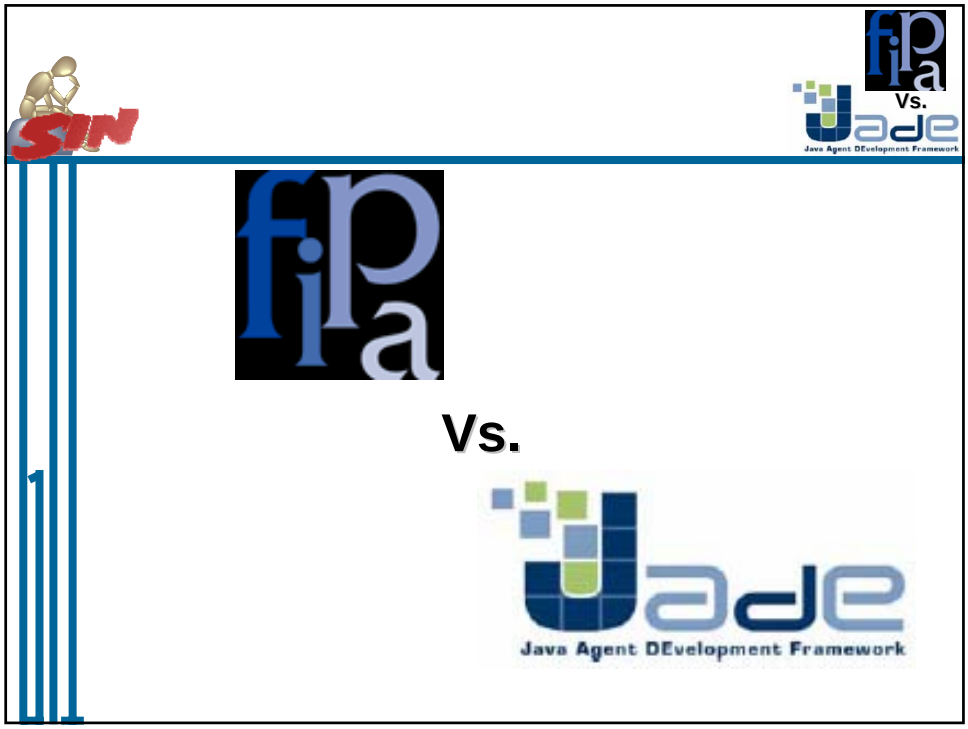


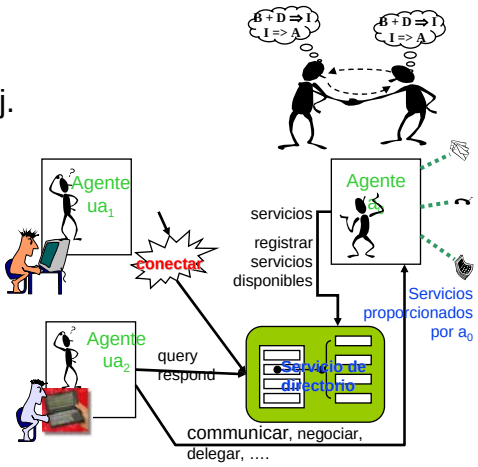
Introducción

- Sin estándares, no hay agentes
  - FIPA (*F*oundation for *I*ntelligent *P*hysical *A*gents)
- Sin herramientas sw, no hay agentes
  - JADE
- Estándares + Herramientas Sw = agentes?

Introducción

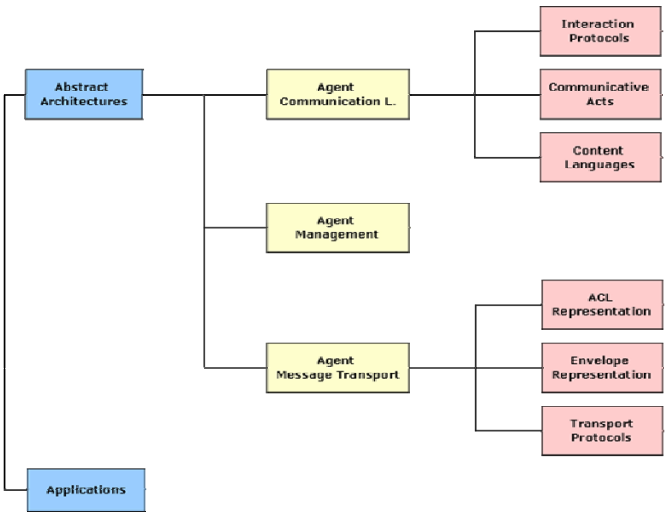
Sin estándares, no hay agentes

- Potencial máximo expresado por un alto nivel de interacción (ej. contract net, subasta)
  - agentes de diversos diseñadores, vendedores, organizaciones
- Estándar: factor que posibilita heterogeneidad y claridad



Pasar de hacer mejores componentes a mejorar la forma en la que los componentes cooperan

