

Sistemas multi-agente

Profesor:

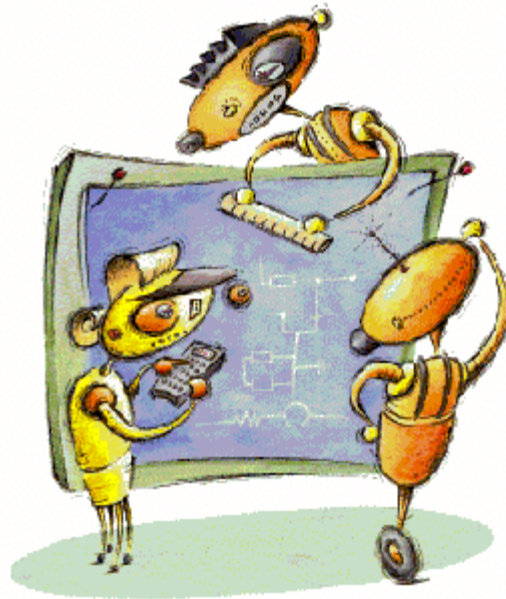
Dr. José Octavio Gutiérrez García

octavio.gutierrez@itam.mx

Actores / Agentes

- Desarrollado por Carl Hewitt, Henry Baker y Gul Agha.
- Los actores son objetos **autónomos** y **concurrentes** que se ejecutan de forma **asincrónica**.
- El modelo de actores (**agentes**) proporciona mecanismos flexibles para la construcción de sistemas de software **distribuidos** y **concurrentes**.



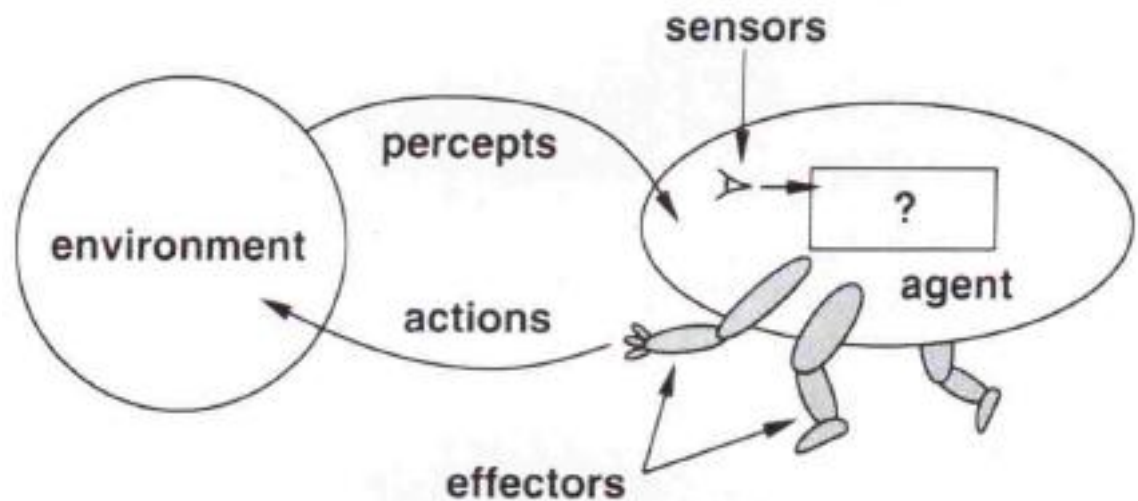


¿Qué es un agente?

¿Qué es un agente?

¿Qué es un agente?

- Un agente es un sistema informático que es capaz de realizar **acciones independientes** en nombre de su propietario [Wooldridge].
- Un agente es una entidad **que percibe su entorno** y actúa sobre él [Russell y Norvig].



¿Qué es un agente?

- "**Agente de Software**" es un tipo particular de agente, que habita en las computadoras y redes, ayudar a los usuarios con tareas informáticas.

Los agentes autónomos son sistemas computacionales que habitan en algún entorno complejo y dinámico. Los agentes perciben actúan autónomamente en este entorno, para cumplir con una **serie de objetivos** [Maes]



Características de un agente

- Actúan en nombre de un usuario u otro programa
- Autónomo
- Reactivo
- Proactivo
- Razona
- Toma de decisiones
- Aprendizaje
- Adaptación
- Interacción con otros agentes (ejemplos: negociación y cooperación)

¿Inteligencia?

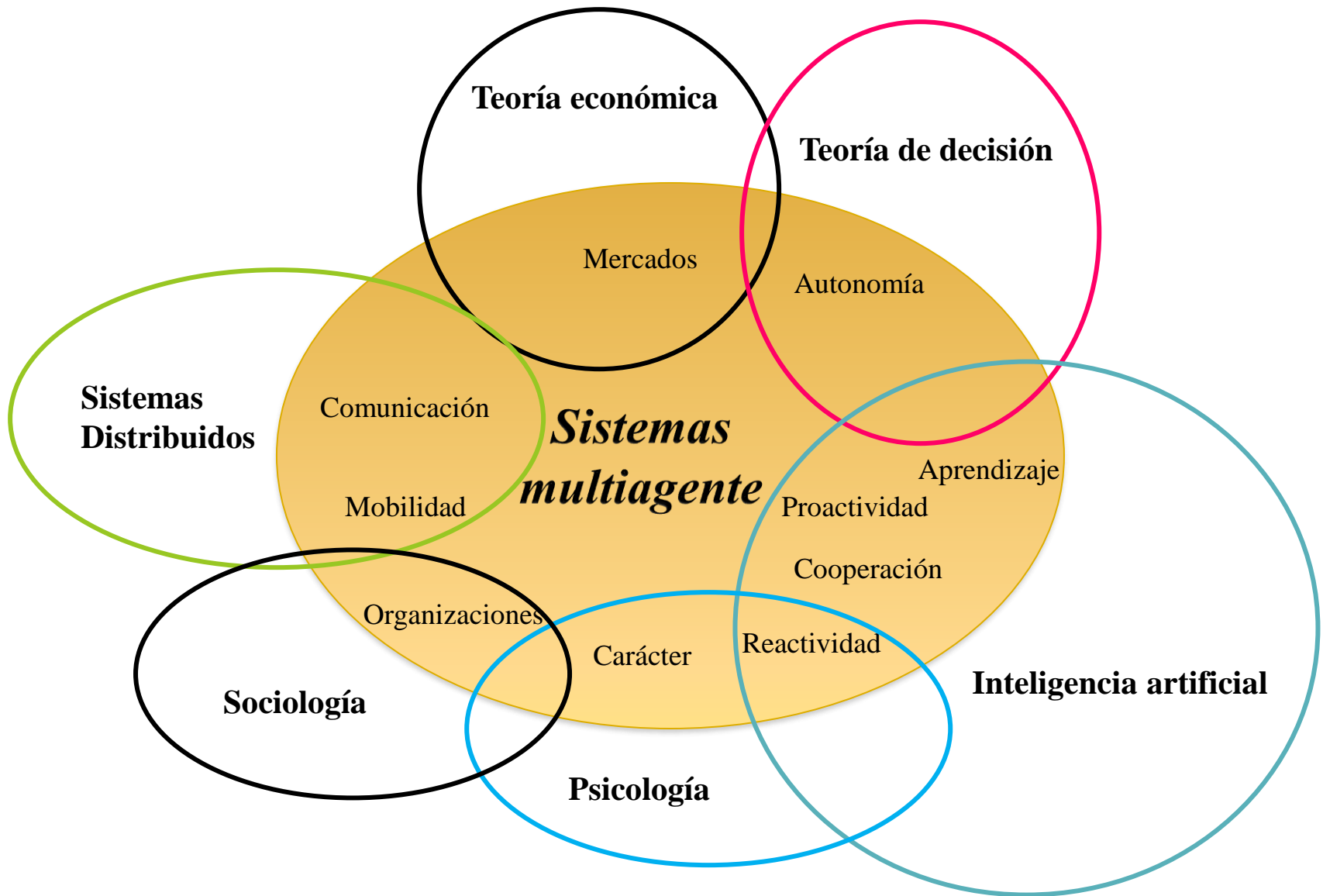


Mi definición de Agente

- Los agentes de software son capaces de **resolver problemas autónomamente** interactuando flexiblemente (es decir, negociando y colaborando) en un entorno dinámico para lograr objetivos **individuales y globales** mediante el **uso** de técnicas de **inteligencia artificial**.



