Introduction to MAS

(Multi-Agent Systems)

P. Laroque

march 2018

Outline I

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

Outline

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

Everyday Language

Dictionary definitions:

1. one that acts or exerts power
2. something that produces or is capable of producing an effect : an active or efficient cause
3. a chemically, physically, or biologically active principle
4. one who is authorized to act for or in the place of another as
    1. a representative, emissary, or official of a government
    2. one engaged in undercover activities (as espionage)
    3. a business representative (as of an athlete or entertainer)

Agent Main Properties

- Autonomous
- "Intelligent", rational, learning...
- Mobile
- Communicating
- Acting (on other agents and on the environment)

Outline

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

More Formal Definition I

(J. Ferber [9, 10]) An agent is a *physical* or *virtual* entity

1. which is capable of *acting* in an environment.
2. which can *communicate* directly with other agents.
3. which is driven by a set of *tendencies* (in the form of *individual objectives* or of a *satisfaction/survival function* which it tries to optimize).
4. which possesses *resources* of its own.
5. which is capable of *perceiving its environment* (but to a limited extent).
6. which has only a *partial representation* of its environment (and perhaps none at all).
7. which possesses *skills* and can offer *services*.
8. which may be able to *reproduce* itself.

More Formal Definition II

9. whose behaviour tends towards satisfying its objectives, taking account of the resources and skills available to it and depending on its perception, its representation and the communications it receives.

Autonomy is central

- Agent are partially independent and can make decisions.
- "Tendency" = individual goal / function optimization

Outline

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

Definition of a MAS

1. An environment E, that is, a space which generally has volume.
2. A set of objects, O. These objects are situated, that is to say, it is possible at a given moment to associate any object with a position in E.
3. An assembly of agents, A, which are specific objects (a subset of O), represent the active entities in the system.
4. An assembly of relations, R, which link objects (and therefore, agents) to one another.
5. An assembly of operations, Op, making it possible for the agents of A to perceive, produce, transform, and manipulate objects in O.
6. Operators with the task of representing the application of these operations and the reaction of the world to this attempt at modification (the "game's rules").

Purely Situated Agents

Such an agent:

1. is situated in an environment,
2. is driven by a survival/satisfaction function,
3. possesses resources of its own in terms of power and tools,
4. is capable of perceiving its environment (but to a limited extent),
5. has practically no representation of its environment,
6. possesses skills,
7. can perhaps reproduce,
8. has behaviour tending to fulfill its survivor/satisfaction function, taking into account the resources, perceptions and skills available to it.

Example

For instance, robots:

- E = the physical space
- A = the robots
- O = other robots + objects (obstacles,...)

Purely Communicating Agents

Such an agent:

1. is in an open computing system (assembly of applications, networks, and heterogeneous systems),
2. can communicate with other agents,
3. is driven by a set of its own objectives,
4. possesses resources of its own,
5. has only a partial representation of other agents,
6. possesses skills (services) which it can offer to other agents,
7. has behaviour tending towards attaining its objectives, taking into account the resources and skills available to it and depending on its representations and the communications it receives.

Example

For instance, most software agents:

- A = O and E = empty
- agents are in a communication network and exchange messages

no perception of other agents

In this course, we focus on this kind of agents

Outline

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

Rationality

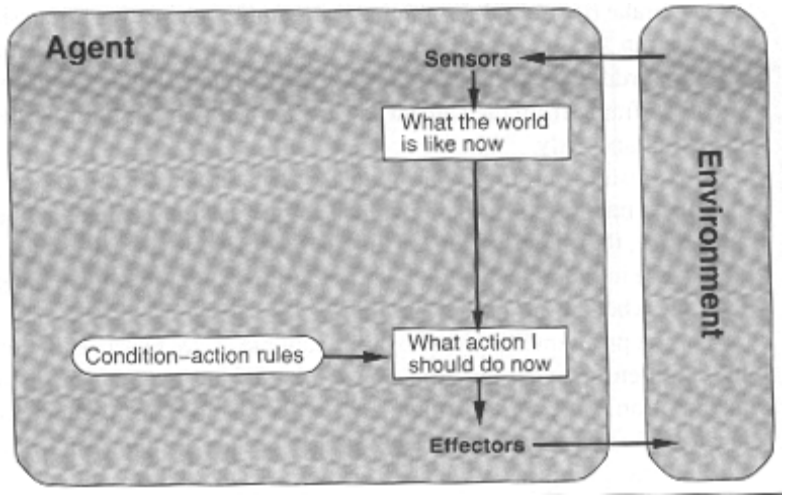
- "Right" decision taken by examining the environment
- At least, maximizes chances of success
- Measure: we need
  1. A function to evaluate success
  2. Access to the agent history
  3. What the agent knows about the environment
  4. The available actions

Perceptions and Actions

- Major role of the agent's "brain": map perceptions to actions
- Ideally, *each* perception is mapped to a given action
- In the real world, mapping is done thanks to
  - analytical functions (quality)
  - production rules
  - neural nets
  - fuzzy sets
  - ...

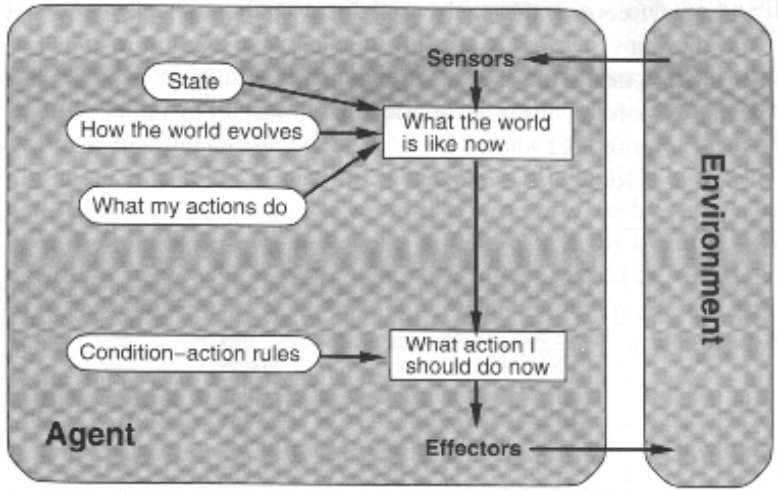
Reflex Agents

- "Stimulus-response" behaviour (Nilsson [7])
- No memory
- In a MAS, can achieve rather complex tasks



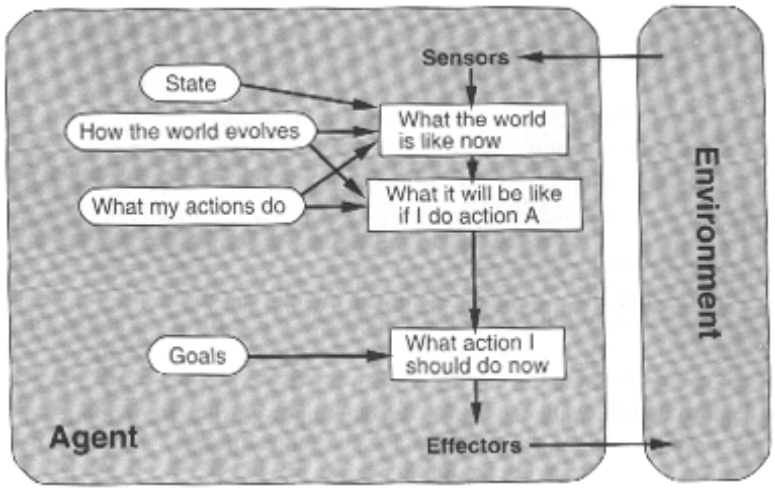
Stateful Reflex Agents

- State memory to "remember" past experiences.
- More sophisticated response to the environment



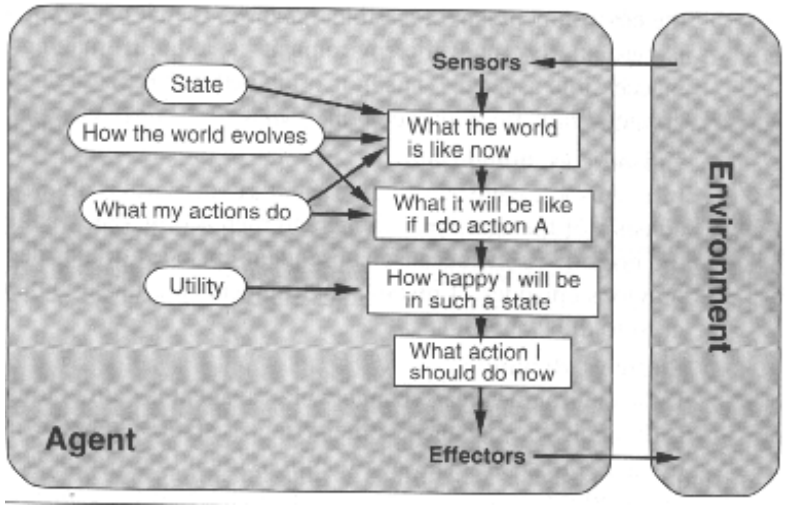
Goal-based Agents

- Can plan before actual move
- Planning is frequently based on search in state space (chess) and evaluation function



Utility-based Agents

- Utility: microeconomics term related to happiness
- Beyond present AI capabilities



Minimal Structure

- Low-level support for networked communications (messages) → CORBA, RMI, .NET, ...
- Services (life-cycle, white pages, yellow pages, ...) → CORBA, RMI, .NET, ...
- Common (*standardized*) languages
  - communication language (message structure)
  - content language (message contents)
  - ontologies (symbols' semantics)