Especificaciones



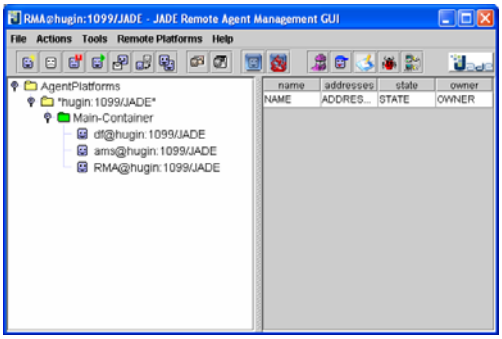
Plataformas de Desarrollo de Agentes

- **FIPA-compliant**: April Agent Platform (Fujitsu Labs of America), FIPA-OS (Emorphia), Grasshopper (IKV++), JADE (CSELT), Zeus (British Telecom).
- **FIPA-OS** cumple con la mayoría de las especificaciones de FIPA – completamente implementada en Java y presenta dos distribuciones (http://www.emorphia.com, formerly by Nortel Networks):
  - Standard FIPA-OS – una distribución se ejecuta sobre JDK1.2 y otra sobre JDK1.1 y
  - MicroFIPA-OS que es una extensión a la versión de JDK 1.1, diseñada principalmente para ejecutar agentes en PDAs.
- **JADE** – Java Development Toolkit – (http://sharon.cselt.it/projects/jade/) – requiere JDK LGPL (Lesser General Public License Version 2).
  - LEAP (Lightweight Extensible Agent Platform) que permite la ejecución de FIPA-compliant agentes en PDAs y teléfonos móviles (http://leap.crm-paris.com/).

Introducción

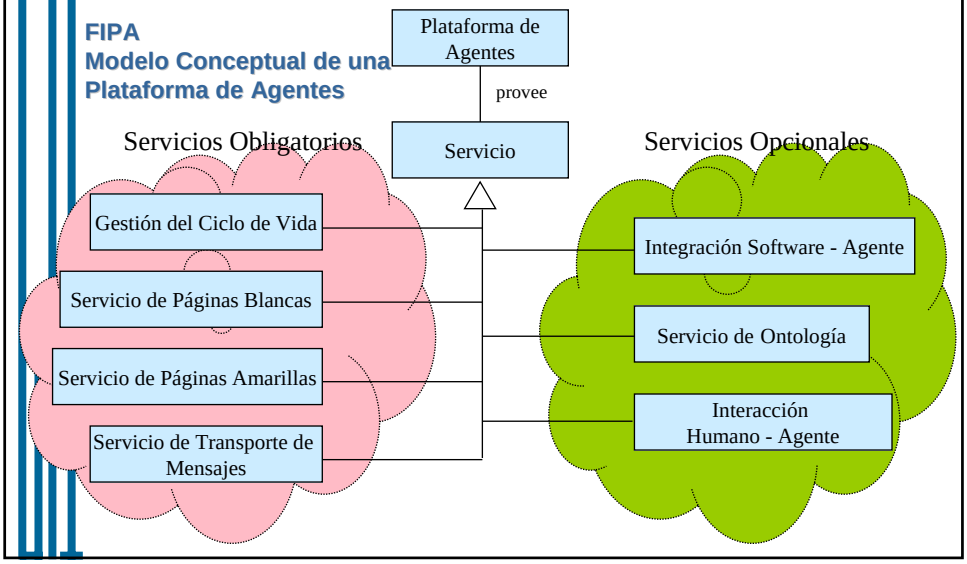
Java Agent DEvelopment Framework

- Combinación de dos productos:
  - Una Plataforma de Agentes FIPA-Compliant
  - Una herramienta de desarrollo de agentes Java

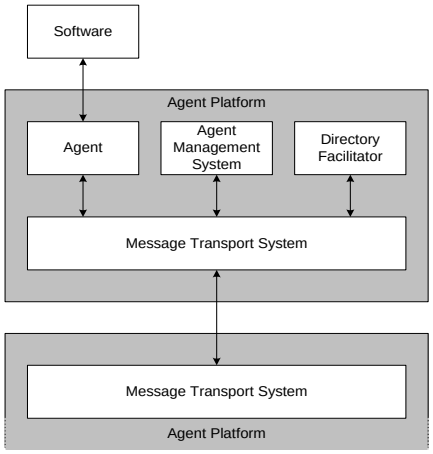


Introducción

FIPA
Modelo Conceptual de una Plataforma de Agentes



Modelo de Gestión del Agente



- El ACC soporta el Sistema de Trasnporte de Mensajes.
- El Servicio de Transporte de Mensajes es el método de comunicación por defecto entre agentes en diferentes APs.
- El ACC también soporta tareas de enrutado de AP.

