction protocols supported by the agent.</td>
<td>Optional</td>
<td>Set of String</td>
<td>FIPA-Agent-Management</td>
</tr>
<tr>
<td>ontology</td>
<td>A list of ontologies known by the agent.</td>
<td>Optional</td>
<td>Set of String</td>
<td>FIPA-SL, FIPA-SL0, FIPA-SL1, FIPA-SL2</td>
</tr>
<tr>
<td>language</td>
<td>A list of content languages known by the agent.</td>
<td>Optional</td>
<td>Set of String</td>
<td></td>
</tr>
</tbody>
</table>
Abstract Architecture & the service model

- Focuses on core interoperability services:
  - ACL, message transport directory
  - Services don’t have to be agents but they can be

- The Abstract Architecture explicitly avoids
  - agent-platform, gateways, bootstrapping, agent configuration and coordination.
  - These elements are not included in the abstract architecture because they are implementation specific. Some elements will exist only in specific instantiations.
  - Hence in practice, FIPA is realized using FIPA implementations such as FIPA-OS to provide these features
FIPA Agent Platform

- **Agent Platform**
  - FIPA Agent Platform
  - Message Transport Service
  - Directory Facilitator
  - Agent Management System
  - Software

- **Message Transport**
  - ACL
  - API
  - FIPA00023
  - FIPA00067

- **Non-agent software**
  - CORBA
  - IIOP
  - HTTP etc.

- **FIPA Agent Platform**
  - FIPA00023 - 60

MATA'01 FIPA & FIPA-OS tutorial
Abstract architecture and Interoperability

Abstract Architecture

Messaging  ACL  Directory

Java1 Instance

Messaging  ACL

gateway

C++ Instance

Messaging  ACL

LDAP
Directory

An instance
Abstract architecture vs. Agent Platform

- FIPA Agent Platform is specified in
  - FIPA00023 agent management specification
  - FIPA00067 message transport specification
- Agent platform can be regarded as a concrete realisation of the abstract architecture [FIPA00001]
Outline

- What is FIPA?
- Specifications
- Using FIPA ~ using the FIPA-OS Implementation
  - Installing an agent platform and running agents
  - A look inside FIPA-OS
  - Developing agent services
FIPA-OS

- A ‘reference implementation’ of the core FIPA specifications for agent interoperability
  - ACL, Agent platform, etc.
- OS means Open Source, freely available and modifiable source code (c.f. Linux)
- Enables adoption of FIPA without the need to implement the core specifications
- Assist in validating and evolving FIPA standards
- Started in August 1999, 12+ formal releases to date (25,000+ downloads)

FIPA-OS is the first Open Source implementation of FIPA
The core types of agent behaviour supported by FIPA-OS

The basic agents supported are:

- **Reactive**: can react to ACL messages from other agents in the environment
- **Proactive**: they can decide when to initiate interaction with other agents
  - N.B. simple goals. E.g., register with the name service, without plans
- **Social**: (see reactive and proactive)
- **Autonomous**: each agent has multiple threads of control
- **Mentalistic features**: via use of ACL
Using FIPA-OS to install and run FIPA agents

1. Download FIPA-OS source & tutorials from source-forge (fipa-os.sf.net)
2. Install FIPA-OS in 1 of 2 ways:
   1. Executable: self extracting zip that automatically runs the configurer tool
   2. Manually unzip and run the configurer tool
      - Installation assumes enough environment space, write access and a suitable version of the JVM
3. Start the agent platform and load agents
4. Test the platform using the IOTest agent and the (Tutorial) Ping agent
Configuring FIPA-OS: using the Wizard (Simple)

- FIPA-OS Wizard aims to simplify initial configuration and start-up
- Can be used when installing FIPA-OS, or anytime the platform needs to be configured
- Wizard modifies following files
  - acc, platform and default profiles
  - SetupFIPAOS batch files
- Wizard GUI consists of multiple panels, depending upon complexity of installation
- Information about configuration options are provided within the GUI
Configuration Wizard

- Stand-alone for simple development
- Distributed platform for serious development

How should this installation of FIPA-OS be configured?

- As a stand-alone platform
  Select this option if you plan to use FIPA-OS in isolation on this computer. This is the simplest option, since the majority of configuration will be done automatically.

- As part of a distributed platform / to interoperate with other platforms
  Select this option if you wish to use this installation of FIPA-OS as part of an Agent Platform which spans several computers, and/or interacts with other FIPA platforms. In this case one of the computers that are part of the AgentPlatform must run the Naming Services for the platform, and one must run the AMS and DF agents.