Vínculos de los sistemas multiagente con otras disciplinas



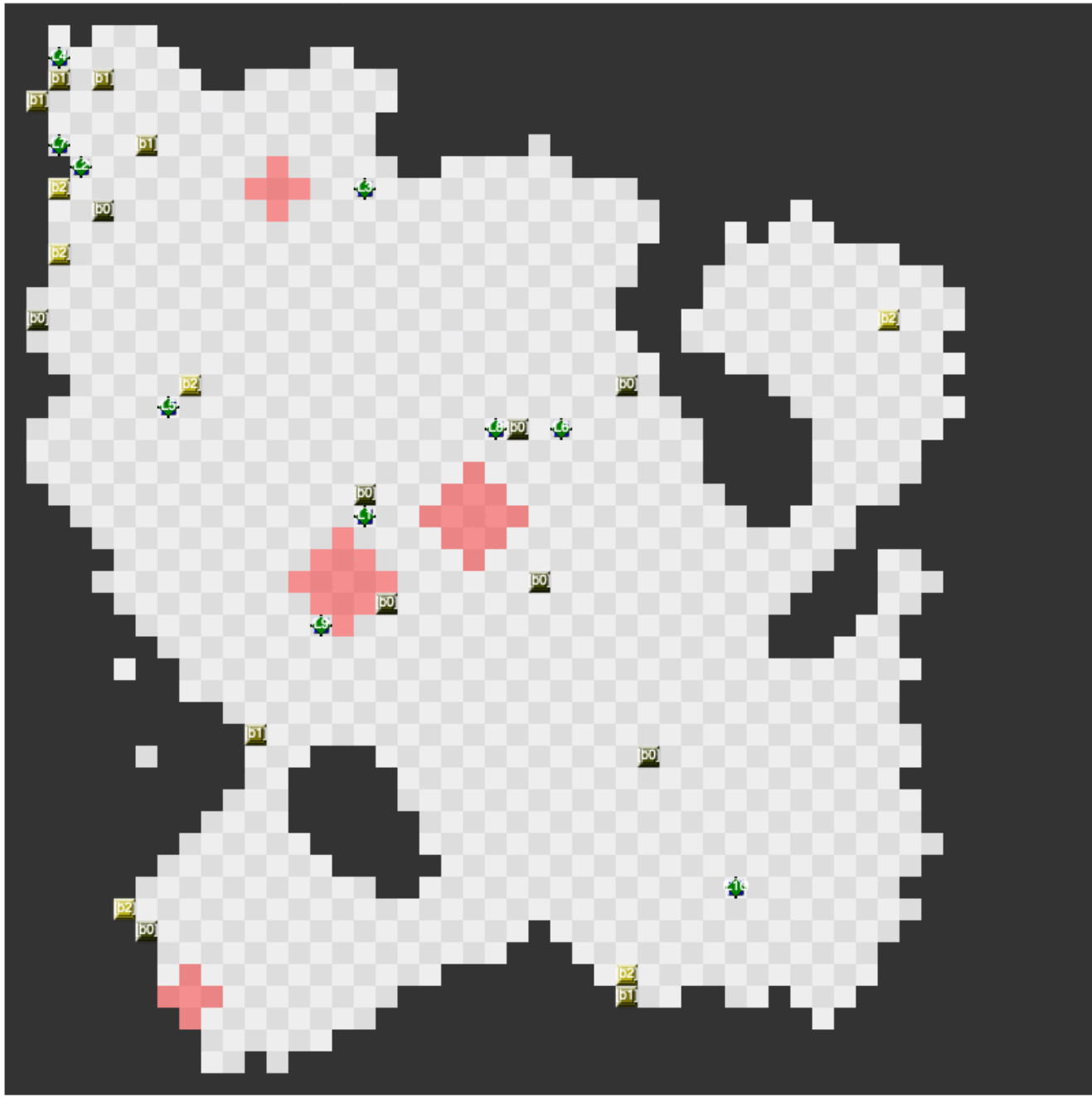
Simulación de multitudes



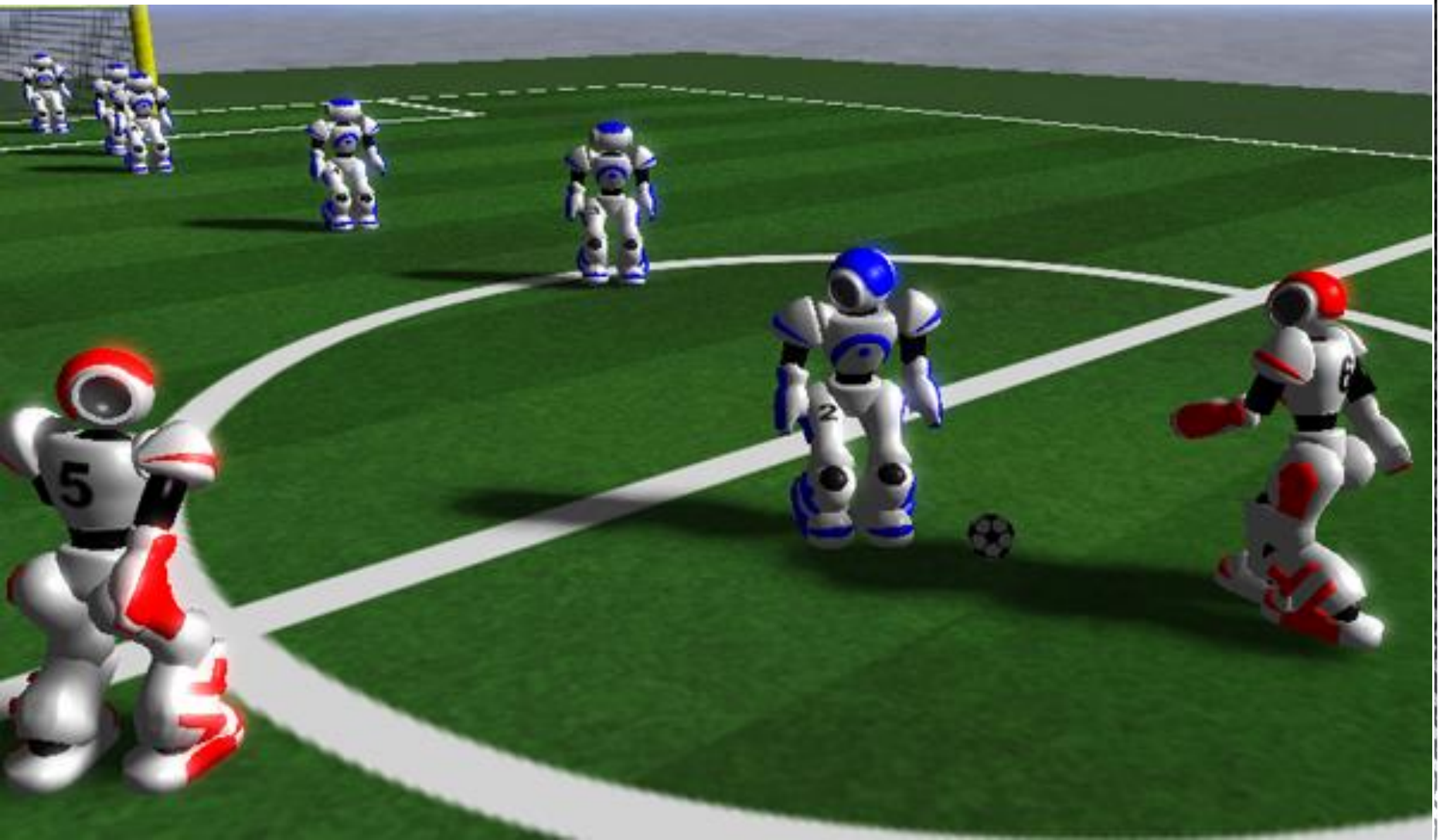
Reconocimiento de áreas



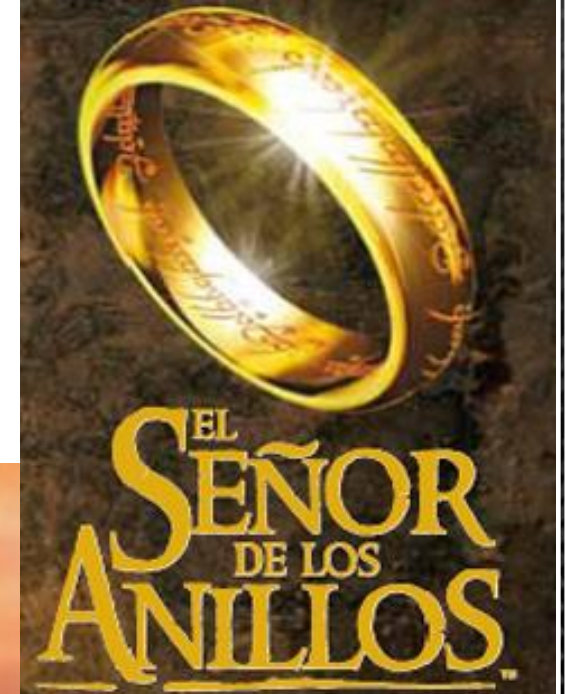
Trabajo en Equipo



Programación de videojuegos



Secuencias de Animación en Películas



Simulación de humanos virtuales



Simuladores de tráfico urbano



Bolsa de valores



Agent-based Cloud computing deals with the design, implementation, and exploitation of agent-based problem solving techniques to achieve complex objectives in continuously evolving Cloud-computing environments.

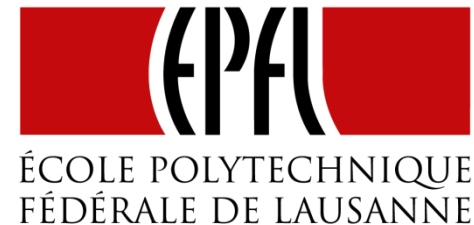




Java Agent DEvelopment framework

JADE - Java Agent Development framework

Imperial College
London



Rockwell
Automation

JADE - Java Agent Development framework

JADE es un middleware

- Aplicaciones Multi-agente Peer-to-Peer.
- El ciclo de vida de los agentes y la movilidad.
- Servicios de páginas Blancas y Amarillas.
- Transporte y análisis de mensajes Peer-to-Peer
- Seguridad
- Planificación de las tareas del agente.
- Herramientas gráficas para la monitorización, logs y depuración.



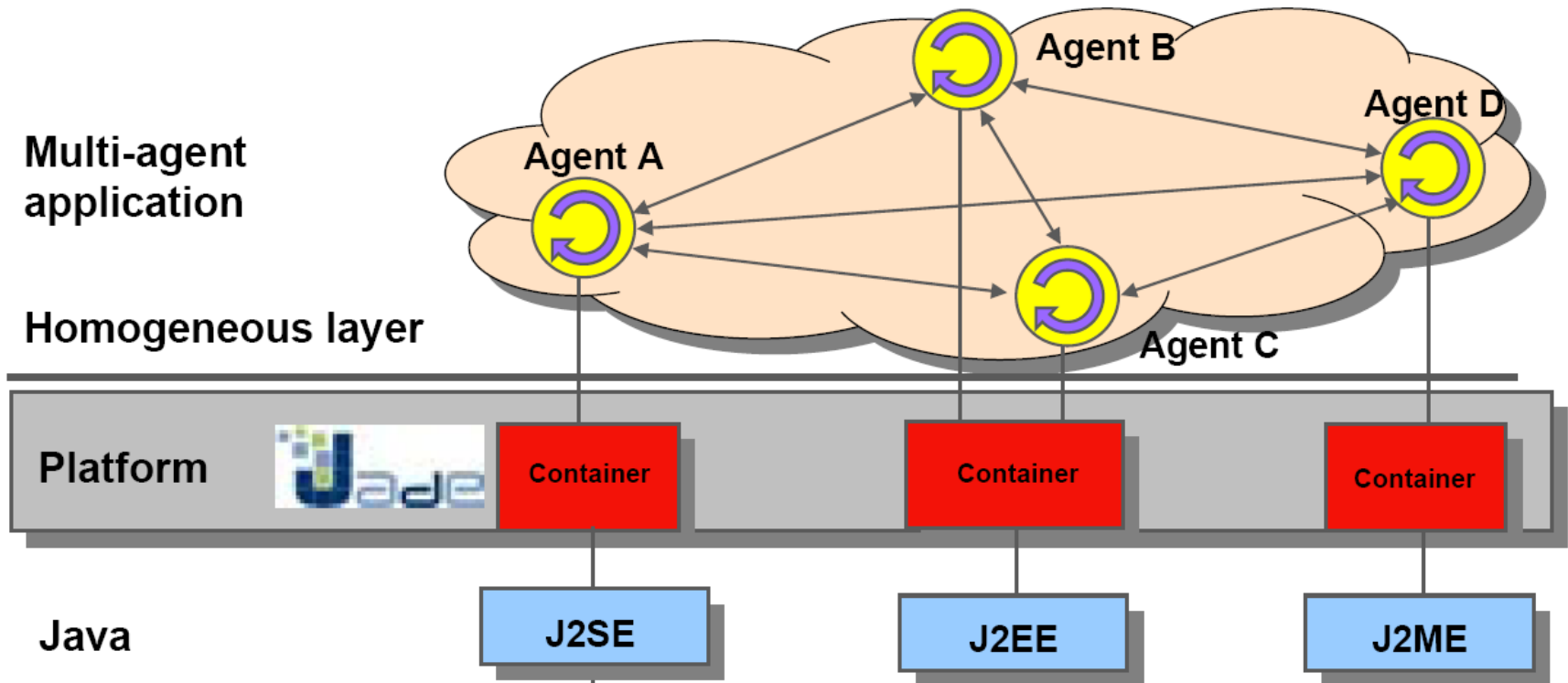
JADE - Java Agent Development framework

- Desarrollado en Java
 - Ejecutable sobre todas las máquinas virtuales de Java
- Distribuido en Open Source bajo licencia LGPL
 - Descargable en <http://jade.tilab.com>
- JADE cumple las especificaciones FIPA