Hacer corresponder la teoría con el diseño

- *Agentes autónomos.*
- *Agentes como entidades sociales.*
- *Mensajes como "speech acts", no invocaciones.*
- *Un agente puede decir "no" y "no me importa".*

- **Agentes como objetos activos.**
- **Se necesita una concurrencia Intra-agent.**
- **Se deben usar mensajes asíncronos.**
- **Mensajería entre iguales (construida sobre interacciones Cliente/Servidor entre objetos distribuidos).**

Introducción

Plataforma de Agentes FIPA-compliant

- **Una plataforma de agentes que implementa los servicios e infraestructura básicos de una aplicación multi-agente:**
  - **ciclo de vida del agente y movilidad del agente**
  - **servicios de páginas blancas y amarillas**
  - **transporte & parseado de mensajes *peer-to-peer*; así como comunicación *multi-party***
  - **seguridad**
  - **Planificación de múltiples tareas de agentes**
  - **conjunto de herramientas gráficas para dar soporte a la monitorización, *logging*, y depuración**
  - **fomenta el desarrollo de aplicaciones pro-activas**

Introducción

Plataforma de Agentes FIPA-compliant

- **Algunas características relevantes:**
  - **posibilita interoperabilidad *FIPA compliant*: Comunicación entre agentes:**
    - En la misma y/o diferentes plataforma
    - Biblioteca de protocolos de interacción
    - Registro automático de los agentes con el AMS
    - Servicio de nombrado
    - IIOP para conectar las diferentes plataformas
  - **proyecto open source originado por TILAB, y gestionado actualmente por un comité internacional**
  - **usado por diversos proyectos de I+D**

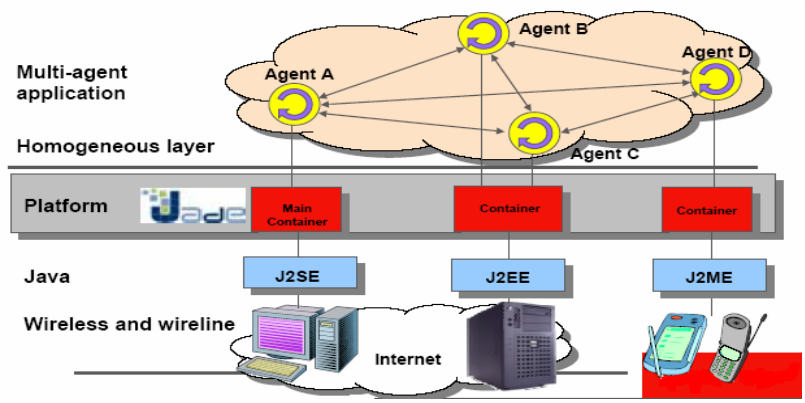
Arquitectura

General

- "Plataforma" JADE
- Arquitectura distribuida
  - Multiples hosts
  - Multiples procesos
  - Varios "contenedores de agentes" (cada agente vive en un contenedor)
- Contenedores de agentes
  - Es una JVM
  - Entorno de ejecución de agentes.
  - Un proceso java = un contenedor
  - 1 agente = 1 hilo de ejecución
  - Controla el ciclo de vida de los agentes (crear, suspender, reanudar, matar)
  - Gestiona la comunicación
  - Transparentes a agentes
  - El contenedor principal mantiene los servicios de la plataforma
  - Linked conjuntamente usando java RMI

Arquitectura

Modelo General



Arquitectura

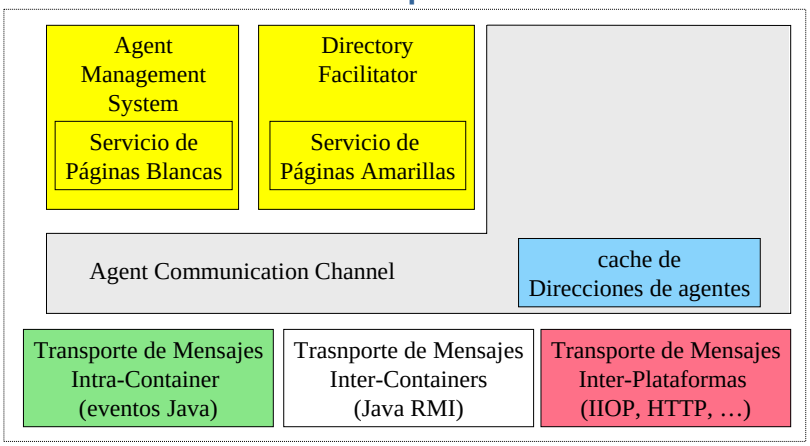
Contenedor Principal

- Servicios de la Plataforma: Implementados como agentes:
  - **AMS**: Servicio de Gestión de agentes
    - Páginas blancas
    - Mantiene al conjunto de agentes de la plataforma
    - Un agente está en la plataforma si está registrado en el AMS
  - **DF**: Facilitador de directorio
    - Paginas amarillas
    - Proporciona un directorio de servicios
    - Mapea descripciones de servicios a identificadores de agente
    - Los agentes pueden añadir, modificar y borrar entradas por ellos mismos
  - ***Sniffer***
  - **RMA**: Agente Gestor Remoto
    - Es el propio GUI

Arquitectura

Contenedor Principal

Arquitectura Interna



Nota: La arquitectura interna de un contenedor JADE es parecida, pero no contiene ni el AMS, ni el DF.