→ FIPA → JADE

Outline

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

History and Goals

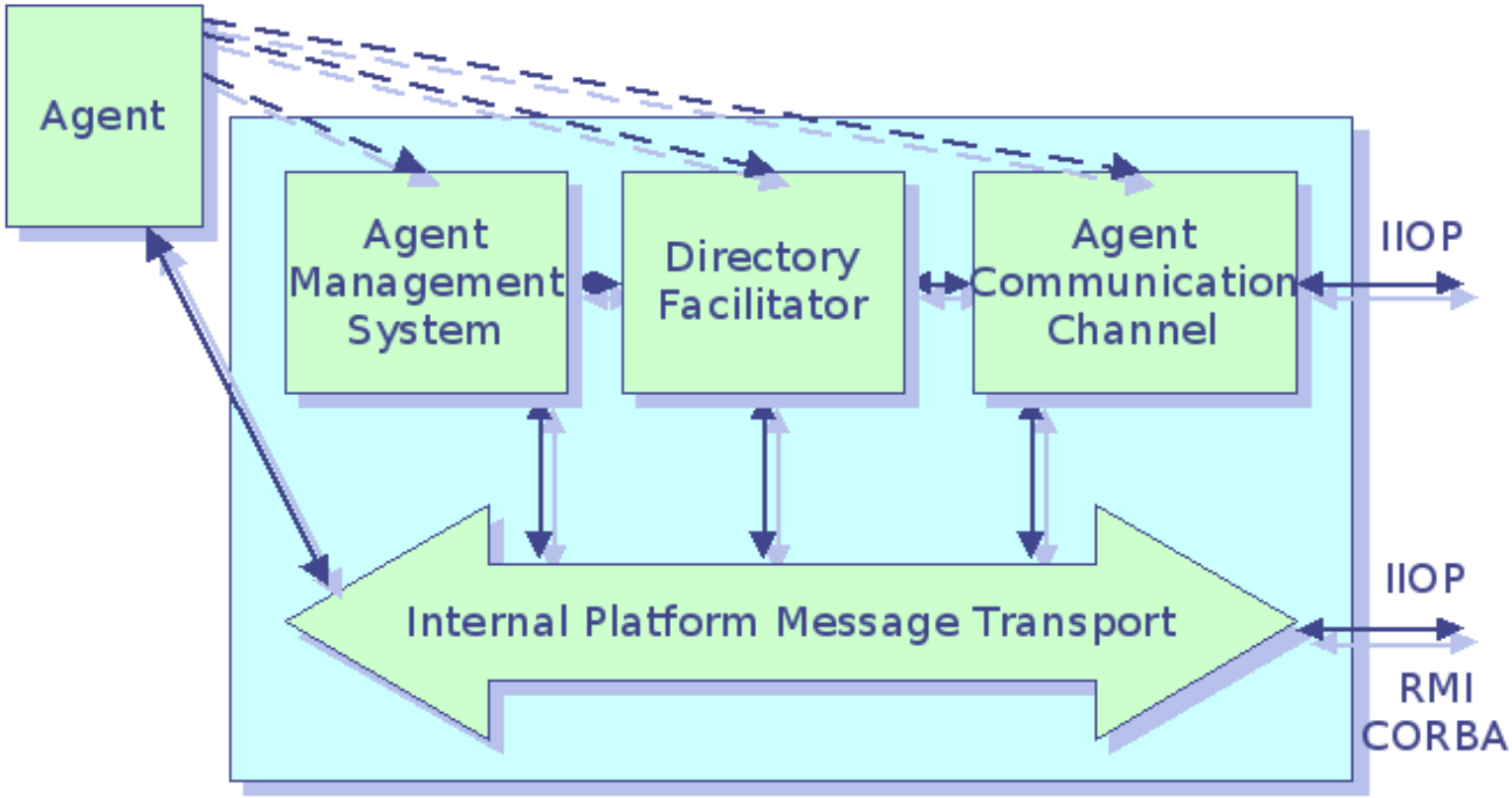
- Foundation for Intelligent Physical Agents [1]
- Created in end 1996
- Standardized in 2002
- Purpose: design standards for heterogeneous agents interacting inside MASes (*specifications*)
- Strong links to OMG and other organizations
- `www.fipa.org`

Scope of FIPA

- Agent Lifecycle Management
- Message Transport ($\rightarrow$ *asynchronous*, $\neq$ RPC)
- Message Structure
- Inter-agent Interaction Protocols
- Ontologies
- Security

NOT the agent!

The FIPA Platform



The AMS

- Agent Management System
- Responsible for agents' life-cycle
- Maintains a list of all agents living in current platform (white pages)
- Controls access to and usage of ACC (Agent Communication Channel)

The DF

- Directory Facilitator
- Register agent descriptions, together with their available *services*
- Agents can ask DF for services (yellow pages)

The ACC

- Agent Communication Channel
- Handles agent communications
  - inside a container (for instance, java events)
  - between containers inside a platform (for instance RMI)
  - between platforms (for instance IIOP CORBA)
- Messages in ACL (Agent Communication Language)

The ACL

Most important features of an ACL message:

- performative: the message type (query, proposal, reject,...)
- sender ($\neq$ message in the traditional, object-oriented meaning)
- receiver, reply-to
- content
- conversation-id (useful for dialogs)

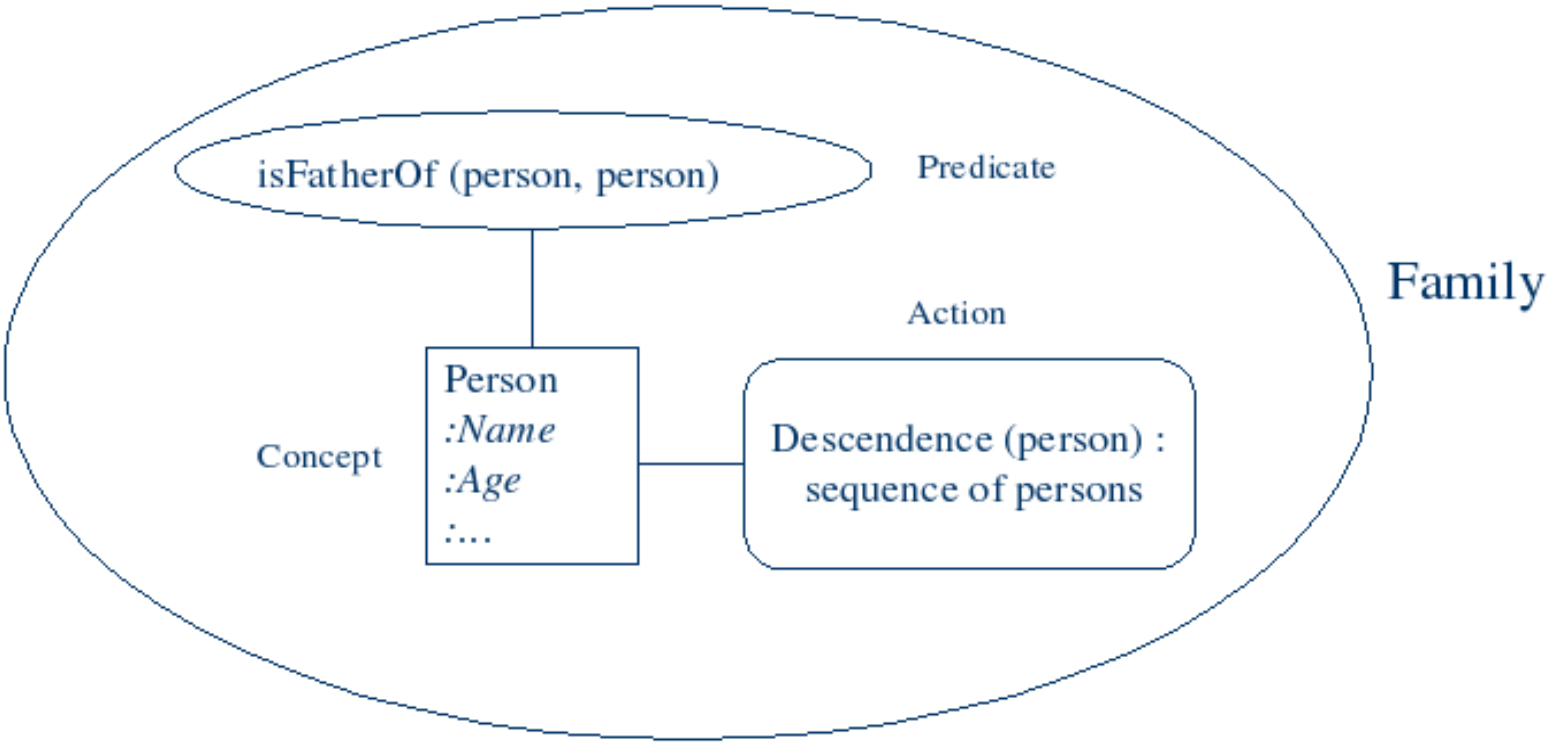
Message Content

- In simple cases, string
- otherwise we need
  - a content language (for syntax)
  - a corresponding ontology (for semantics)
  - a protocol (for communication process)
- Ex: the default language used by JADE is SL (Semantic Language).