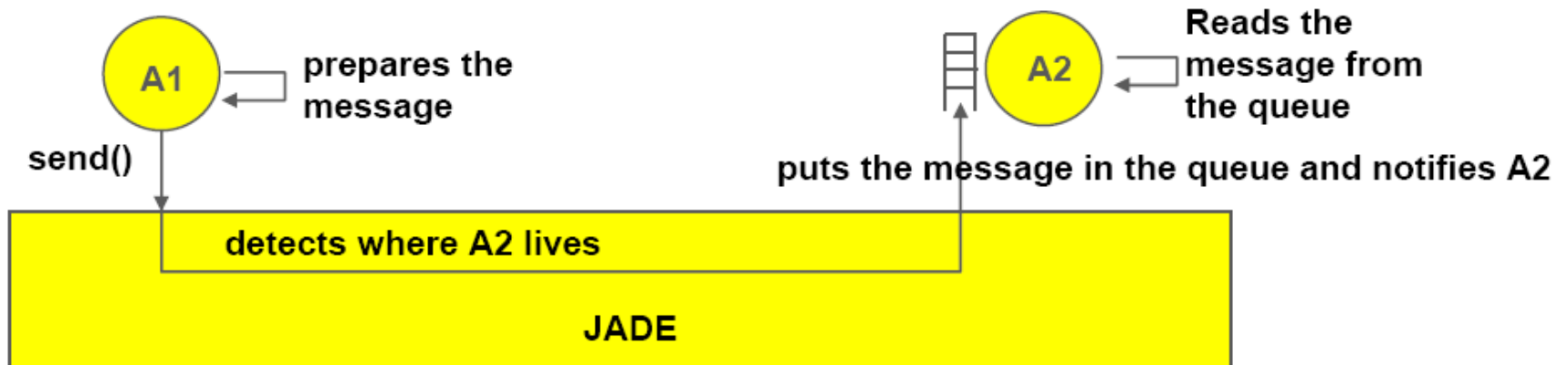
Modelo arquitectónico

✓



Modelo de comunicación

- Basado en paso de mensajes asíncronos



Plataforma

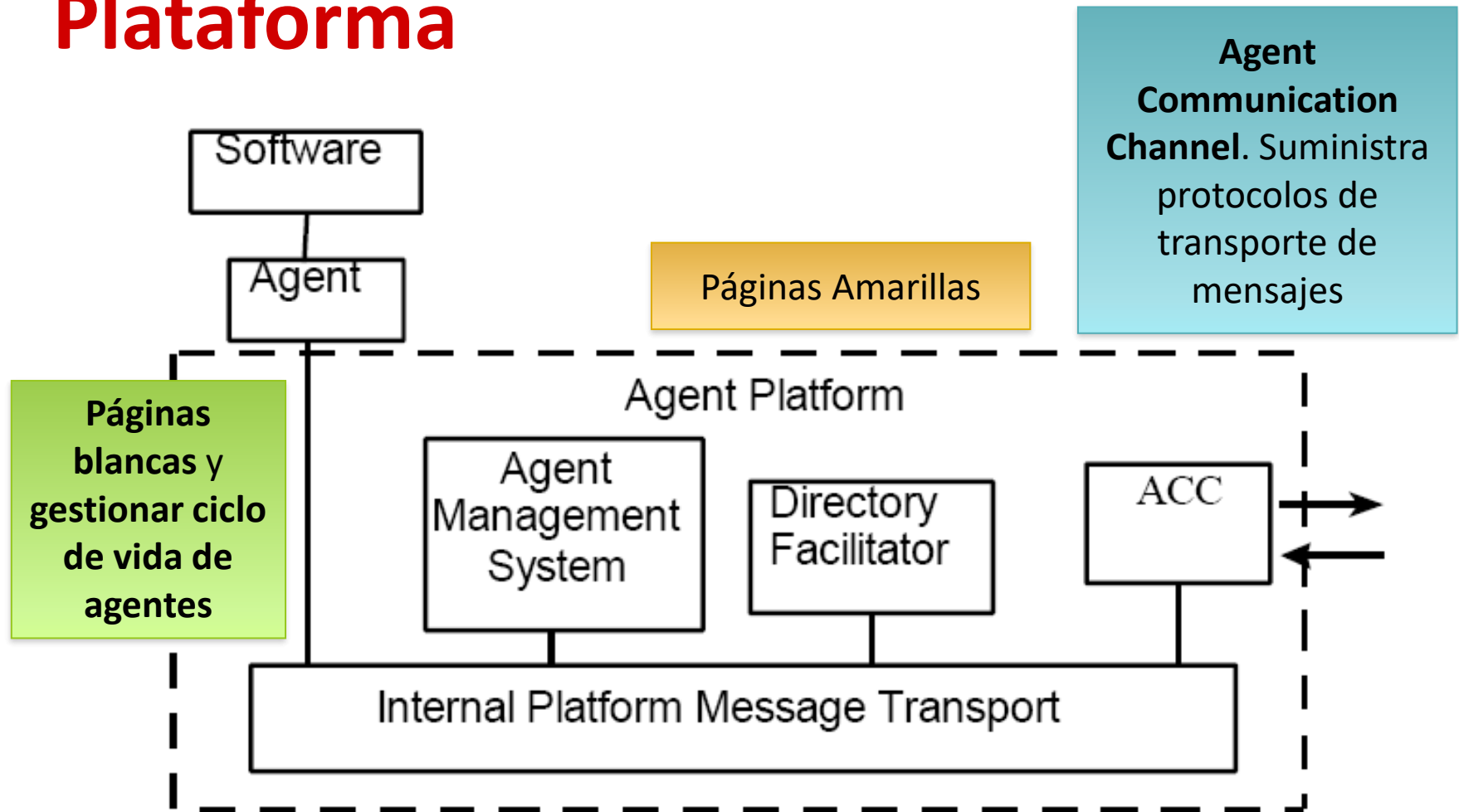


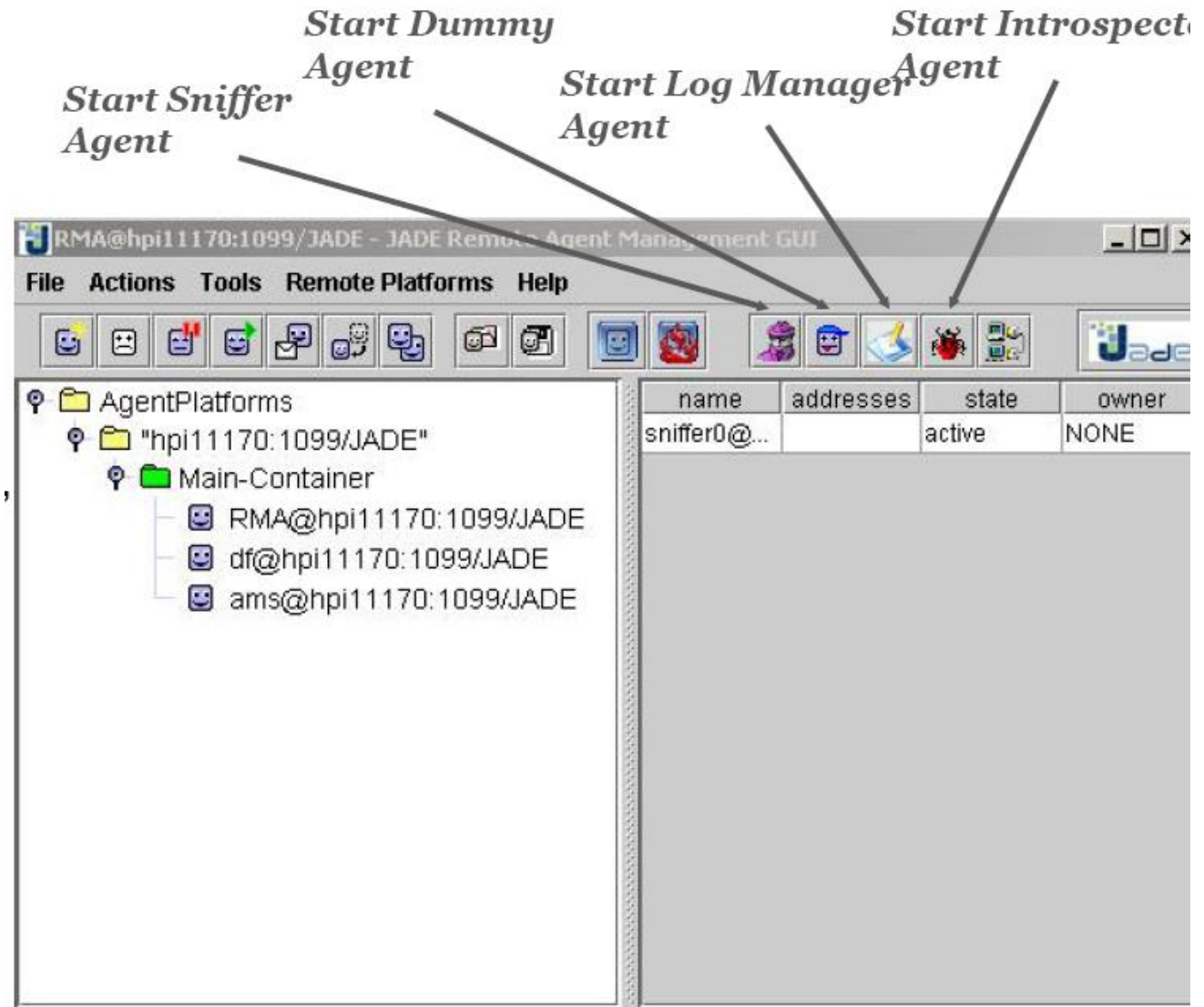
Figure 1 – FIPA reference model of an Agent Platform

Principales herramientas de Jade

- **RMA (Remote Monitoring Agent)**
- **DummyAgent**
- **SnifferAgent**
- **IntrospectorAgent**
- **Log Manager Agent**
- **DF (Directory Facilitator) GUI**

Remote Monitoring Agent

- **Monitorear** y controlar la plataforma y todos sus contenedores remotos.
- Gestión remota del **ciclo de vida** de los agentes (creación, suspender, reactivar, matar, migrar, clonar)
- Lanzar otras **herramientas** gráficas.