  - Start platform NamingServices here
  - Select this to start the Agent Platform Naming Services on this computer

  - Start platform agents (AMS & DF) here
  - Select this to start the AMS and DF for the Agent Platform here

  - Start platform ACC here
  - Select this to start the ACC for the Agent Platform here (the ACC is only needed if your platform will interact with other Agent Platforms)
Initialising a FIPA Agent Platform (AP)

1. Start any transport specific Naming services
   - E.g., Sun CORBA and RMI
   - Start by using batch files provided
2. Start the Message Transport Service (ACC)
   - Via Agent Loader or manually via batch file
3. Start the core AP agents
   - (Use of FIPA-OS Agent Loader enables agents to be managed easily)
   - E.g., Name Service (AMS), Directory Service (DF)
4. Start any End Service and End user agents
   - (Use of FIPA-OS Agent Loader enables agents to be managed easily by users)
   - E.g., Ping service Agent
   - E.g., IOTest Agent (service user agent)
Starting Up: using the Agent Loader

- By starting the Agent Loader, the platform agents (DF, AMS) will start up
- New agents can be started by selecting them in the right list, and clicking “Start”
- Unlisted agents can be started by selecting “Start other…”

Exercise
- Install and Configure FIPA-OS using the Wizard for a ‘stand alone platform’
- Install the Tutorial agents
- Start-up the platform using the AgentLoader (StartFIPA-OS)
Testing the Platform

- Start up the IOTestAgent from the GUI and try sending the example ACL messages provided in the ‘examples’ directory.
- Platform can also be tested by using the tutorial agents (separate installation) like Ping Agents.

Exercise
- Start the IOTestAgent using the AgentLoader
- Start the PingAgent using the AgentLoader
- Use the IOTestAgent to send the example ‘ping’ message (acping.txt) to the PingAgent
- Use the IOTestAgent to register with the AMS and DF
Outline

- Using FIPA ~ using the FIPA-OS Implementation
  - Installing an agent platform and running agents
  - A look inside FIPA-OS
  - Developing agent services
High Level Architecture of FIPA-OS Agent Shell

KEY
- Mandatory Component
- Switchable Implementation
- Optional Component

Agent Implementation
- JESS Agent Shell
- Agent Shell
- Task Manager
- Conversation Manager
- Message Transport Service
- Message Transport Protocols

Database
- Database's
- Parser's
- Choice Constraint Language
FIPA-OS: Conversation Manager

Agent Implementation

Conversation Manager

Active Conversations
- Conversation 23
- Conversation 24
- Conversation 29

Ended Conversations
- Conversation 1
- Conversation 2
- Conversation 22

Dispatcher / Scheduler

Comms
FIPA-OS: Task Manager

- Separates agent ‘tasks’ into distinct objects
- Messages are automatically routed to the correct state
- Inter-task events are possible
Outline

- ....
- Using FIPA ~ using the FIPA-OS Implementation
  - Installing an agent platform and running agents
  - A look inside FIPA-OS
  - Developing agent services
Developing agents & services

FIPA-OS

Agent Implementation
- JESS Agent Shell
- Agent Shell
- Task Manager
- Conversation Manager
- Message Transport Service
- Message Transport Protocols

Functions

- Specify the agent architecture, agent organisation & roles
- Specify service Interface & Implementation
- Interlink agent platforms and services
- Support mandatory services: Naming, directory, transport
- Set transport encoding, content languages
- Handle, parse & transmit ACL messages
Specifying the agent architecture, organisation & roles

Determined by

- Conversation patterns used
- The middle agent hierarchy depth
- Platform & Service interlinking

- E.g., SearchAgent
  - service discovery: uses a 3 tier client server arch. & the fipa-request conversation pattern
  - service usage: uses a 2 tier client server arch. & fipa-request conversation pattern
Interlinking or Federating Agent Platforms
Developing agents & services

- Define service description to advertise service (in DF)
  - Use the standard FIPA agent management ontology
- Define run-time service interface
  - Define a domain-specific ontology
Using Ontologies


Agent internal representation of Domain Ontology, e.g., .....Java objects

Ontology language

Communication representation of Domain Ontology: XML/RDF

translate

translate
Specifying a FIPA agent service

**Service Interface = ACL**

1. (query-ref
   protocol: FIPA-query
   language: string
   ontology: none
   content: (ping) ..)
2. (agree ....)
3. (inform content: (pong) ..)