Ontologies

- Structured representation of knowledge
  - Concepts (abstract objects representation)
  - Predicates (binary conditions on concepts)
  - Actions (operations proposed by agents on concepts)
- Goal: share knowledge by narrowing the universe of discourse

A Simple Example



Ongoing Work

- Agent Modelling
- Agent Methodology [8]
- Semantic Framework
- Service Framework
- Security

Outline

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

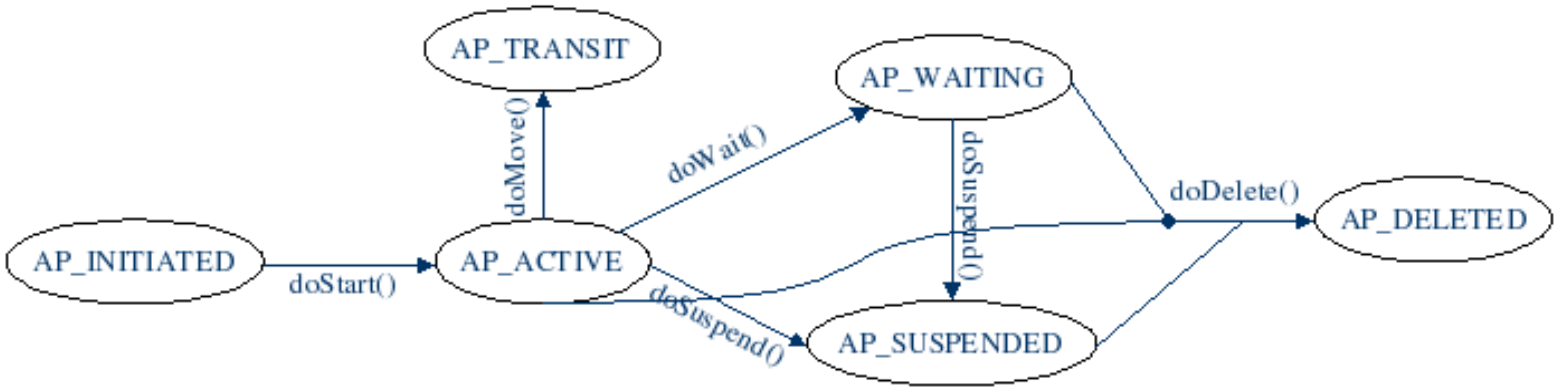
What is JADE?

- JADE (Java Agent DEvelopment framework [2])
- JAVA implementation of FIPA 2002
- Several tools (dummy agent, sniffer, introspector...)
- Based on J2se 1.4 and above
- Free software (LGPL)
- LEAP library to use JADE in mobile Java environments down to J2ME-CLDC MIDP 1.0

JADE Agents

- Compliant with FIPA 2002
- Can have several *behaviours* that define its actions
- Communicate with other agents using asynchronous *messages*
- Publish *services*

Agent Life Cycle



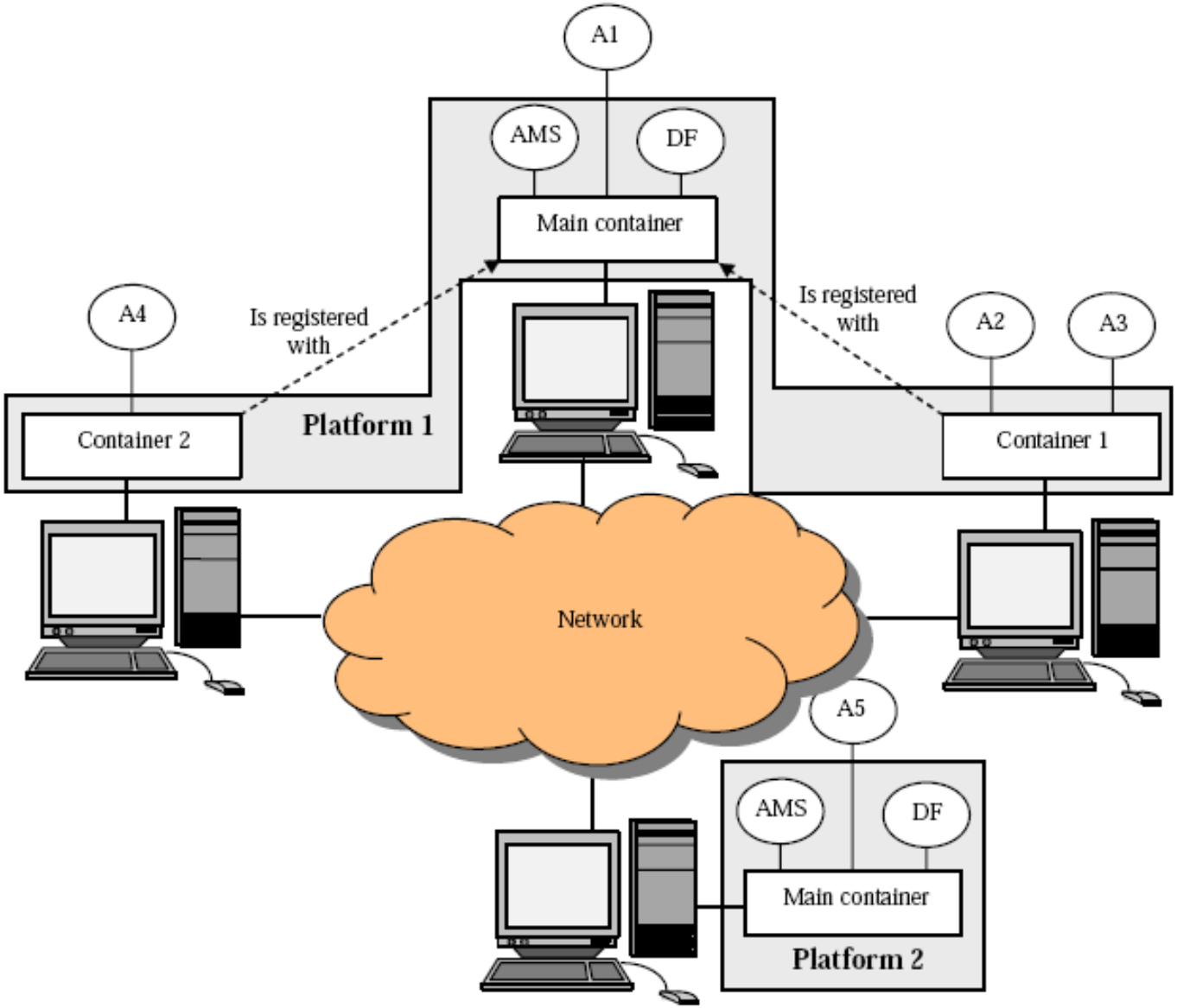
Outline

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

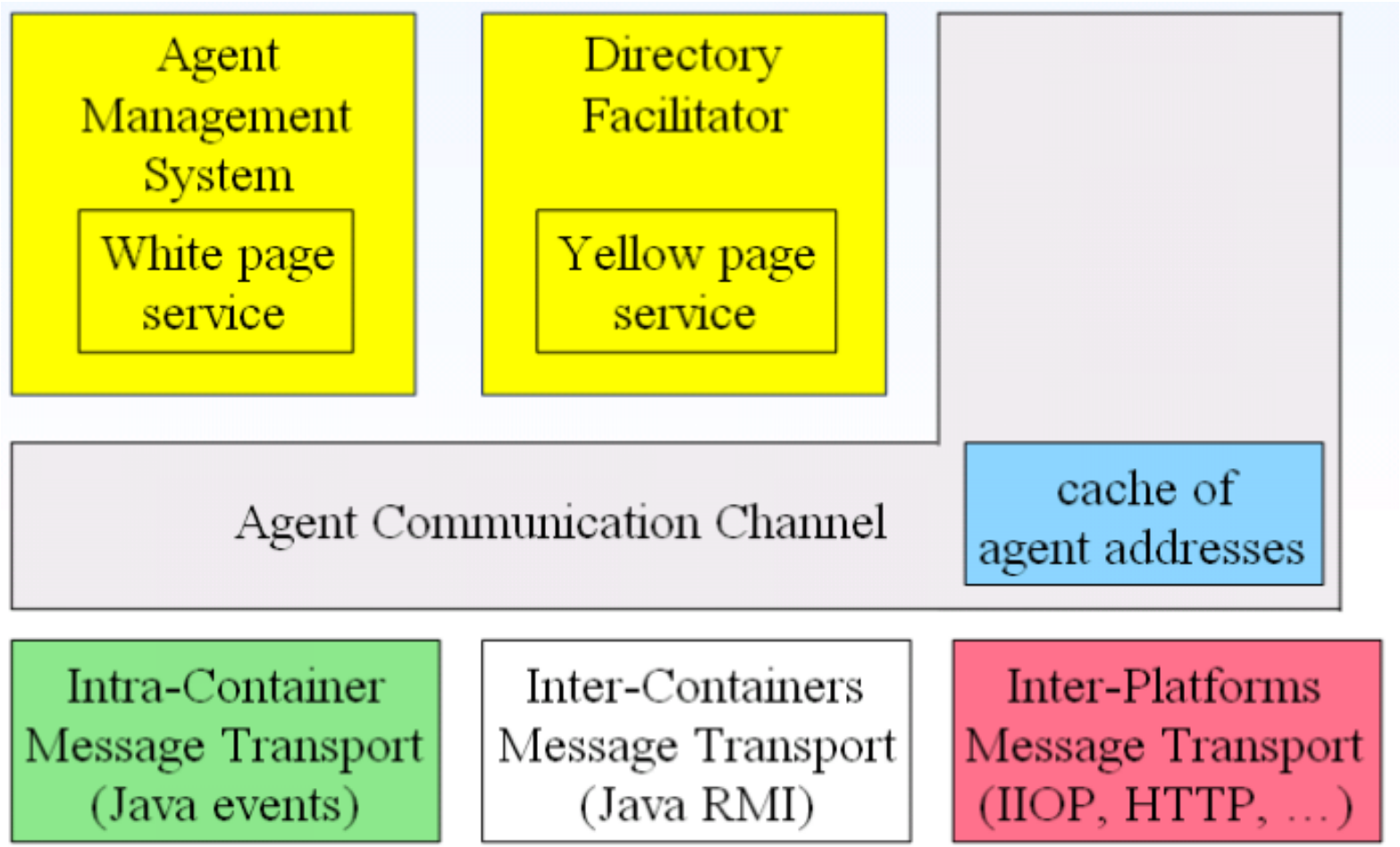
JADE Components

- Agents: use JADE to offer/use services and communicate with each others
- Containers: an agent must live in *exactly* one container; it can move to another container (*mobility*)
- Platform: set of containers whose agents can communicate with each others. Exactly *one* container is the *main container*
- A main container embed at least 3 agents:
  1. The AMS agent (Agent Messaging Service)
  2. The DF agent (Directory Facility)
  3. The RMA (Remote Monitoring Agent: platform and agents' life cycle)