Dummy Agent



- Compone y envía mensajes “ad-hoc”

da0@IBM10312:1099/JADE - DummyAgent

General Current message Queued message

ACLMessage Envelope

Sender: Set

Receivers: da1@IBM10312:1099/JADE

Reply-to:

Communicative act: propose

Content:

Language:

Encoding:

Ontology:

Protocol: Null

Conversation-id:

In-reply-to:

Reply-with: BM10312:1099/JADE1096016308500

Reply-by: Set

User Properties:

24/09/04 10:59: PROPOSE
24/09/04 10:58: QUERY-IF
24/09/04 10:58: FAILURE
24/09/04 10:58: ACCEPT-PROPOSAL
24/09/04 10:58: CANCEL
24/09/04 10:58: ACCEPT-PROPOSAL
24/09/04 10:52: CONFIRM

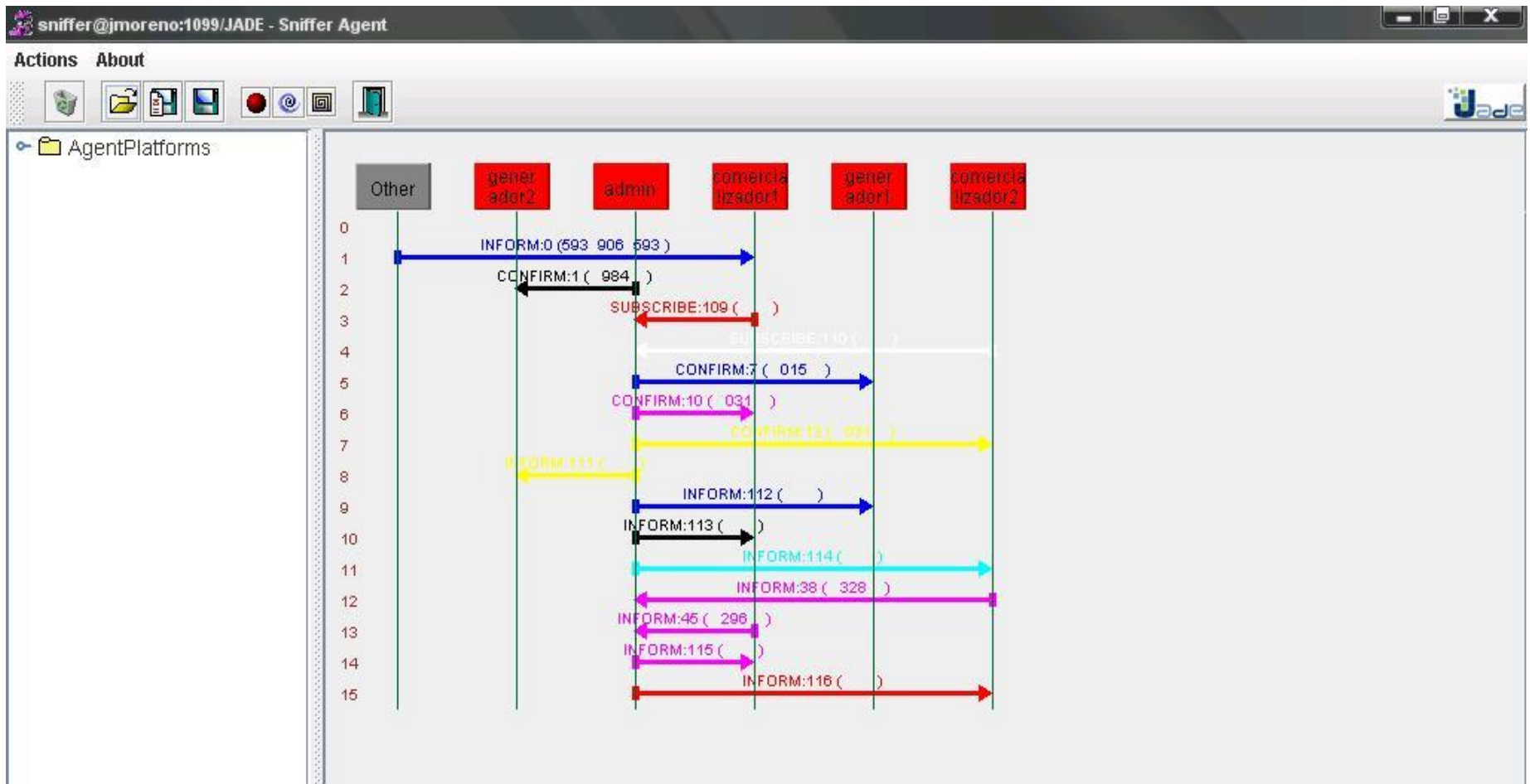
Sniffer Agent



- Muestra el **flujo de interacciones** entre los agentes seleccionados.
- Muestra el **contenido** de cada **mensajes**

The screenshot displays the 'Sniffer Agent' application window. The title bar reads 'sniffer0@hpi11170:1099/JADE - Sniffer Agent'. The interface includes a menu bar with 'Actions' and 'About', a toolbar with various icons, and a left-hand tree view showing the hierarchy of agent platforms. The tree view includes 'AgentPlatforms', '"hpi11170:1099/JADE"', and 'Main-Container', with several agent instances listed below, including 'RMA@hpi1117', 'sniffer0@hpi11', 'df@hpi11170:1', 'ams@hpi11170', and 'sniffer0-on-Mai'. The main area shows a message flow diagram with three vertical lifelines labeled 'Other', 'df', and 'ai'. Four messages are shown: '0' (blue arrow from 'Other' to 'df'), '1' (blue arrow from 'df' to 'ai'), '2' (blue arrow from 'ai' to 'df'), and '3' (blue arrow from 'df' to 'ai'). The messages are labeled 'INFORM:0 (591 075)', 'REQUEST:0 (591)', and 'INFORM:0 (591 813)'. An 'ACL Message' dialog box is open on the right, showing details for a message. The dialog has tabs for 'ACLMessage' and 'Envelope'. The 'ACLMessage' tab is active, displaying fields for 'Sender' (ca0@IBM10312:1099/JADE), 'Receivers' (df@IBM10312:1099/JADE), 'Reply to' (empty), 'Communicative act' (confirm), 'Content' (alive), 'Language' (PlainText), 'Encoding' (empty), 'Ontology' (empty), 'Protocol' (ping-protocol), 'Conversation-id' (conversation1), 'In-reply-to' (empty), 'Reply with' (reply1), 'Reply by' (20041024T085219000Z), and 'User Properties' (empty). The 'OK' button is at the bottom right of the dialog.

Sniffer Agent

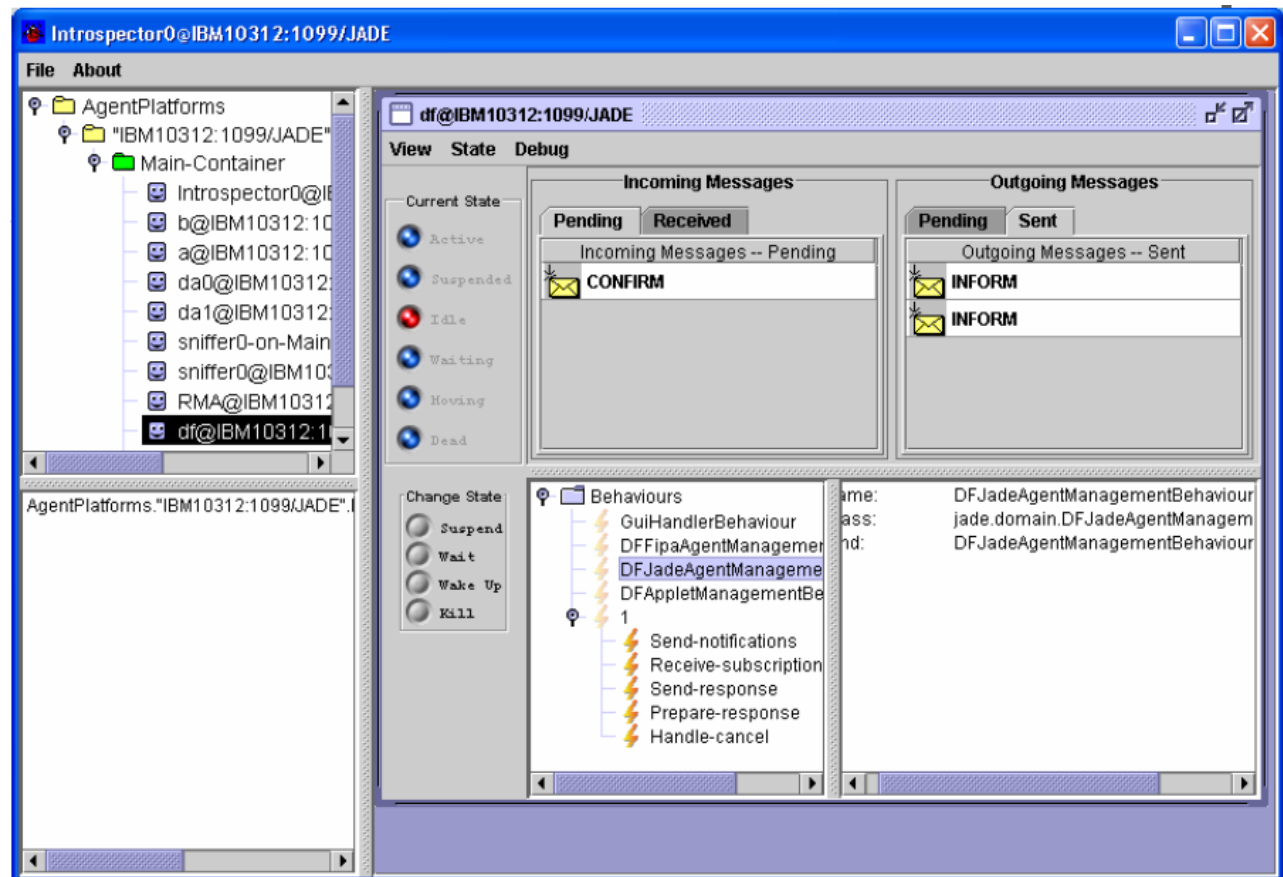


Introspector Agent



- **Monitorear estado interno del agente**
 - Mensajes recibidos/enviados/pendientes
 - Comportamientos planificados (activo, bloqueado) y sub-comportamientos