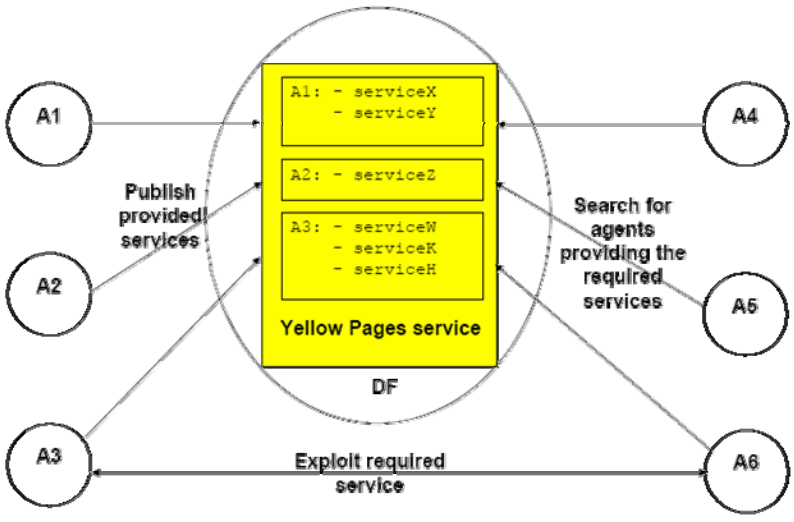
Gestión de Agentes

- **AMS** y **DF** tienen un subconjunto común de acciones:
  - register
  - deregister
  - search
  - modify

Agent Management System

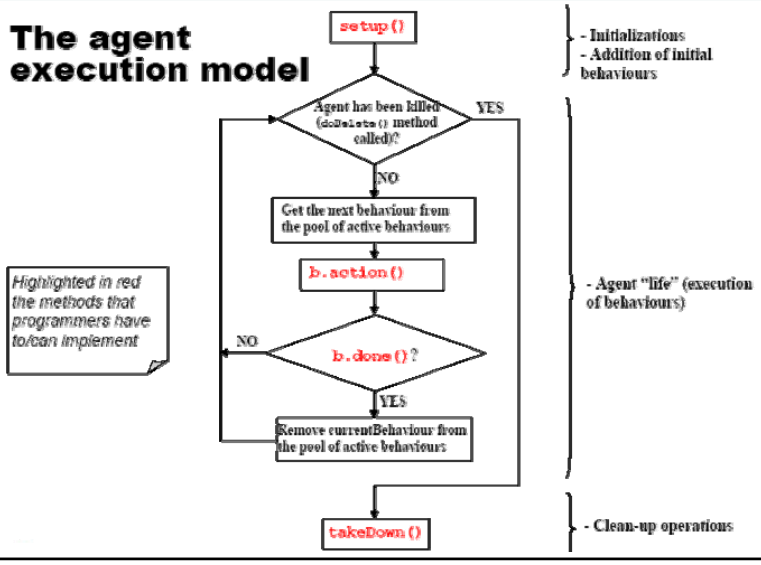
- La **autoridad** en una plataforma JADE; encargada de todas las acciones de gestión de la plataforma (crear/eliminar agentes, eliminar containedores, ...)
- Otros agentes pueden solicitar al AMS realizar estas acciones usando
  - El protocolo de interacción **fipa-request**
  - El lenguaje **SL**
  - La ontología **JADE-Management** y las acciones relacionadas
- **getAMS()** ➔ el AID del AMS

Directory Facilitator



Implementación

Control de Flujo



FIPA ACL

vs.

Elementos del Mensaje

- **performative**: Qué acción realiza el mensaje
- **sender**: Iniciador del mensaje
- **receiver**: Receptor del mensaje
- **reply-to**: Recipiente del mensaje ***reply***
- **content**: Contenido del mensaje
- **language**: Lenguaje usado para expresar el contenido
- **encoding**: Codificación usada por el contenido
- **ontology**: Ontología usada para el contenido
- **protocol**: Protocolo al que pertenece el mensaje
- **conversation-id**: Conversación a la que pertenece el mensaje
- **reply-with**: Responder con esta expresión
- **in-reply-to**: Acción a la cual se esta respondiendo con este mensaje
- **reply-by**: Tiempo para recibir el ***reply by***

FIPA ACL

Biblioteca de Actos Comunicativos

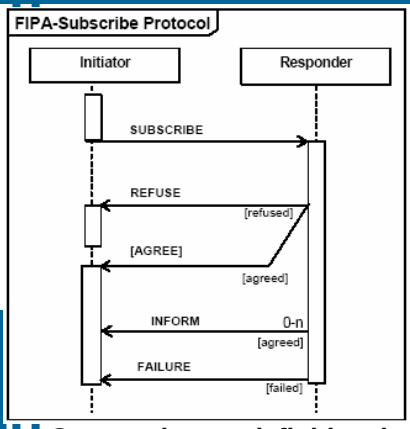
- **accept-proposal**: aceptar una propuesta recibida previamente
- **agree**: estar de acuerdo en realizar alguna acción
- **cancel**: cancelar alguna acción pedida previamente
- **cfp**: solicitar propuestas para realizar una acción dada
- **confirm**: informar a un receptor que una proposición es cierta
- **disconfirm**: informar a un receptor que una proposición es falsa
- **failure**: informar a otro agente que se intentó una acción pero falló
- **inform**: informar a un receptor que una proposición es cierta
- **not-understood**: informar a un receptor que el emisor no entendió el mensaje
- **query-if**: preguntarle a otro agente si una determinada proposición es cierta
- **request**: solicitar a un receptor que realice alguna acción