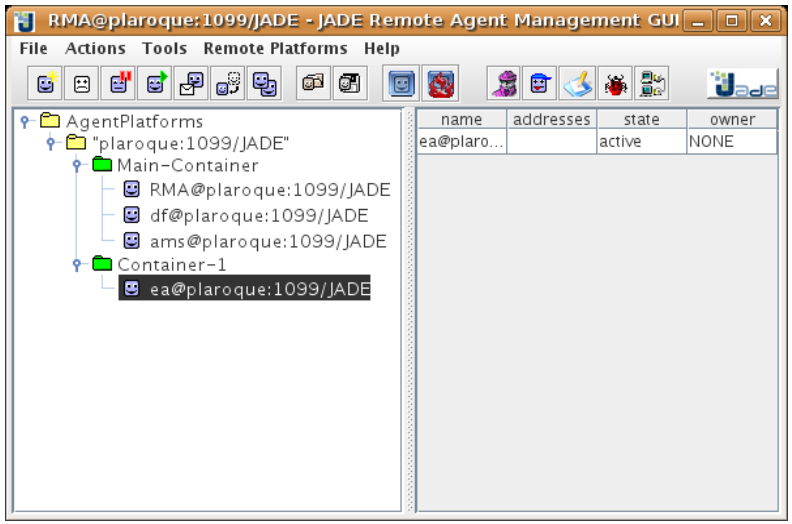
JADE Network Structure



Typical JADE Communications

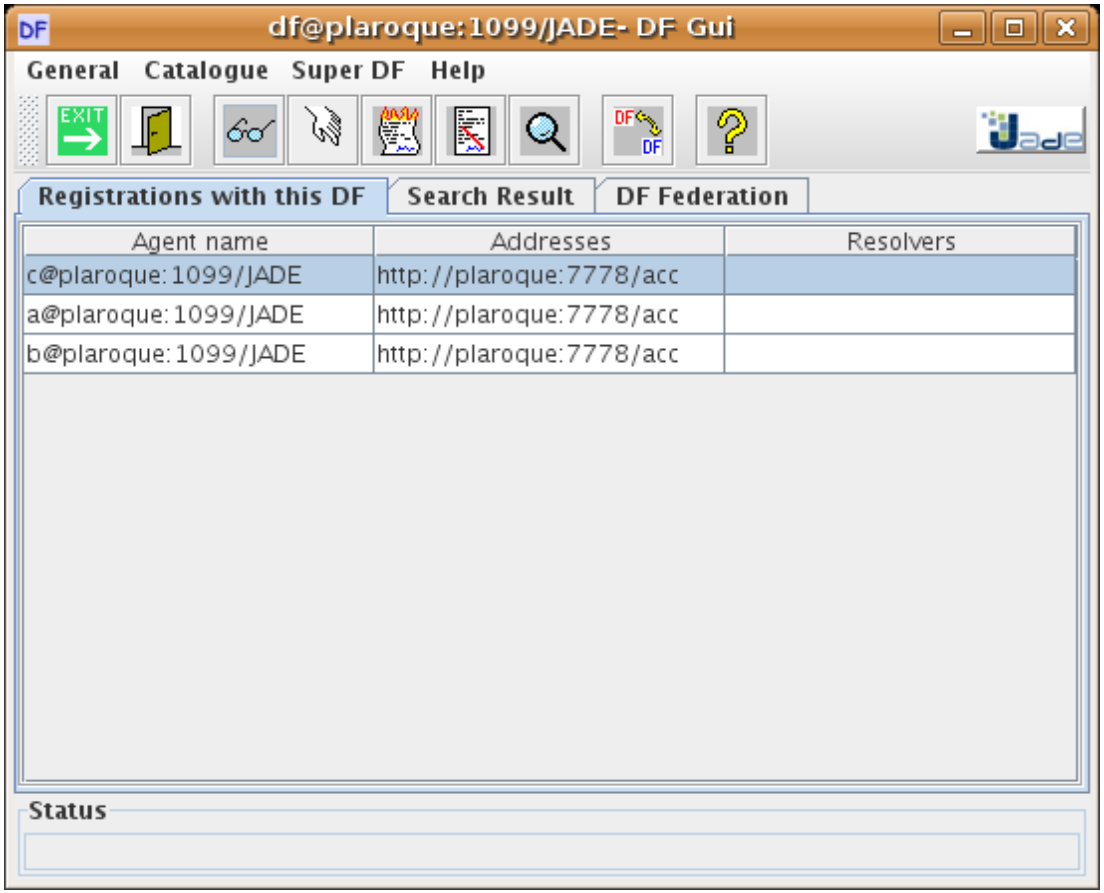


Jade GUI



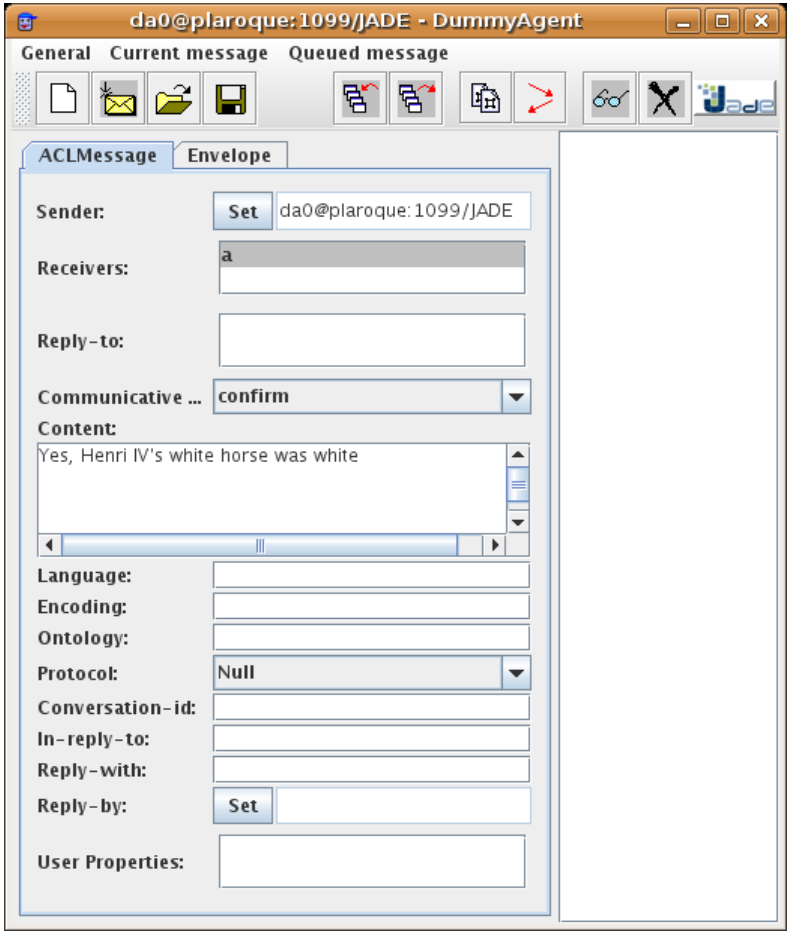
- Control agents (create, kill, suspend...)
- Start other tools