- **Depuración**
 - step-by-step
 - break points



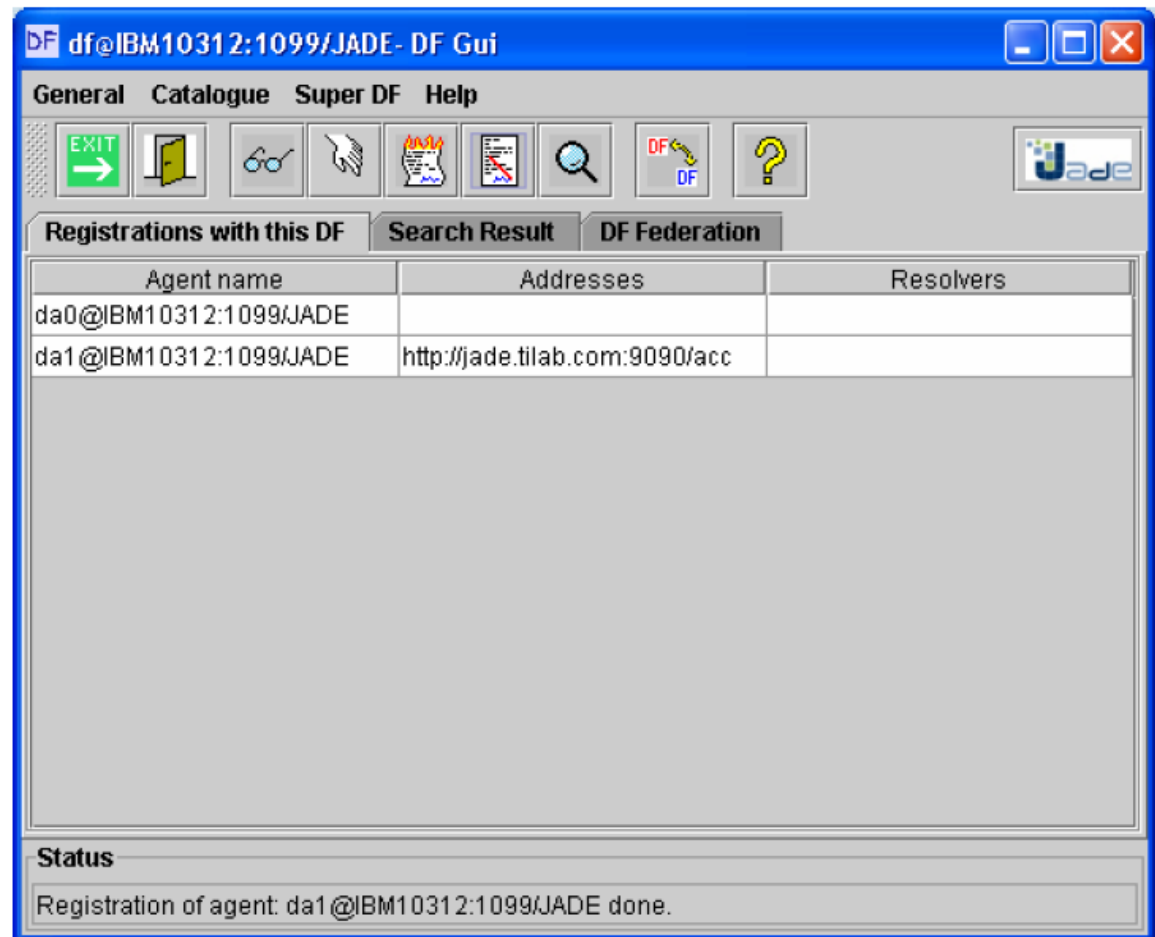
Log Manager Agent



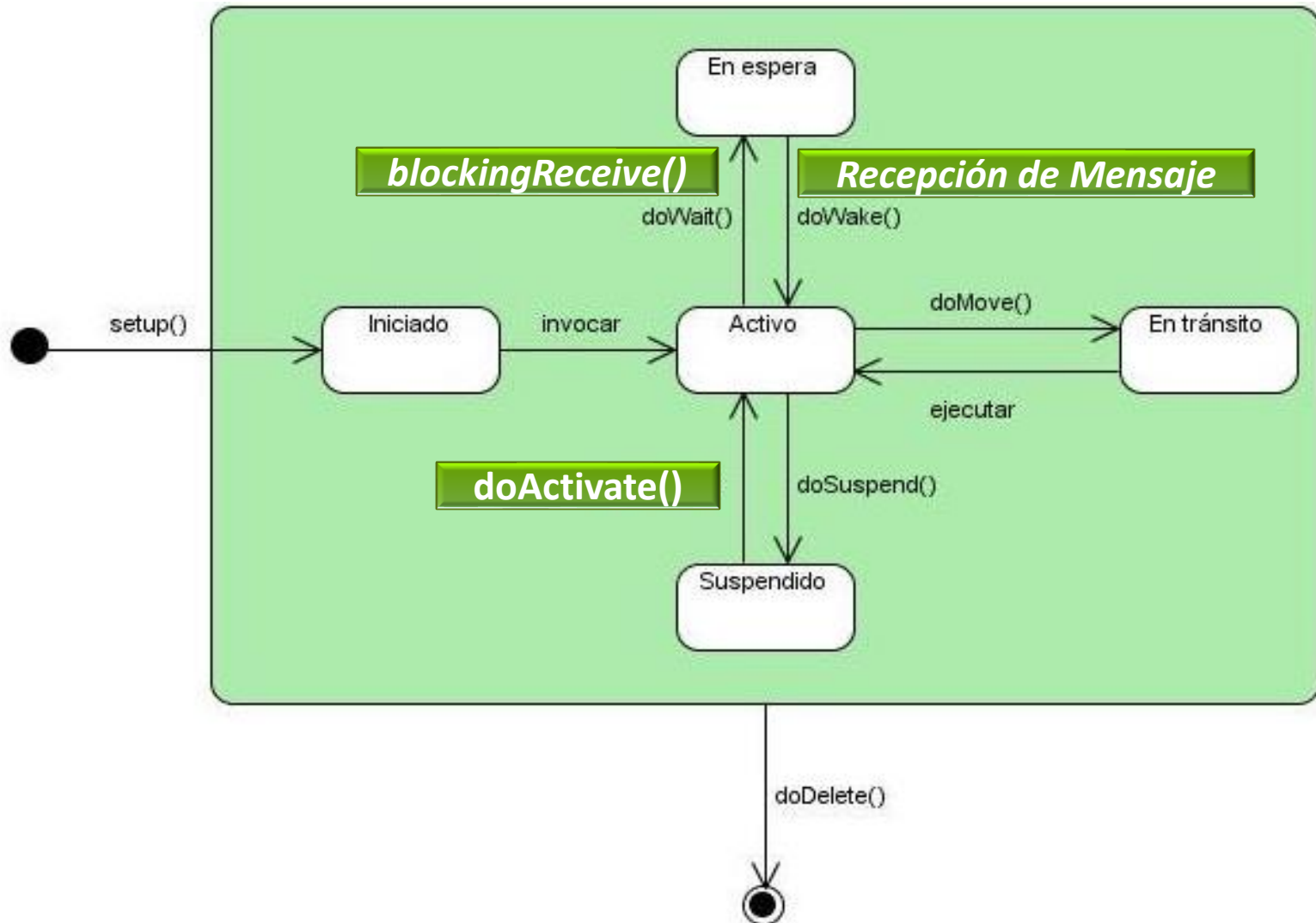
da0@hpi11170:1099/JADE - LogManagerAgent			
Logger Name	Set Level	Handlers	Set log file
jade.content.lang.sl.SL0Ont...	INFO	java.util.logging.ConsoleHandler	
jade.content.lang.sl.SL1Ont...	INFO	java.util.logging.ConsoleHandler	
jade.content.lang.sl.SL2Ont...	INFO	java.util.logging.ConsoleHandler	
jade.content.lang.sl.SLOntol...	INFO	java.util.logging.ConsoleHandler	
jade.content.onto.BasicOntol...	SEVERE	java.util.logging.ConsoleHandler, java.util.logging.FileHandler	myLog.txt
jade.content.onto.Ontology	INFO	java.util.logging.ConsoleHandler	
jade.content.onto.Serializabl...	INFO	java.util.logging.ConsoleHandler	
jade.content.schema.AgentA...	INFO	java.util.logging.ConsoleHandler	
jade.content.schema.Aggreg...	INFO	java.util.logging.ConsoleHandler	
jade.content.schema.Conce...	INFO	java.util.logging.ConsoleHandler	
jade.content.schema.Conte...	INFO	java.util.logging.ConsoleHandler	
jade.content.schema.Conte...	FINER	java.util.logging.ConsoleHandler	
jade.content.schema.IRESc...	INFO	java.util.logging.ConsoleHandler	
jade.content.schema.Object...	INFO	java.util.logging.ConsoleHandler	
jade.content.schema.Predic...	SEVERE	java.util.logging.ConsoleHandler	
jade.content.schema.Primiti...	WARNING	java.util.logging.ConsoleHandler, java.util.logging.FileHandler	log.txt
jade.content.schema.TermS...	INFO	java.util.logging.ConsoleHandler	
jade.content.schema.Variabl...	CONFIG	java.util.logging.ConsoleHandler	
jade.core.AgentContainerImpl	FINE	java.util.logging.ConsoleHandler	
jade.domain.DFGUIManage...	FINE	java.util.logging.ConsoleHandler	
jade.domain.DFMemKB	FINER	java.util.logging.ConsoleHandler	
jade.domain.FIPAAgentMan...	FINEST	java.util.logging.ConsoleHandler	
jade.domain.FIPAAgentMan...	ALL	java.util.logging.ConsoleHandler	
jade.domain.JADEAgentMan...	INFO	java.util.logging.ConsoleHandler	
jade.domain.ams	INFO	java.util.logging.ConsoleHandler	
jade.domain.df	INFO	java.util.logging.ConsoleHandler	