FIPA ACL

Biblioteca de Actos Comunicativos

- **propose:** enviar una propuesta para realizar una cierta acción
- **query-ref**: preguntar a otro agente por el objeto referenciado en una expresión
- **refuse**: rechazar realizar una acción
- **reject-proposal**: rechazar una propuesta durante una negociación
- **request-when**: solicitar al receptor que realice alguna acción cuando una proposición dada sea cierta
- **request-whenever**: solicitar al receptor que realice alguna acción cada vez que una proposición dada sea cierta
- **subscribe**: una intención persistente de notificar al emisor de un determinado valor, y volver a notificarle cada vez que dicho valor cambie
- **propagate**: el receptor trata el mensaje como si fuese dirigido directamente a él, y debe identificar los agentes en este descriptor y enviarles el mensaje a ellos
- **proxy**: el receptor debe seleccionar agentes objetivo denotados por una descripción dada, y enviarles un mensaje embebido

Interaction protocols



Secuencias predefinidas de mensajes intercambiados por agentes durante una conversación

- **Jade automáticamente gestiona el flujo de mensajes y los timeouts**
- **callback methods que deberían ser redefinidos para tomar las acciones necesarias**
- **jade.proto contiene comportam. para los roles de iniciador y respondedor:** FIPA-Request, FIPA-Contract-Net, FIPA-Subscribe
- **Otros protocolos de interacción de FIPA:** FIPA-Query, FIPA-Request-When, FIPA-AIterated-Contract-Net, FIPA-Auction-English, FIPA-Auction-Dutch, FIPA-Brokering, FIPA-Recruiting, FIPA-Propose

Protocolos y Codificación de Mensajes

- Protocolo de Transporte de Mensajes:
  - Hay 3 posibles protocolos:
    - IIOP
    - WAP
    - HTTP
- Codificación de Mensajes:
  - Hay 3 posibilidades de codificación:
    - String encoding
    - bit-efficient encoding
    - XML
- Cada uno de los protocolos de transporte soporta todos los tipos de codificación de mensajes.
- Sólo el sobre del mensaje depende del protocolo de transporte.

Ejemplo de Mensaje ACL

```
(request
:sender (:name dominicagent@whitestein.com:8080)
:receiver (:name rexhotel@tcp://hotelrex.com:6600)
:ontology personal-travel-assistant
:language FIPA-SL
:protocol fipa-request
:content
    (action movenpickhotel@tcp://movenpick.com:6600
    (book-hotel (:arrival 25/11/2000) (:departure 05/12/2000) ...
    )))
```

- Any language can be used as a Content Language, e.g.:
  - KIF, Prolog, SQL, Serialized Objects, Binary Large Objects
  - FIPA-SL, FIPA-CCL, FIPA-RDF, FIPA-KIF

Implementación

Comunicación

- Estructura de un mensaje:
  - Sobre
    - Construido, implementado por la plataforma
    - Enrutamiento: indica remitente (sender) y destinatarios del mensaje y de donde procede el mensaje (como email)
  - Mensaje ACL
    - Desempeño (REQUEST, INFORM)
    - Conjunto de atributos estándar
      - Lenguaje del cuerpo del mensaje
      - ID de la conversación
      - Ontología del mensaje
    - Cuerpo
      - Interpretado por los agentes
      - Conforma a algún lenguaje de contenido y ontología
- Interpretación de mensajes:
  - Semántica ACL definida sólo sobre el cuerpo del mensaje ACL
  - Toda la información en el sobre es de soporte (FIPA no indica cómo ni si se usa por el agente para alguna inferencia).

Implementación

Comunicación

Vs.

- Contenido del Mensaje
  - El contenido del mensaje está en concordancia con el lenguaje de contenido (CL) y la ontología
  - Semántica de alto nivel del CL: SL
    - Diferentes niveles de complejidad
      - SL0: Objetos, acciones, resultados
      - SL1: predicados de operadores lógicos básicos, AND/OR/NOT
      - SL2: modalidades de alto nivel: creencias, intenciones objetivos
  - Objetos en el lenguaje son descritos por la ontología

Implementación

Comunicación

Vs.

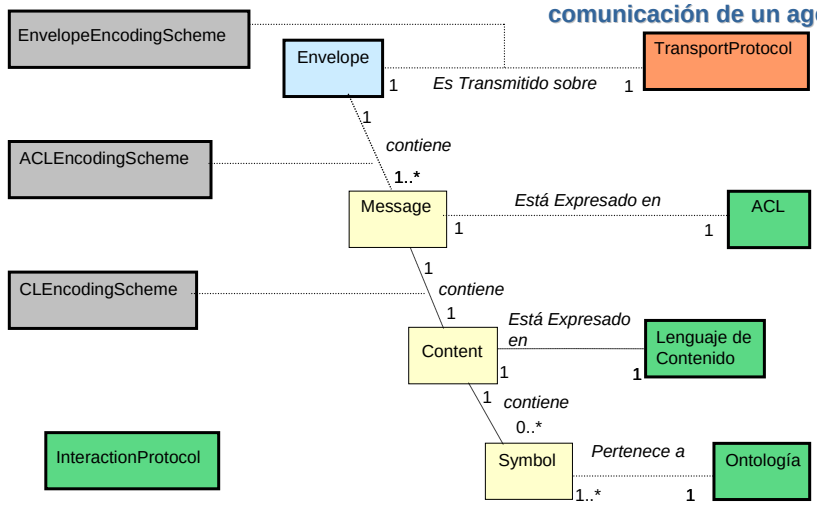
- Ontologías
  - Una ontología proporciona descripciones de conceptos en el mundo y relaciones entre ellos
  - Similar al análisis orientado a objetos para datos
  - Correspondencia entre objetos descritos por ontología y objetos java