**Agent Service**
e.g., Ping

**DF service description**

name: ping
description: ...
Protocol: FIPA-query
Ontology: none
........

**DF agent**

**Service provider agent**

**Service user agent**
Message handling (SearchAgent): task & conversation design
Tutorial Summary

- Develop a mind-set for how (FIPA) MAS type agents operate
- Understand how to develop a simple (FIPA) agent service
- Understand how FIPA-OS can be used to develop agent services
Thank you!

Some useful URLs:
- [http://www.fipa.org](http://www.fipa.org)
- [http://fipa-os.sf.net](http://fipa-os.sf.net)

Some FIPA agent projects
- [http://www2.elec.qmul.ac.uk/~stefan](http://www2.elec.qmul.ac.uk/~stefan)
- At FIPA web-site

Acknowledgement: thanks to Emorphia Ltd for the use of some slides for this presentation
Reserve slides
The FIPA specification life-cycle: specify -> experiment -> standard

- Preliminary: following approval of TC
- Experimental: expires after 2 years, promotes approval of FAB
- Standard: demotes approval of FAB + vote of FIPA members
- Obsolete: expires after 6 months, promotes approval of FAB + vote of FIPA members
- Standard: deprecated with a retire date
- Standard: retires after retire date

Promotes approval of FAB
Expires after 6 months
Promotes approval of FAB +
vote of FIPA members
Deprecates with a retire date
An agent consists of objects but it is more than a set of objects

- An agent has a strong notion of autonomy
- Agents are active, they have their own threads of control
- Async. comms. (MP)
- FIPA agents support a universal lingua franca
- FIPA agents support a richer semantic, varied communication for cooperation

- An object can be controlled externally
- Objects are passive
- Synch. comms. (MI)
- Objects use proprietary interfaces
- Objects support syntactic, synchronous communication
Content languages vs. ontologies

Content language
Ontology language?
- Representation for handling input, generating new output & processing information
- Domain independent
- E.g., SL(0-2), CCL, OIL?
- Defined in the content language specifications

Ontology
domain instance ontology
- Representation for Defining Storing, retrieving & indexing domain information
- Domain dependent
- E.g., fipa-mgt-ontology
- These are defined in the management specs
FIPA Test Suites

- 1\textsuperscript{st} one specified by Motorola and EPFL, Implemented by the LEAP project (specifications available at http://www.agentcities.org/Testsuite)
- To be used as a conformance test suite by the Agentcities project
- Tests the Connection and Communication layers for FIPA platforms
Test suite (2): testing FIPA AP Connection and Communication

- Agent Message Transport Service
  - Send message to one/multiple/non-existing agents...

- Conversation management
  - conversation-id, reply-with/in-reply-to

- Agent Management Service
  - ap-description
  - dynamic registration (register, change registration, search, deregister)
  - security

- Directory Facilitator
  - register, change registration, search, deregister
  - security
  - federation
Test Suite (3): Design

HTML, Screen output, logs
Configuring FIPA-OS Using the Configurator (Advanced)

- Can be used when installing FIPA-OS, or anytime the platform needs to be configured
- Configurator modifies following files
  - acc, platform and default profiles
  - SetupFIPAOS batch files
- Configurator GUI consists of five panels
ACC Profile Configuration

Details of the external MTP’s that the ACC should bind into upon start up

Details about a remote agent platforms

Details of the platform MTP’s that the ACC should bind into upon start up

Filename to which the ACC publishes its MTP’s addresses

Type of database used by the ACC and the location
Configuration Wizard

Stand-alone for simple development

Distributed platform for serious development
Platform Profile Configuration

The HAP name used by agents on the platform - this should be globally unique, like an IP address or domain name.

This is the internal MTP address via which the AMS can be contacted.

This specifies where the profiles for entities belonging to this platform can be located.
FIPA-OS Script Configuration

Indicating the **version** of FIPA-OS

Location of **JVM**

Choice of whether to show **debugging** information or not

If debugging is used, what level messages are **shown** (5 = MAX)

The **directory** where FIPA-OS is installed

Location of the **platform.profile**

Allow the disabling of any JIT or HotSpot performance **compiler**

Choice of whether to use **Agent Loader** GUI or not

If debugging is used, what level messages are written to **file**
Default Profile Configuration

The type of **database** used by agents using the default profile and it’s location

Details of the **platform MTP’s** that Agents should bind into upon start up

Details of the protocols known by Agents

MATA'01 FIPA & FIPA-OS tutorial
Agent Loader Configuration

Details of Agents known by Agent-Loader