Directory Facilitator - GUI

- **Navegar**, inscripción, baja, modificación y búsqueda de agentes
- **Federación** con otros DF



Ciclo de vida de un Agente



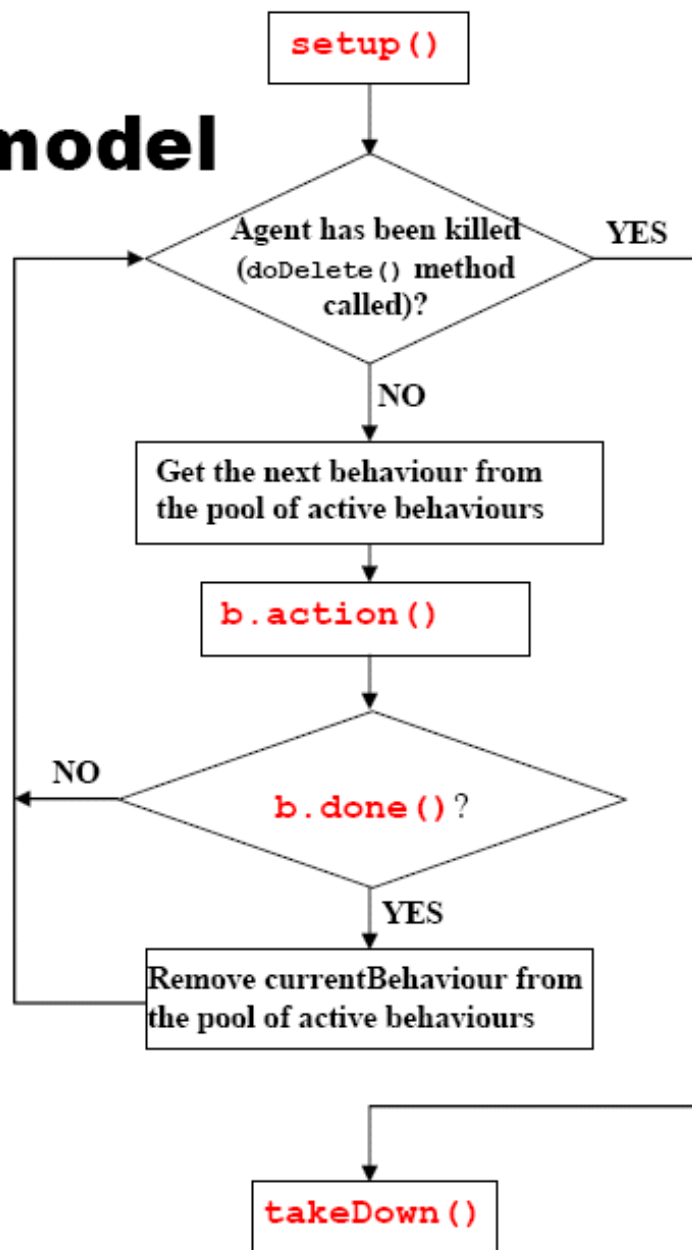
Comportamientos (Behaviours)

- **Funcionalidad** que incorpora el agente
- Son métodos que permiten realizar **acciones** en “**hilos**” de ejecución
- Dan soporte al agente para realizar **varias tareas** y establecer los **tiempos de ejecución** de cada una



The agent execution model

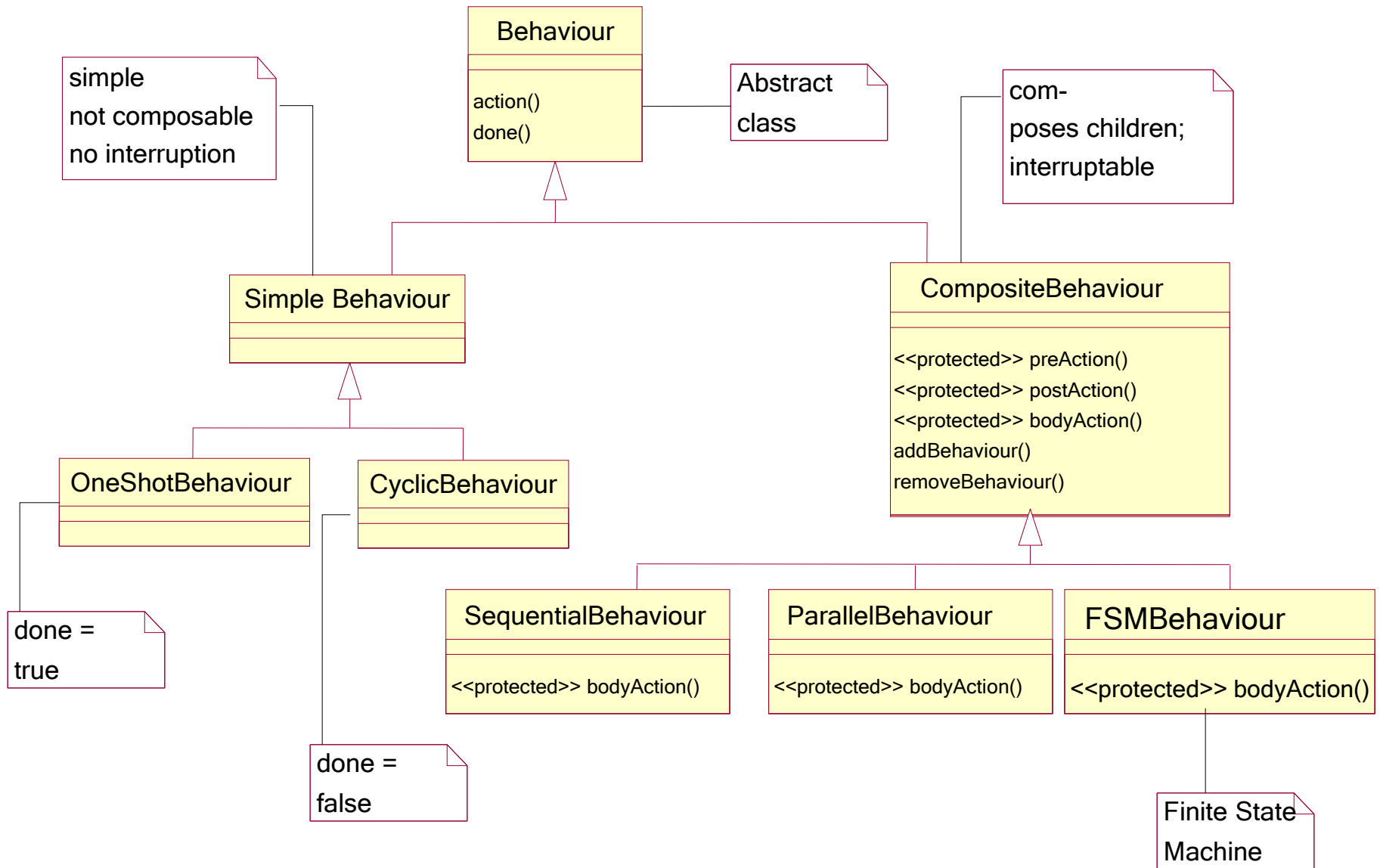
Highlighted in red the methods that programmers have to/can implement



- Initializations
- Addition of initial behaviours

- Agent "life" (execution of behaviours)

- Clean-up operations



¡Programemos nuestro primer Sistema Multi-Agente!



Instalar JADE



Directorio JADE



CLASSPATH

C:\JADE\lib\;

C:\JADE\lib\jade.jar;

C:\JADE\lib\commons-codec\;

C:\JADE\lib\commons-codec\commons-codec-1.3.jar;

Proyecto en Netbeans

✓ **Agents** C:\Users\Octavio\IdeaProjects\Agents

> .idea

agents

✓ libraries

> commons-codec-1.3.jar

> jade.jar

✓ src

✓ mx.itam.packages.agents

Agents

HelloAgent

Importar librerías de JADE

Crear Plataforma y
desplegar agentes

Agente

Classes JADE



```
import jade.core.Agent;
import jade.core.AID;
import jade.core.behaviours.SimpleBehaviour;
import jade.core.behaviours.CyclicBehaviour;
import jade.core.behaviours.OneShotBehaviour;
import jade.core.behaviours.TickerBehaviour;
import jade.core.behaviours.SequentialBehaviour;
import jade.core.behaviours.ParallelBehaviour;
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.domain.FIPAException;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
```