Implementación

Comunicación

Vs.

Modelo Conceptual FIPA de la comunicación de un agente



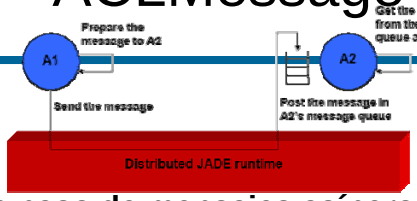
Implementación

Comunicación

Vs.

```
(REQUEST
:sender (agent-identifier :name da0)
:receiver (set (agent-identifier :name df))
:content "((action (agent-identifier :name df)
            (register (df-agent-description
                        :name (agent-identifier :name da0)
                        :services (set (service-description
                                        :name sub-sub-df :type fipa-df
                                        :ontologies (set fipa-agent-management)
                                        :languages (set FIPA-SL)
                                        :protocols (set fipa-request)
                                        :ownership JADE ))
                                    :protocols (set) :ontologies (set) :languages (set)
            )) ) )"
:reply-with rwsub1234 :languaje FIPA-SL0
:ontology FIPA-Agent-Management : protocol fipa-request
:conversation-id convsub1234
)
```

ACLMessage



- **Basado en paso de mensajes asíncrono.**
- **Formato de mensaje definido por el ACL (FIPA)**
- **Mensajes intercambiados por agentes son instancias de jade.lang.acl.ACLMessage**
- **Enviar un mensaje = crear un objeto ACLMessage y llamar al método send() de la clase Agent**
- **Se consigue leer mensajes de la cola privada de mensajes por medio del método receive()**
- **get/setPerformative(), get/setSender(), add/getAllReceiver(), get/setLanguage(), get/setOntology(), get/setContent()**

Implementación

Comunicación

- Construir un mensaje
  - Crear un objeto mensaje (ACL Mensaje)
  - Poner desempeño

```
ACLMessage msg = new ACLMessage( ACLMessage.INFORM );
```

  - Poner remitente (automático) y receptores

```
AID dest = ...;
msg.addReceiver( dest );
```

  - Poner atributos mensaje (get/set)
  - Rellenar contenido

```
msg.setContent("I sell seashells at $10/kg" );
```

- Enviar mensaje

```
send(msg);
```

Implementación

Comunicación

Implicaciones en la implementación

- Biblioteca de protocolos de interacción
- Estructura soporta el parseado
  - *envelope parser*
  - ACL *parser*
  - CL *parser*
    - Incluye soporte para serialización de Java uuencoded
  - Ontology checker
- Estructura extendida por el usuario
  - Permite definir/salvar/cargar nuevas ontologías
  - Interfaz para CLParser/Encoder
  - Usado automáticamente por la estructura

Implementación

Comunicación

- Recibir un mensaje: Dos métodos
  - Suspender actividad del **agente** hasta que llega un mensaje

```
ACLMessage msg = blockingReceive();
```

  - Consultar la cola de mensajes (diversos comp. activos)

```
ACLMessage msg = receive();
```

O bien, pueden tratar de recibir un tipo concreto de mensaje, según un filtro

```
MessageTemplate mt =
   MessageTemplate.MatchPerformative(ACLMessage.CFP);
ACLMessage msg = myAgent.receive(mt); // igual con blockingReceive();

if (msg != null)
   <.... handle message...>
else <... do something else like block() ...>
```

Ejemplo de Recepción de Mensajes

- Imprime copias de todos los mensajes que recibe:

```
public class Receiver extends Agent {
    protected void setup() {
        addBehaviour(new CyclicBehaviour(this) {
            public void action() {
                ACLMessage msg= receive();
                if (msg!=null)
                    System.out.println( " - " +
                        myAgent.getLocalName() + " <- " +
                        msg.getContent() );
                block(); /// Interrumpe el comp. hasta que llegue un mensaje
            }
        });
    }
}
```

Respondiendo Mensajes

```
ACLMessage msg = receive();
ACLMessage reply = new ACLMessage( ACLMessage.INFORM );
reply.setContent( "Pong" );
reply.addReceiver( msg.getSender() );
send(reply);
```

O alternativamente, con createReply();

```
public void action() {
    ACLMessage msg = receive();
    if (msg!=null) {
        System.out.println(" - " + myAgent.getLocalName() + " <- " + msg.getContent() );
        ACLMessage reply = msg.createReply();
        reply.setPerformative( ACLMessage.INFORM );
        reply.setContent(" Pong" );
        reply.send();
    }
    block();

}
```