Jade DF



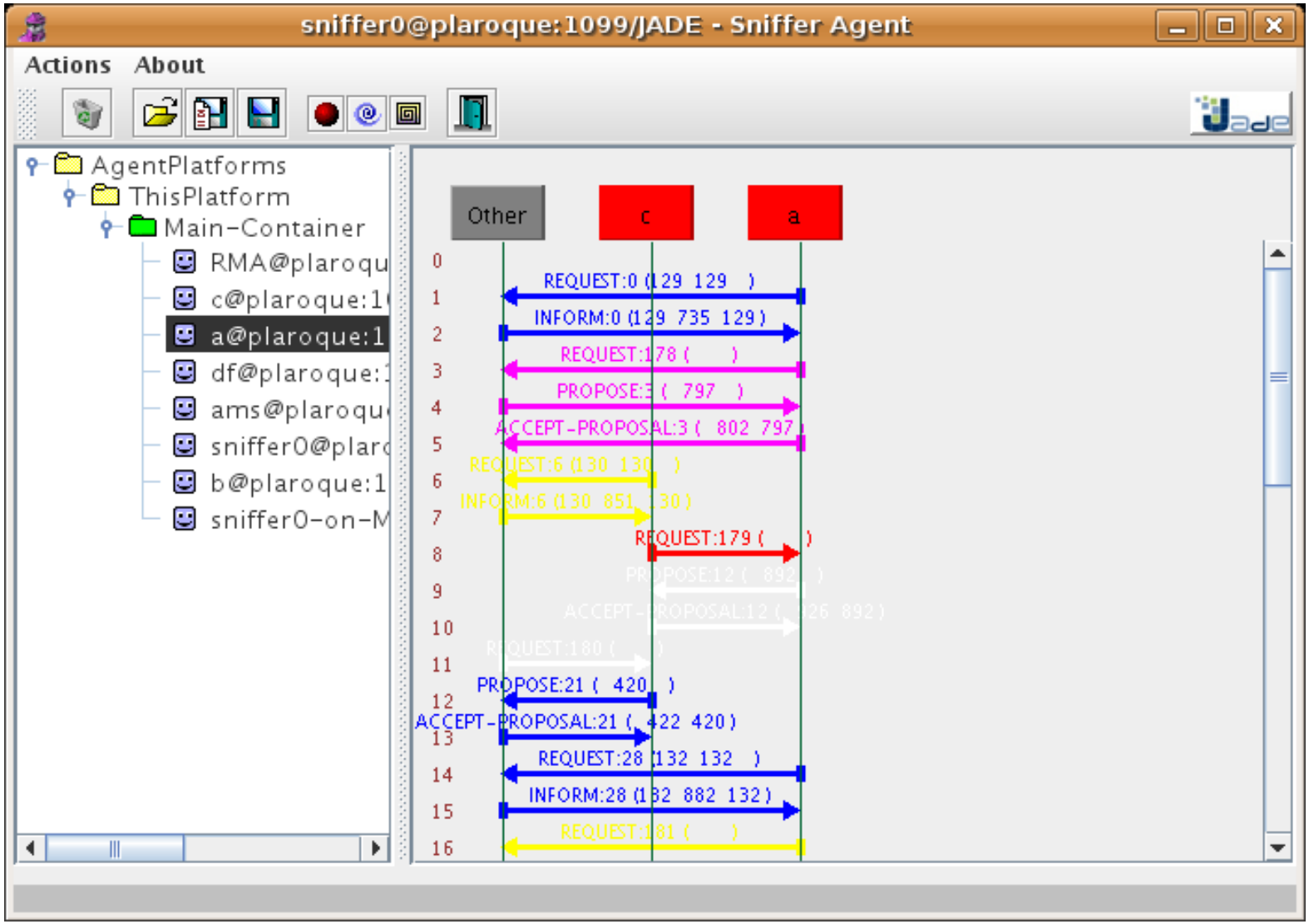
- Yellow pages services

Dummy Agent

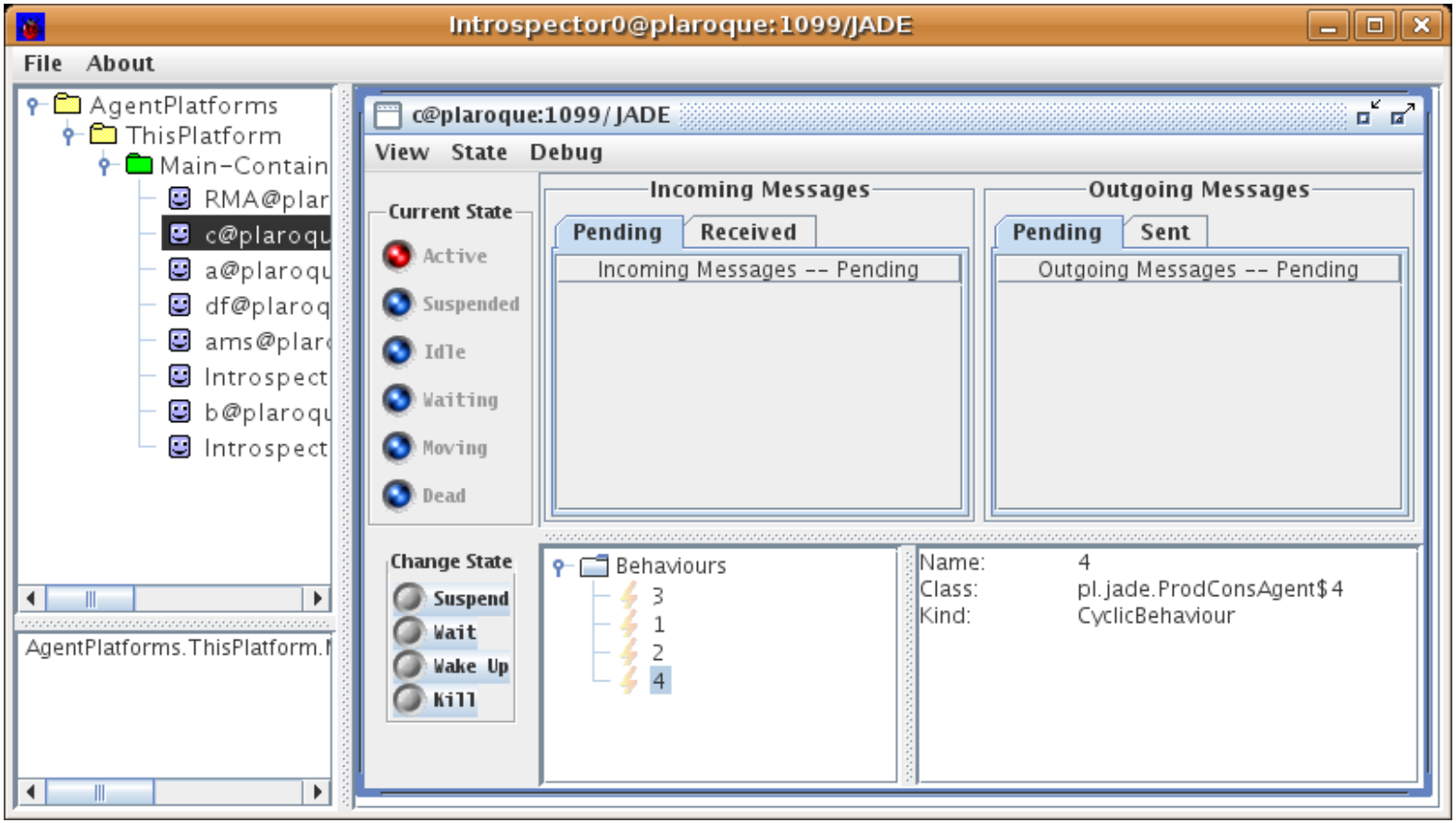


- Test tool
- Used to send and receive ACL messages

Sniffer Agent



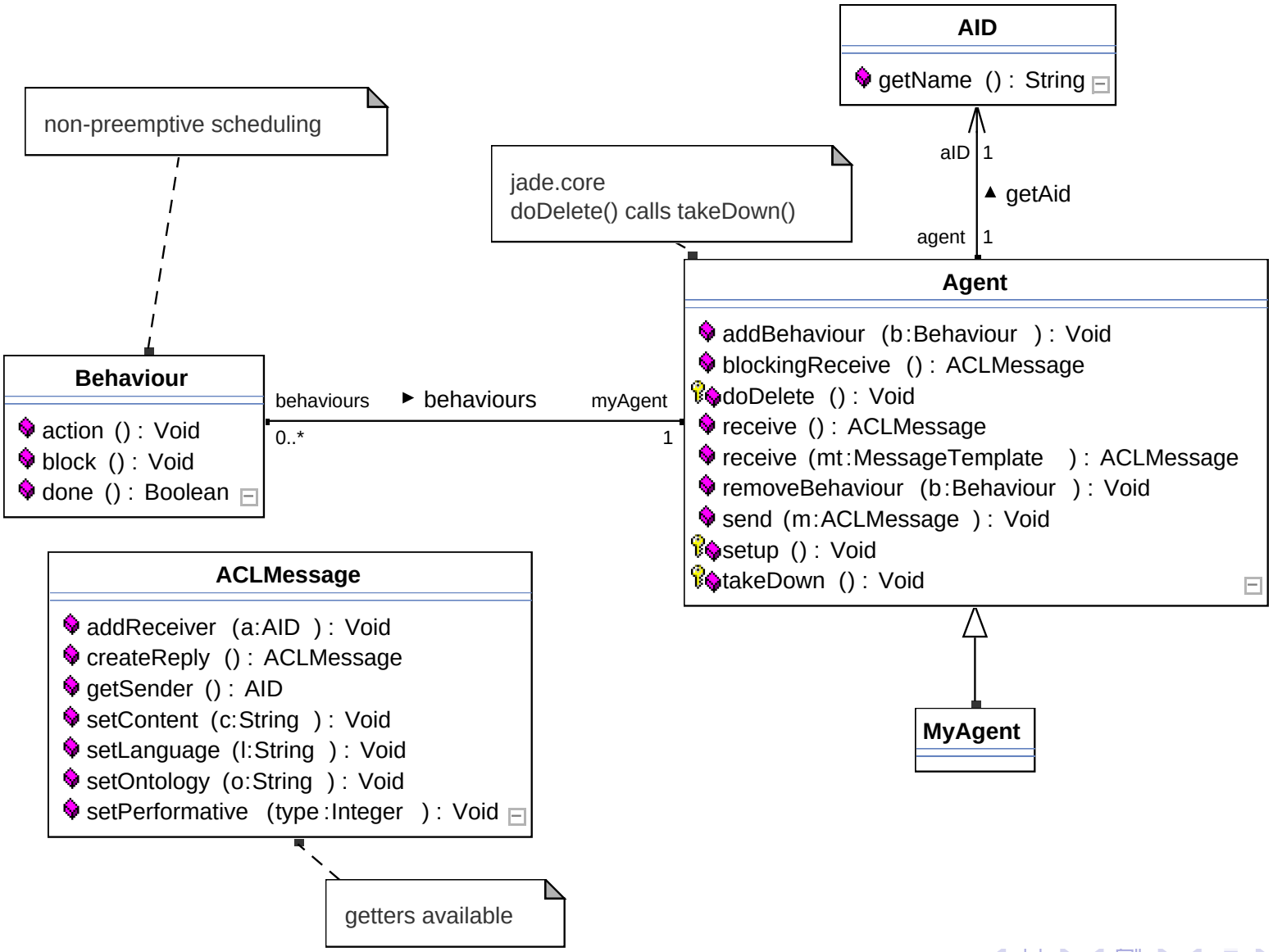
Introspector Agent



Outline

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

Using the Agent Class



Agent Setup

- Invoked when agent starts running
- Initialize instance variables whose value depends on command-line parameters(no `main()`): `getArguments()`
- Register agent in the DF
- In more complex cases, register languages and ontologies
- Attach one or more behaviours to the agents (attached behaviours are automatically started)

Passing Arguments

```
Object[] getArguments();
```

- array size is 0 if no argument
- gui arguments are separated with commas
- command-line arguments list is given just after the agent class, between '()'
- ex: `java jade.Boot -agents a:myPackage.MyAgent'("toto" 3)'`

Agent Termination

- `doDelete()` is called to destroy the agent
- `takeDown()` is then automatically run to do the cleanup
- Otherwise the agent is still "running", though idle

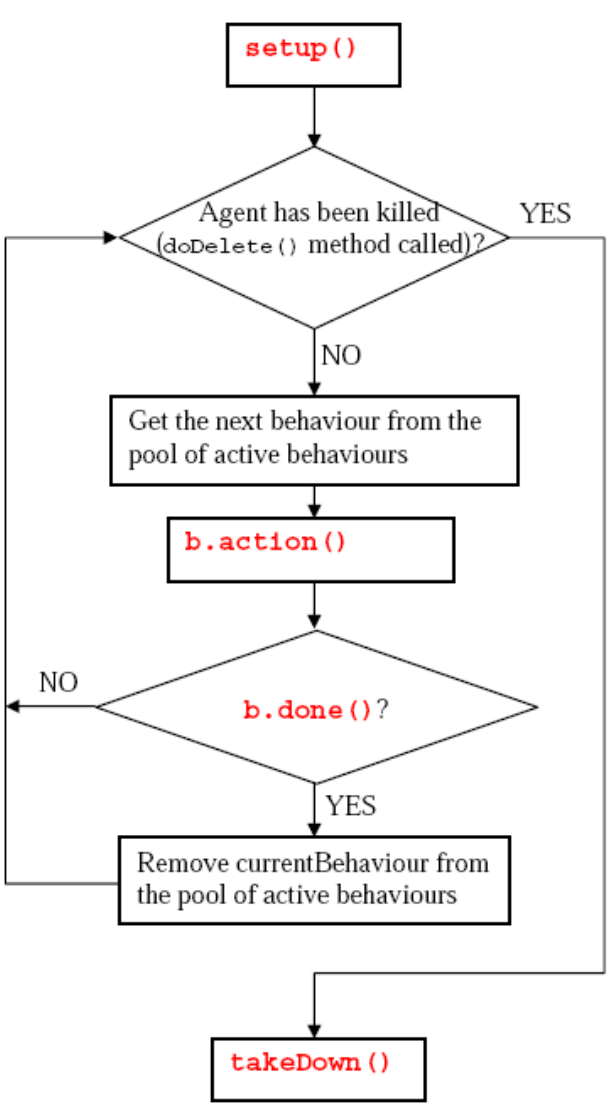
Outline

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

What is a Behaviour?

- Kind-of control thread for the agent