Código mínimo de un Agente

```
package mx.itam.packages.agents;
```

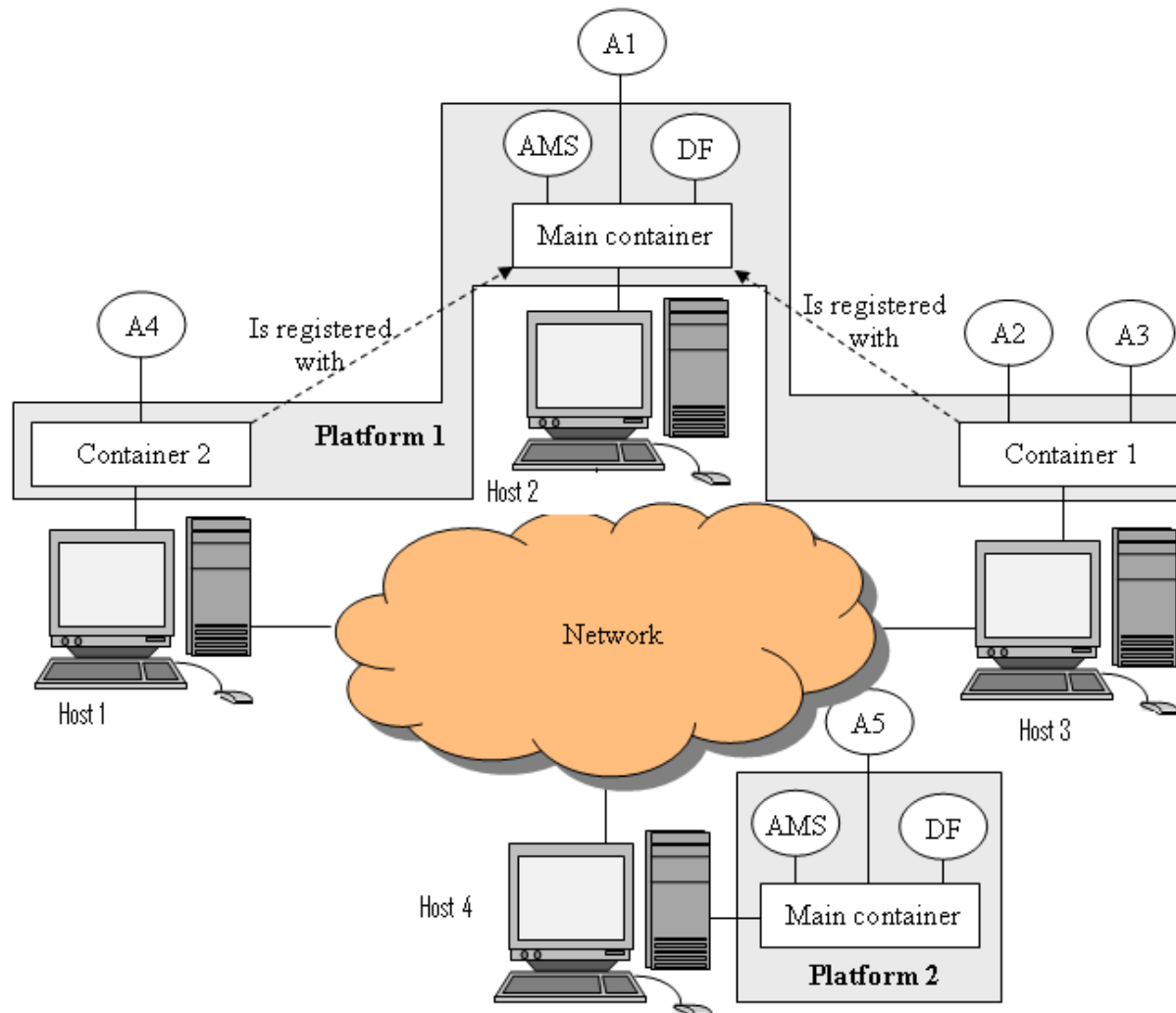
```
import jade.core.Agent;
```

```
public class HelloAgent extends Agent {  
    protected void setup() {  
        System.out.println("Hello World. ");  
        System.out.println("My name local is "+getLocalName());  
        System.out.println("My GUID is "+getAID().getName());  
    }  
}
```



Globally
Unique
Identifier

Arquitectura de plataforma Multi-agente



Crear contenedor principal de agentes

Listener Port

Nombre de la
Plataforma



```
jade.core.Runtime mainRuntime;  
mainRuntime = jade.core.Runtime.instance();
```

```
jade.core.ProfileImpl mainProfile;  
mainProfile = new jade.core.ProfileImpl("localhost", 21300, "AgentPlatform", true);  
mainProfile.setParameter("jade _core_messaging_MessageManager_maxqueuesize", "90000000");
```

```
jade.wrapper.AgentContainer myMainContainer;  
myMainContainer = mainRuntime.createMainContainer(mainProfile);
```


Creo mi primer agente



```
myMainContainer.createNewAgent("MyMainHelloWorldAgent",  
    "mx.itam.packages.agents.HelloAgent", null);
```

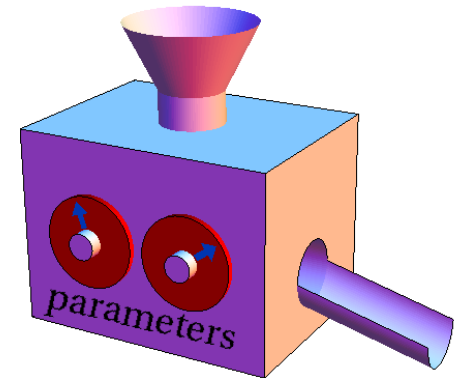
```
myMainContainer.getAgent("MyMainHelloWorldAgent").start();
```



Package -> Clase

Pasando parámetros a mi Agente

```
Object [] agentParams = new Object [6];  
agentParams[0] = "Param 0";  
agentParams[1] = "Param 1";  
agentParams[2] = "Param 2";  
agentParams[3] = "Param 3";  
agentParams[4] = "Param 4";  
agentParams[5] = "Param 5";
```

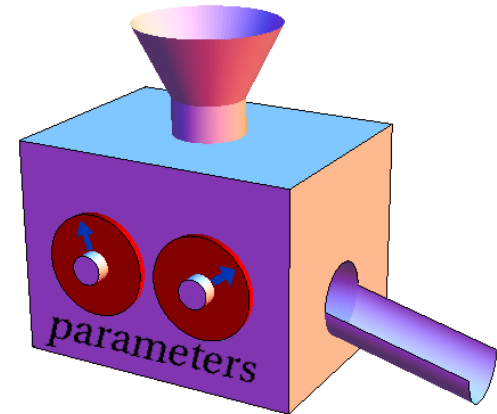


```
myMainContainer.createNewAgent("MyMainHelloWorldAgent",  
                                "mx.itam.packages.agents.HelloAgent",  
                                agentParams);  
myMainContainer.getAgent("MyMainHelloWorldAgent").start();
```

Recibiendo parámetros en mi Agente

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

```
if (args!=null)
    for (int i =0; i<args.length; i++){
        System.out.println("Hi "+ (String)args[i]+" !" );
    }
```



Creo mis primeros agentes (de Sistema)

```
myMainContainer.createNewAgent("sniffer","jade.tools.sniffer.Sniffer", null);  
myMainContainer.getAgent("sniffer").start();
```



```
myMainContainer.createNewAgent("RMA","jade.tools.rma.rma", null);  
myMainContainer.getAgent("RMA").start();
```

Remote Monitoring Agent

Comunicación entre agentes

REQUEST

:sender AgentePedro
:receiver AgenteLuis
:protocol RequestReply
:conversation_id example6
:reply_with 275
:reply_by wed 3pm
:conversation_id conversación6
:ontology Amigos
:content (objetivo
 (reunion "Poker")
 (fecha "8/07/13 4pm")
)
)

Semántica de la Comunicación
Inform, Request, Failure, Refuse,

...

Detalles del mensaje

Coordinación del dialogo

Detalles de la comunicación
*Específico del dominio
(ontologías)*

Enviando mensajes con el Dummy Agent

da0@IBM10312:1099/JADE - DummyAgent

General Current message Queued message

ACLMessage Envelope

Sender: Set

Receivers: da1@IBM10312:1099/JADE

Reply-to:

Communicative act: propose

Content:

Language:

Encoding:

Ontology:

Protocol: Null

Conversation-id:


In-reply-to:

Reply-with: BM10312:1099/JADE1096016308500

Reply-by: Set

User Properties:

24/09/04 10:59: PROPOSE
24/09/04 10:58: QUERY-IF
24/09/04 10:58: FAILURE
24/09/04 10:58: ACCEPT-PROPOSAL
24/09/04 10:58: CANCEL
24/09/04 10:58: ACCEPT-PROPOSAL
24/09/04 10:52: CONFIRM



Verificando mensajes con el Introspector Agent



Introspector0@AgentPlatform

File About

Main-Contain

- Introspec
- Introspec
- MyMainH
- RMA@A
- ams@Ag
- df@Ager

AgentPlatforms.ThisPlatform.M

MyMainHelloWorldAgent@AgentPlatform

View State Debug

Current State

- Active
- Suspended
- Idle
- Waiting
- Moving
- Dead

Change St...

- Suspend
- Wait
- Wake Up
- Kill

Incoming Messages

Pending Received

Incoming Messages -- Pending

Outgoing Messages

Pending Sent

Outgoing Messages -- Pending

Behaviours

- MessageListener
- MessageListener

Name: MessageListener
Class: agents.HelloAgent\$MessageLister
Kind: CyclicBehaviour

Procesando mensajes con Behaviours

```
private class MessageListener extends CyclicBehaviour {  
    private Agent agt;  
    private MessageTemplate mt;  
  
    public MessageListener(Agent agt, String topicID) {  
        this.agt = agt;  
        mt = MessageTemplate.and( MessageTemplate.MatchConversationId(topicID),  
                                   MessageTemplate.MatchPerformative(ACLMessage.INFORM));  
    }  
  
    public void action() {  
        ...  
        ...  
    }  
}
```

Propietario

¿Qué tipo de mensajes
voy a recibir?



Procesando mensajes con Behaviours

```
private class MessageListener extends CyclicBehaviour {  
    ...  
  
    public MessageListener(Agent agt, String topicID) {  
        ...  
    }  
  
    public void action() {  
        ACLMessage msg = agt.receive(mt);  
        if (msg == null) { block(); return; }  
        try {  
            String content = msg.getContent();  
            System.out.println("I received a message from " + msg.getSender()+  
                               " saying: " + msg.getContent());  
        } catch (Exception ex) {  
            ex.printStackTrace();  
        }  
    }  
}
```



Procesando mensajes con Behaviours

```
private class MessageListener extends CyclicBehaviour {
    private Agent agt;
    private MessageTemplate mt;

    public MessageListener(Agent agt, String topicID) {
        this.agt = agt;
        mt = MessageTemplate.and( MessageTemplate.MatchConversationId(topicID),
                                   MessageTemplate.MatchPerformative(ACLMessage.INFORM));
    }

    public void action() {
        ACLMessage msg = agt.receive(mt);
        if (msg == null) { block(); return; }
        try {
            String content = msg.getContent();
            System.out.println("I received a message from " + msg.getSender()+
                               " saying: " + msg.getContent());
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```



Ya tengo mis Behaviours...
ahora ¿cómo las asigno a mis agentes?



En el método *setup()* de los agentes

```
addBehaviour(new MessageListener(this, "TopicA"));
```

```
addBehaviour(new MessageListener(this, "TopicB"));
```

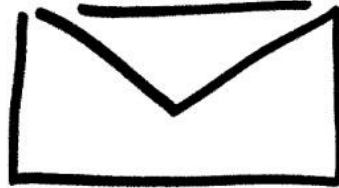
Envío de mensajes