Buscando en el AMS

- Conseguir un vector con todos los agentes conocidos:

```
// Required imports
import jade.domain.AMSService;
import jade.domain.FIPAAgentManagement.*;
...
AMSAgentDescription [] agents = null;
try {
    SearchConstraints c = new SearchConstraints();
    c.setMaxResults ( new Long(-1) ); /// Todos
    agents = AMSService.search( this, new AMSAgentDescription (), c );
        /// agente que busca, Descripción de agente (filtro), restricciones
}
catch (Exception e) { ... }
```

AMSAgentDescription no es AID:

```
AID myID = getAID();
for (int i=0; i < agents.length; i++) {
    AID agentID = agents[i].getName();
    System.out.println(
        ( agentID.equals( myID ) ? "*** " : " " ) + i + ": " + agentID.getName() );
}
```

Filtros para Recepción Selectiva

- Distintos comportamientos para gestionar distintos mensajes o de distintos agentes,…
- *MessageTemplate* class:
  - Permite definir filtros para cada atributo de ACLMessage
  - Posibilidades:
    - **MatchPerformative**(*performative*) donde *performative* puede ser:
      - ACLMessage.INFORM
      - ACLMessage.PROPOSE
      - ACLMessage.AGREE
      - ...etc
    - **MatchSender**(AID)
    - **MatchConversationID**(String): Permite dedicar un comportamiento a gestionar una conversación/negociación.
    - **and**(Template1, Template2)
    - **or** (Template1, Template2)
    - **not**(Template)
    - MatchOntology(String)
    - MatchProtocol(String)
    - MatchOntology(String)

Ejemplo: Filtro para recibir mensajes INFORM del agente "a1":

```
MessageTemplate mt = MessageTemplate.and(
        MessageTemplate.MatchPerformative( ACLMessage.INFORM ),
        MessageTemplate.MatchSender( new AID( "a1", AID.ISLOCALNAME))) ;
. . .
ACLMessage msg = receive( mt );
if (msg != null) {
        ... handle message
}
block();
```

DF: Registration

```
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.*;
import jade.domain.FIPAException;
....
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName( getAID() );
try {
        DFService.register( this, dfd );
}
catch (FIPAException fe) {
        fe.printStackTrace();
}
```

Registrar como *buyer*:

```
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName( getAID() );
ServiceDescription sd = new ServiceDescription();
sd.setType( "buyer" );
sd.setName( getLocalName() );
dfd.addServices(sd);
try {
        DFService.register(this, dfd );
}
catch (FIPAException fe) {
        fe.printStackTrace();
}
```

DF: Desregistrar

Vs.

- Un agente sólo puede aparecer en 1 entrada del DF.
- Cuando un agente muere, se elimina automáticam. su entrada del AMS, pero no del DF.

```
protected void takeDown() {  /// Invocado automáticam. cuando el agente muere
    try { DFService.deregister(this); }
    catch (Exception e) {}
}
```

DF: Buscando

- Ejm.: Encontrar un ***buyer***:

```
DFAgentDescription dfd = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
SearchConstraints ALL = new SearchConstraints();
ALL.setMaxResults(new Long(-1));
sd.setType( "buyer" );
dfd.addServices(sd);
DFAgentDescription[] result = DFService.search(this, dfd, ALL);
    /// Devuelve un agente, excepto si tiene como tercer argumento
    ///         el máx. número de respuestas (-1 indica todos).
System.out.println(result.length + " results" );
if (result.length>0)
    System.out.println(" " + result[0].getName() );
```

DF: Suscripción

- Ejm.: Hacer que cuando se registre un agente ofertando un determinado tipo de servicio, el DF avise (con un mensaje de tipo INFORM):

```
DFAgentDescription dfd = new DFAgentDescription();
ServiceDescription sd = new ServiceDescription();
sd.setType(....);
dfd.addServices(sd);
SearchConstraints sc = new SearchConstraints();
sc.setMaxResults(new Long(1));
send(DFService.createSubscriptionMessage(this, getDefaultDF(), dfd, sc));
```

Una forma más compacta de hacer esto:

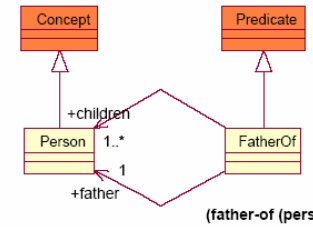
```
DFAgentDescription template = // fill the template
addBehaviour( new SubscriptionInitiator( this,
        DFService.createSubscriptionMessage( this, getDefaultDF(), template, null)) {
            protected void handleInform(ACLMessage inform) {
                try {
                    DFAgentDescription[] dfds =
                        DFService.decodeNotification(inform.getContent());
                            // do something with dfds
                }
                catch (FIPAException fe) {
                    fe.printStackTrace();
                }
            }
        }
);
```