- Actually one single thread common to all the agent behaviours: non-preemptive scheduling
- `action()` is similar to `Thread.run()`
- New behaviours can be added at any time during agent life

Behaviour Scheduling



- Initializations
- Addition of initial behaviours

Highlighted in red the methods that programmers have to implement

- Agent "life" (execution of behaviours)

- Clean-up operations

- cooperative scheduling: once started, a behaviour runs until `action()` returns
- consequence: no `while(true)` `{...}` inside a behaviour!

Threads and Behaviours

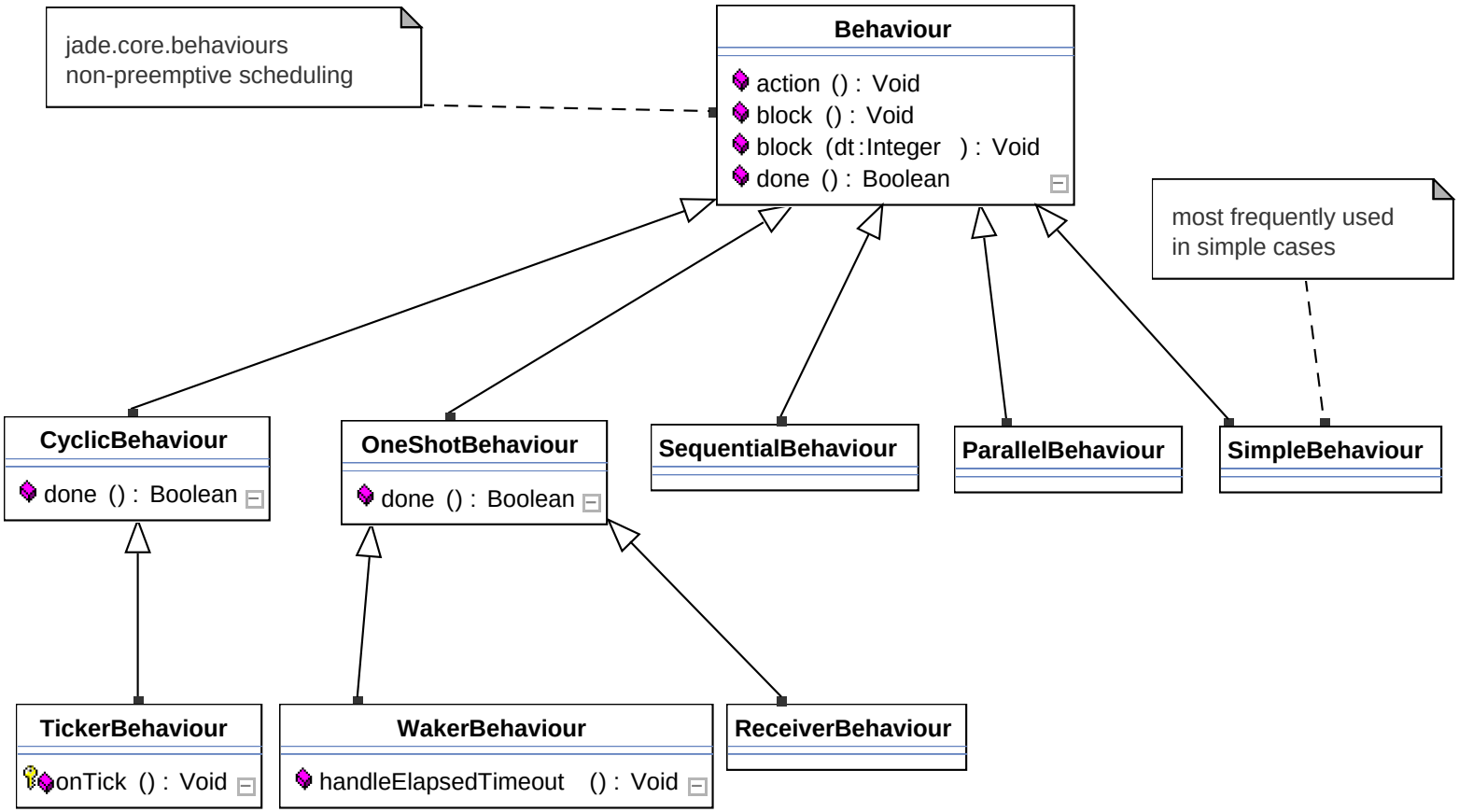
Threads > Behaviours

- additional efforts to the behaviour programmer (explicit context switches)

Behaviours > Threads

- far less resource consuming ($\rightarrow$ portable devices)
- no synchronization issue between concurrent behaviours competing for a resource
- simpler to "snapshot" the agent's state ($\rightarrow$ mobility)

Most Common Behaviours



Scheduling Behaviours in the Future

- WakerBehaviour: `action()` only runs `handleElapsedTime()` after given timeout, then completes

```
protected void setup() {
  System.out.println("Adding waker behaviour");
  addBehaviour(new WakerBehaviour(this, 10000) {
    protected void handleElapsedTimeout() {
      // perform operation X
    }
  } );
}
```