```
private class MessageSender extends OneShotBehaviour {  
    private Agent agt;  
    private String conversationID;  
    private String receiver;  
  
    MessageSender(Agent agt, String conversationID, String receiver) {  
        this.agt = agt;  
        this.conversationID = conversationID;  
        this.receiver = receiver;  
    }  
  
    public void action() {  
        try{  
            ACLMessage msg = new ACLMessage(ACLMessage.INFORM);  
  
            msg.addReceiver(new AID(receiver, AID.ISGUID));  
            msg.setConversationId(conversationID);  
            msg.setContent("Contenido");  
            agt.send(msg);  
        } catch (Exception ex) {  
            ex.printStackTrace();  
        }  
    }  
}
```



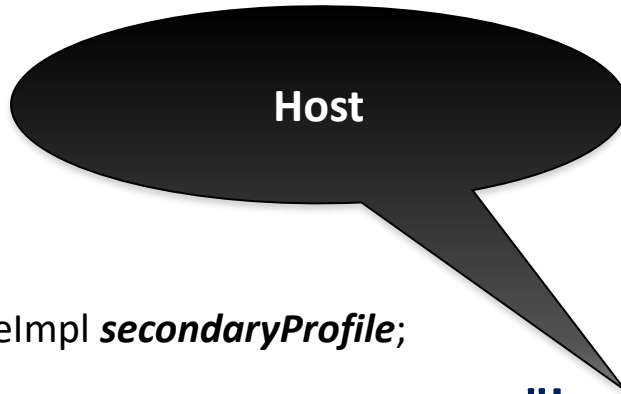
Mensajes ACL (Agent Communication Language)

- **ACCEPT_PROPOSAL**
- **AGREE**
- **CANCEL**
- **CFP**
- **CONFIRM**
- **DISCONFIRM**
- **FAILURE**
- **INFORM**
- **NOT_UNDERSTOOD**



- **PROPOSE**
- **REFUSE**
- **REJECT_PROPOSAL**
- **REQUEST**
- **REQUEST_WHEN**
- **REQUEST_WHENEVER**
- **SUBSCRIBE**
- **PROPAGATE**
- **UNKNOWN**

Crear contenedor secundario de agentes



```
jade.core.ProfileImpl secondaryProfile;
```

```
secondaryProfile = new jade.core.ProfileImpl("localhost", 2001, "AgentPlatform", false);
```

```
secondaryProfile.setParameter("jade _core_messaging_MessageManager_maxqueuesize", "90000000");
```

```
jade.wrapper.AgentContainer mySecondaryContainer;
```

```
mySecondaryContainer = mainRuntime.createAgentContainer(secondaryProfile);
```

*Crear y desplegar otro agente
HelloWorld, pero ahora en un
contenedor secundario*

Agentes interactuando...



En el método setup:

```
addBehaviour(new MessageListener(this, "TopicA"));
```

```
addBehaviour(new MessageSender(this, "TopicA", (String)args[?]))
```



Simple Behaviours

```
private class GenericBehaviour extends SimpleBehaviour {  
    private Agent agt;  
    private int state = 1;  
    private boolean finished = false;  
  
    public GenericBehaviour (Agent agt) {  
        this.agt = agt;  
    }  
  
    public void action() {  
        switch( state ) {  
  
            case 1:    block(2000);    break;  
            case 2:    System.out.println( "--- Message 1 --- " );  
                     block(12000);    break;  
            case 3:    System.out.println( " -- Message 2 --" );  
                     finished = true;  
                     doDelete();    break;  
  
        }  
  
        state++;  
    }  
  
    public boolean done() {  
        return finished;  
    }  
}
```

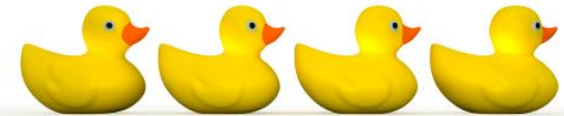
Simple.

Ticker Behaviours

```
private class Clock extends TickerBehaviour {  
    private Agent agt;  
  
    public Clock(Agent agt, long period) {  
        super(agt, period);  
        this.agt = agt;  
    }  
  
    protected void onTick() {  
        System.out.println(agt.getLocalName()+" "+  
                           System.currentTimeMillis());  
    }  
}
```



Sequential Behaviours



SequentialBehaviour **rowOfBehaviours** = new *SequentialBehaviour*(this);

```
rowOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "S"));
rowOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "e"));
rowOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "c"));
rowOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "u"));
rowOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "e"));
rowOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "n"));
rowOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "c"));
rowOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "i"));
rowOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "a"));
```

addBehaviour(rowOfBehaviours);

ImprimeTexto es
OneShotBehaviour
aunque puede ser
una conducta de
cualquier tipo

Parallel Behaviours



```
int termination = ParallelBehaviour.WHEN_ALL;
```

```
ParallelBehaviour setOfBehaviours = new ParallelBehaviour(this, termination);
```

```
setOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "P"));
setOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "a"));
setOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "r"));
setOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "a"));
setOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "l"));
setOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "e"));
setOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "l"));
setOfBehaviours.addSubBehaviour(new ImprimeTexto(this, "o"));
```

```
addBehaviour(setOfBehaviours);
```

Parallel Behaviours



```
int termination = ParallelBehaviour.WHEN_ALL;
```

```
ParallelBehaviour setOfBehaviours = new ParallelBehaviour(this, termination);
```

```
setOfBehaviours.addSubBehaviour(new ImprimeT...
```

WHEN_ALL -> la conducta termina cuando todas sus subconductas terminan.

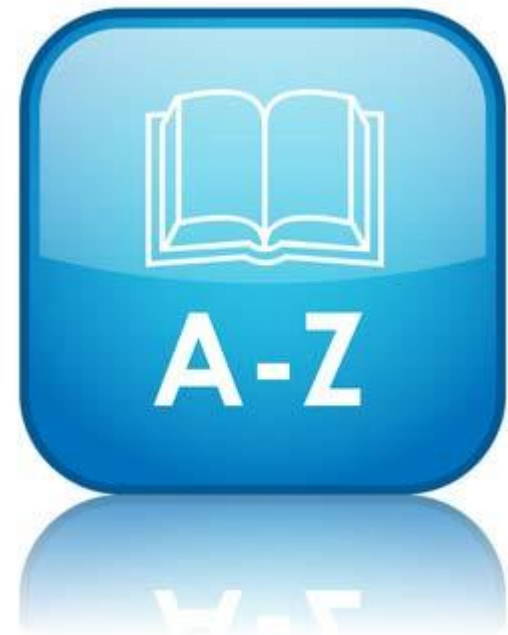
WHEN_ANY -> la conducta termina cuando alguna de sus subconductas termina.

n -> un valor entero positivo **n** indica que la conducta terminará cuando **n** subconductas hayan terminado

```
addBehaviour(setOfBehaviours);
```

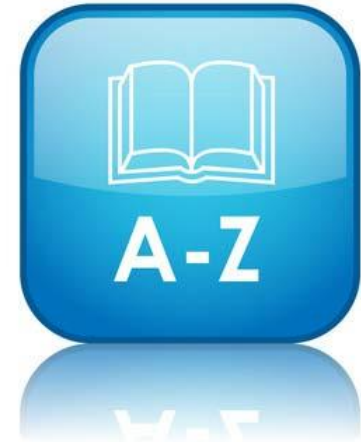
Directory Facilitator – Registro

```
static public void publishService(Agent a, String serviceType){  
    DFAgentDescription dfd = new DFAgentDescription();  
    dfd.setName((jade.core.AID)a.getAID());  
    ServiceDescription sd = new ServiceDescription();  
    sd.setType(serviceType);  
    sd.setName(a.getAID().getName());  
    dfd.addServices(sd);  
    try {  
        DFService.register(a, dfd);  
    }  
    catch (FIPAException fe) {  
        fe.printStackTrace();  
    }  
}
```



Directory Facilitator – Baja de registro

```
static public void unPublishService(Agent a){  
    try {  
        DFService.deregister(a);  
    }  
    catch (FIPAException e) {  
        e.printStackTrace();  
    }  
}
```



```
protected void takeDown() {  
    unPublishService(this);  
}
```

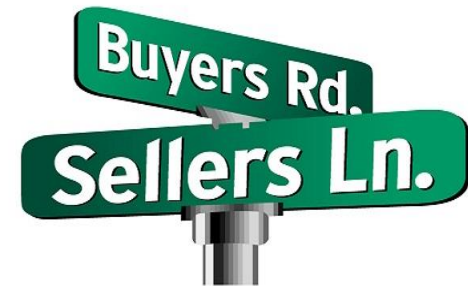
Cuando el agente se va, debe dar de baja su servicio

Directory Facilitator – Búsqueda

```
static public String searchAgent(Agent agt, String serviceType) {  
    try{  
        DFAgentDescription dfd = new DFAgentDescription();  
        ServiceDescription sd = new ServiceDescription();  
        sd.setType(serviceType);  
        dfd.addServices(sd);  
        DFAgentDescription[] result = DFService.search(agt, dfd);  
        if (result.length>0)  
            return result[0].getName().getName();  
    } catch (Exception ex){  
        ex.printStackTrace();  
    }  
    return "";  
}
```



Programando Agentes



```
public class BuyerAgent extends Agent {  
    protected void setup() {...}  
  
    private class Asking extends TickerBehaviour {...}  
  
    static public String searchAgent(Agent agt, String serviceType) {...}  
}
```



```
public class SellerAgent extends Agent {  
    protected void setup() {...}  
  
    protected void takeDown () {...}  
  
    static public void unPublishService(Agent a) {...}  
  
    static public void publishService(Agent a, String serviceType) {...}  
  
    private class MessageListenerAndReplier extends CyclicBehaviour {...}  
}
```

Material adicional sobre el modelo Actores y Agentes

- C. Hewitt et al., **Laws for Communicating Parallel Processes**, IFIP 77, pp. 987-992, N-H 1977
- C. Hewitt, and H. Baker, **Actors And Continuous Functionals**, MIT-LCS-TR-194, 1978.
- Agha, G.A. **Actors: a model of concurrent computation in distributed systems**, The MIT Press, Cambridge, Massachusetts, 1985.
- Wooldridge M., **An introduction to multiagent systems**, second ed, John Wiley & Sons Ltd, Chichester, England, 2009.
- **Jade – Java Agent Development Framework**, *jade.tilab.com*
- **Jade Tutorial and Primer** -
<http://www.iro.umontreal.ca/~vaucher/Agents/Jade/JadePrimer.html>