- `TickerBehaviour`: repetitively runs `onTick()` every N ms

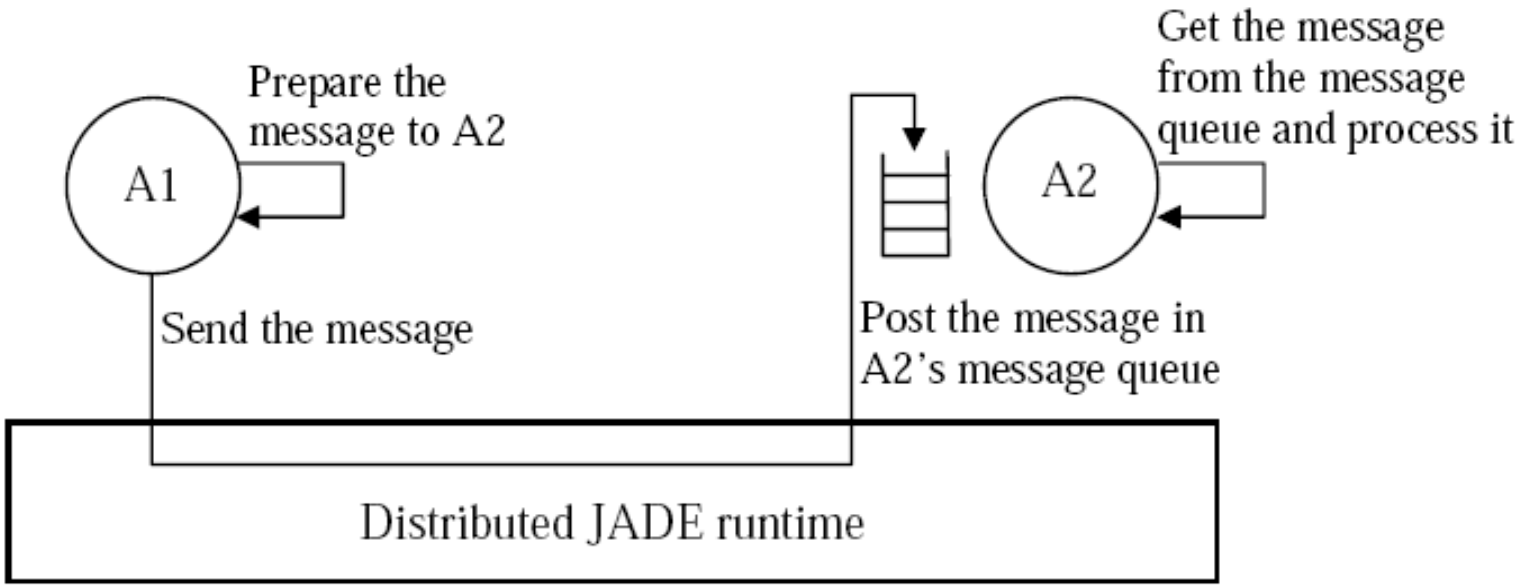
Sample Generic Stateful Behaviour

```
public class MyThreeStepBehaviour extends Behaviour {
  private int step = 0;
  public void action() {
    switch (step) {
    case 0: // perform operation X
      step++; break;
    case 1: // perform operation Y
      step++; break;
    case 2: // perform operation Z
      step++; break;
    }
  }
  public boolean done() {
    return step == 3;
  }
```

Outline

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

Jade Message Passing Principle



- Message queue is not strictly FIFO: existing selection mechanism

ACL Brief Overview

- Agent Communication Language (see [1])
- Agents can `send()` and `receive()` messages
- Message important fields:
  - the sender
  - the list of receivers
  - the "performative" (message category: REQUEST, INFORM etc.)
  - the content
  - the content language (used to encode and parse content: syntactic level)
  - the ontology (vocabulary and its "meaning" - ex. *kill*: semantic level)
- Most fields can be used to filter messages from th message box (`MessageTemplate`)

Sending a Message

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("Peter", AID.ISLOCALNAME));
msg.setLanguage("English");
msg.setOntology("application programming");
msg.setContent("Paul has been killed");
send(msg);
```

Receiving a Message

```
public void action() {
  ...
  ACLMessage msg = receive();
  if (msg != null) {
    // Process the message
  }
  else block() // schedules next execution
}
```

Note:

`ACLMessage receive(MessageTemplate mt);`
Allows to select only messages matching `mt`

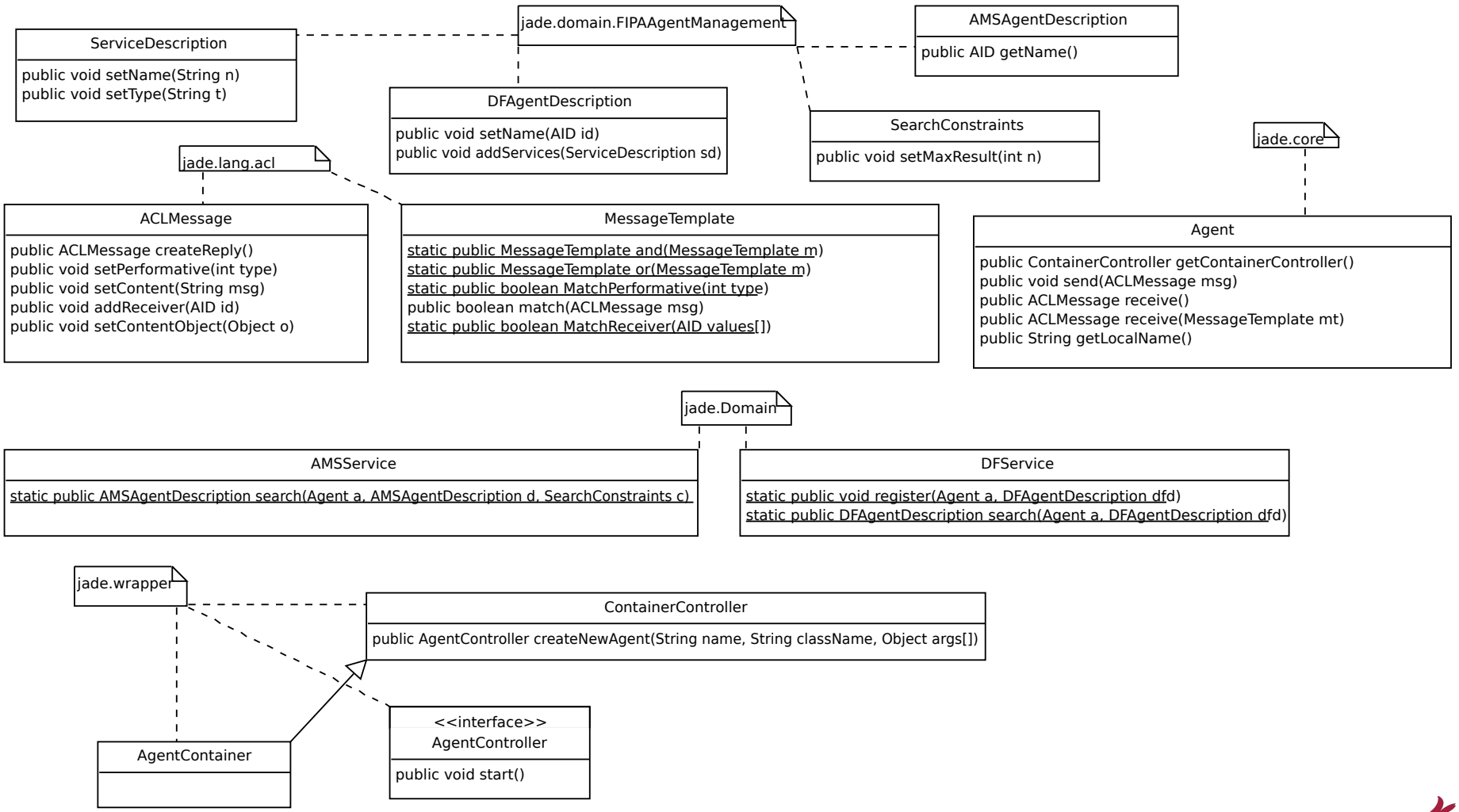
Blocking Receive

- By default, `receive()` is a non-blocking method (return null if no message)
- A `blockingReceive()` method exists

the thread is then blocked:

- *always* use `receive()` inside `action()`
- use `blockingReceive()` *only* in `setup()` / `takeDown()`

Message-related classes



Outline

1. Introduction
   - Terminology
   - Definition
   - Multi-Agent Systems
   - A Simple Agent Classification
2. Communicative (Internet) Agents
   - Introduction to FIPA
3. Introduction to Jade
   - What is JADE?
   - Jade Structure and Components
   - Agent creation
   - Agent Tasks: Behaviours
   - Agent Communications: Messages
   - Yellow Pages Service

Need for a DF

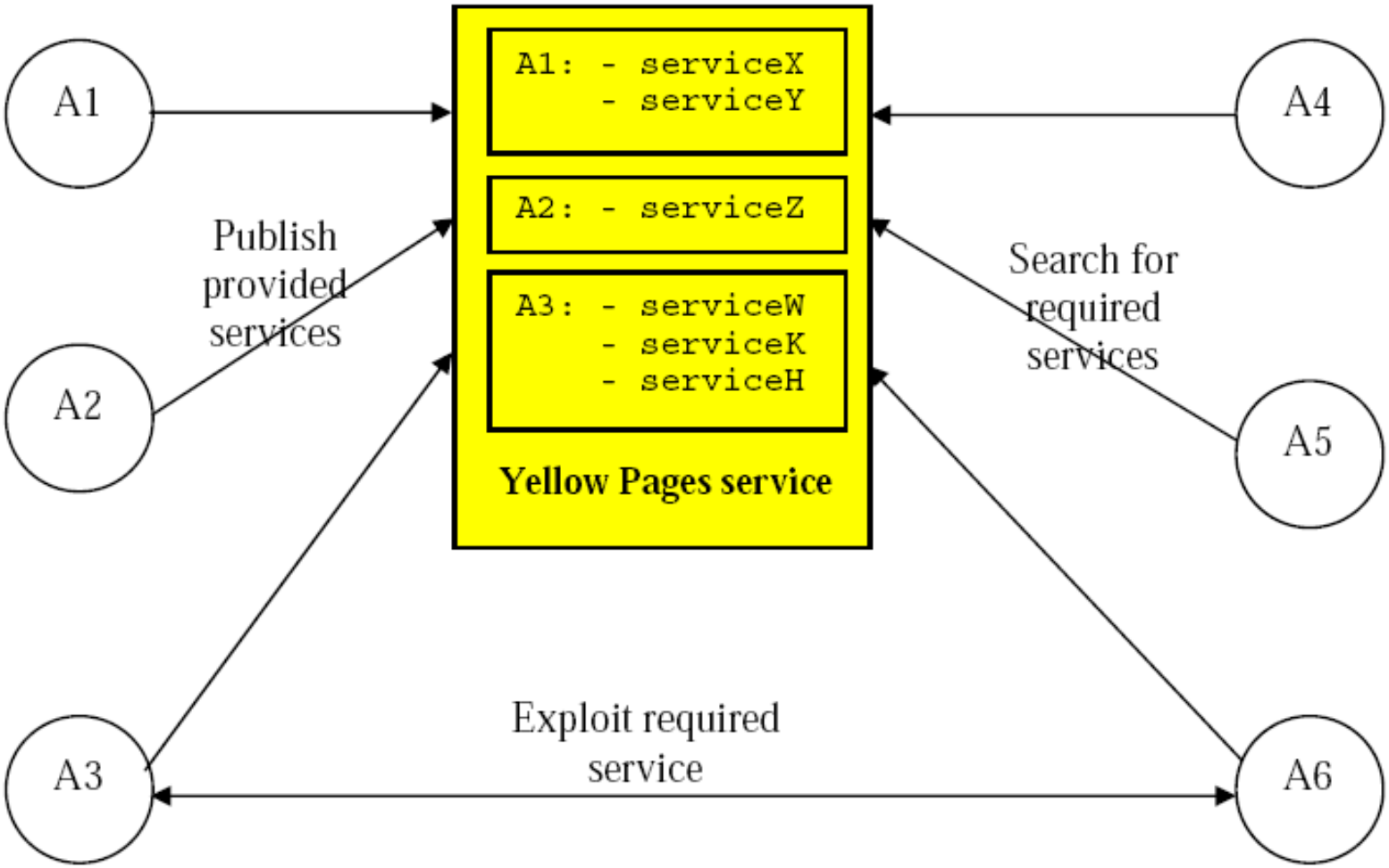
- New agents can appear, existing agents can die
- Agents can acquire new capabilities (propose different services)

$\longrightarrow$ need for a mechanism for agents

1. to register as providing services
2. to discover which agents can provide required service

Each platform as a local DF, whose name is `df`

Jade YP Service



Publishing a Service

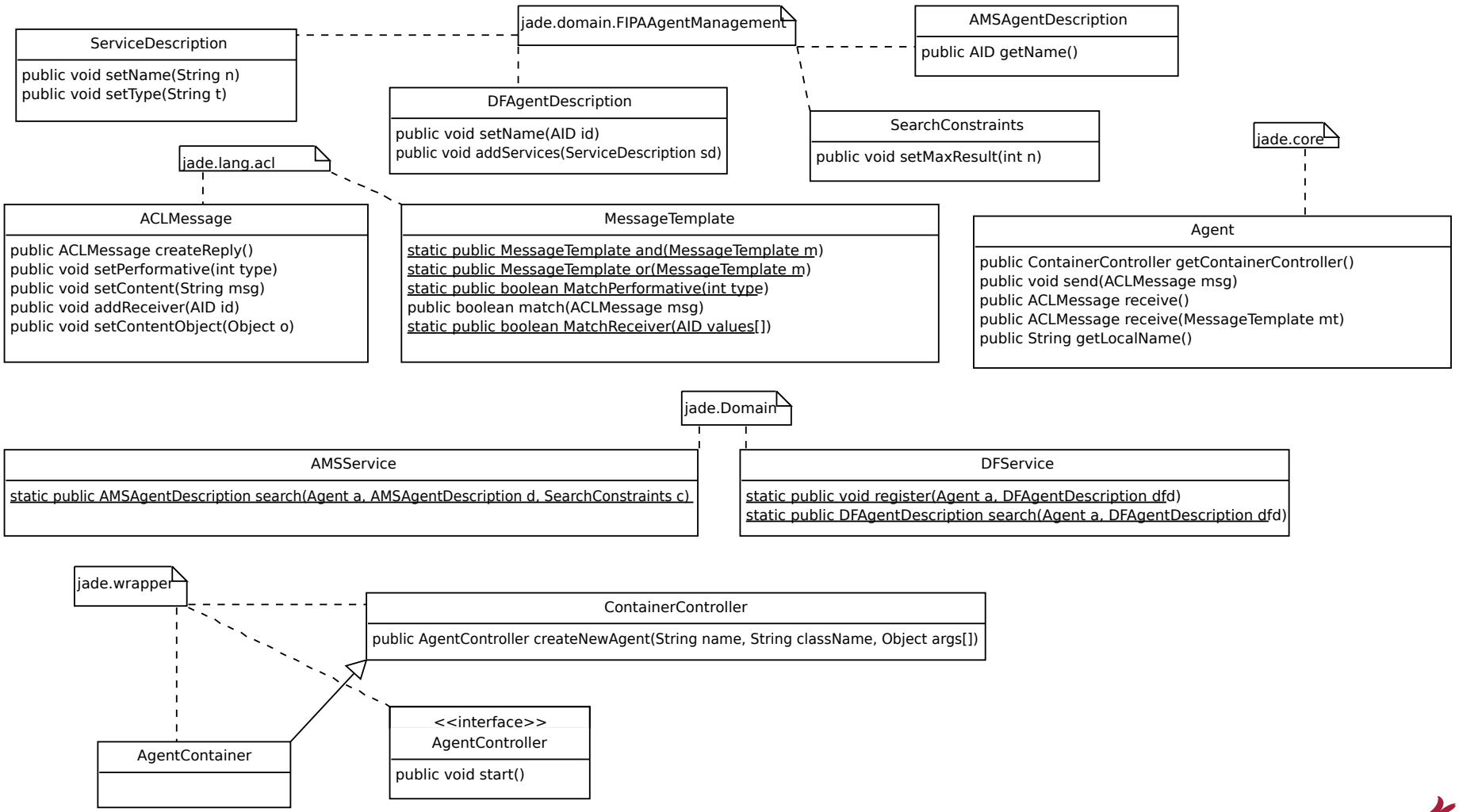
The agent registers to the DF by providing a description containing

- The agent ID
- The list of proposed services
- The list of languages / ontologies needed to use the services, if any

Searching for a Service

- The agent searches the DF by providing a template description (`search()` method)
- Agents can also *subscribe* to the DF to be notified when a needed service is available

DF-related Classes



Registration in Practice

Usually in setup():

```
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID());
ServiceDescription sd = new ServiceDescription();
sd.setType("car-rental");
sd.setName("Hertz-Cergy");
dfd.addServices(sd);
try {
  DFService.register(this, dfd);
} catch (FIPAException fe) {
  fe.printStackTrace();
}
```

De-registration in Practice

Usually in `takeDown()`:

```
// Deregister from the yellow pages
try {
  DFService.deregister(this);
} catch (FIPAException fe) {
  fe.printStackTrace();
}
```

Searching in Practice

Usually in setup() or action():

```
DFAgentDescription template = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
sd.setType("car-rental");
template.addServices(sd);
try {
  DFAgentDescription[] result
    = DFService.search(myAgent, template);
  agents = new AID[result.length];
  for (int i = 0; i < result.length; ++i) {
    agents[i] = result[i].getName();
  }
} catch (FIPAException fe) {
  fe.printStackTrace();
}
```

References I

- FIPA website: http://www.fipa.org
- JADE website: http://jade.cselt.it
- CORBA website: http://www.corba.org/
- OMG website: http://www.omg.org
- Java RMI website: http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp
- An interesting website on agents: http://www.agentlink.org/
- Nilsson's web page on agents: http://www.robotics.stanford.edu/users/nilsson/trweb/tr.html

References II

- M. Nikraz, G. Caire, P. A. Bahri, "A Methodology for the Analysis and Design of MAS using JADE", `http://jade.cselt.it/doc/JADE_methodology_website_version.pdf`
- "Multi-Agent System: An Introduction to Distributed Artificial Intelligence", Jacques Ferber, Harlow: Addison Wesley Longman, 1999 Paper: ISBN 0-201-36048-9
- "Integrating Tools and Infrastructures for Generic Multi-Agent Systems", Olivier Gutknecht, Jacques Ferber, Fabien Michel, Proceedings of the Fifth International Conference on Autonomous Agents, 2001
- Arnaud Revel, "From Robots to Web-Agents: Building Cognitive Software Agents for Web-Information Retrieval by Taking Inspiration from Experience in Robotics", Web Intelligence 2005: 434-437