Implementación

Ayudas al programador

- ¿Qué partes de FIPA oculta JADE al programador?
  - no hay que implementar la Plataforma
    - AMS, DF y ACC se lanzan automáticam. al inicio
  - no hay que implementar ni ontología ni funcionalidades relativas a la gestión de agentes
    - El propio constructor de Java registra el agente en la AP
      - Se le da un nombre y una dirección
    - La clase `Agent` aporta una interfaz simplificada a los servicios del **DF** (registro, búsqueda, …)
  - no es necesario implementar el Transporte de Mensajes y el Parseado
    - automaticamente (y posiblemente eficientemente) hecho por la plataforma al enviar/recibir mensajes
  - no es necesario implementar Protocolos de Interacción
    - se deben extender sólo por medio de métodos

Manejando Expresiones de Contenido

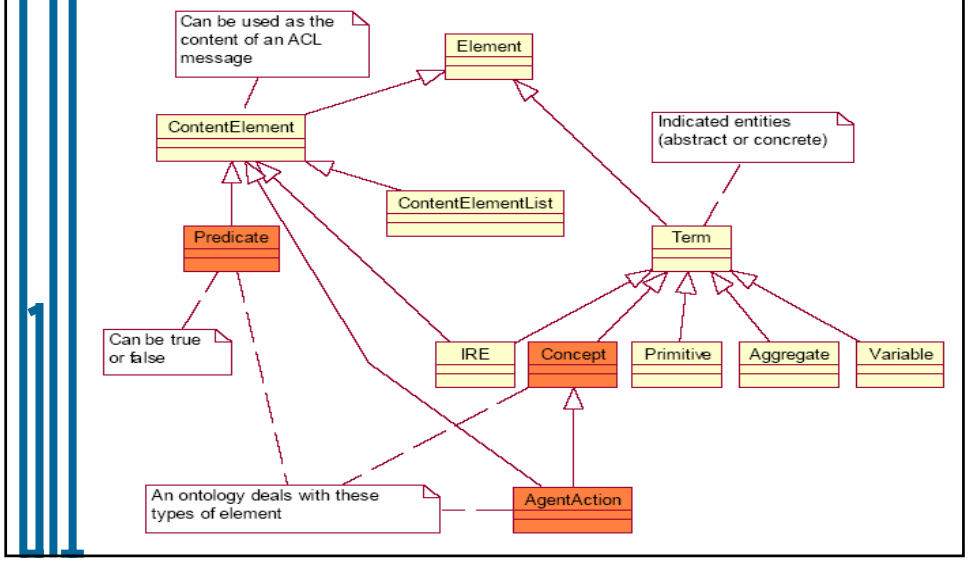



```
Person john = new Person("John", 35);
Person bill = new Person("Bill", 67);
FatherOf f = new FatherOf();
f.setFather(bill);
f.addChildren(john);
```

```
(father-of (person :name Bill :age 67) (set (person :name John :age 35) ) )
```

- Creating the Ontology (domain specific)
  - Defining the schemas ontology elements
  - Defining the corresponding Java classes
- Handling content expressions as Java objects
- Using the ContentManager to fill and parse message contents
- Ontologies can be created with Protégé (with the beangenerator plugin)

Modelo de Referencia del Contenido



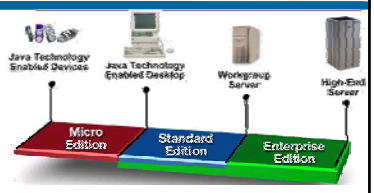
jade.core	This package contains the *microkernel* of **JADE** system.
jade.core.behaviours	This package is a subpackage of `jade.core` and contains the classes used to implement basic agent behaviours.
jade.domain	This package and its sub-packages contains ***FIPA*** specific agents and ontologies.
jade.domain.FIPAAgentManagement	This package contains the definition of the FIPA-Agent-Management ontology as specified by the FIPA standard FIPA Agent Management Specification - document no.
jade.domain.JADEAgentManagement	This package contains the definition of the JADE-Agent-Management ontology, the vocabulary with the list of used symbols, and all the Java classes that implement the concepts of the ontology.
jade.domain.mobility	This package contains the definition of the JADE-mobility ontology, the vocabulary with the list of used symbols, and all the Java classes that implement the concepts of the ontology.
jade.domain.persistence	
jade.lang.acl	This package contains the support for the FIPA Agent Communication Language (ACL) including the ACLMessage class, the parser, the encoder, and an helper class for representing templates of ACL messages.
jade.proto	This package contains role behaviours for ***FIPA*** standard protocols.
jade.proto.states	This package contains classes for common states of an interaction protocol, such as "waiting for a given message", "selecting between a number of alternatives", ...
jade.util	This package contains utility classes and in particular: classes for handling properties, the Logger class for logging capabilities; the leap subpackage, that is a replacement for the Java collection framework that is not supported by J2ME.
jade.util.leap	This package contains a set of classes that provides a replacement for the Java collection framework that is not supported by J2ME.
jade.wrapper	Together with the classes jade.core.Profile and jade.core.Runtime this package provides support for the JADE in-process interface that allows external Java applications to use JADE as a kind of library and to launch the JADE Runtime from within the application itself.

Características Avanzadas

- Usar JADE (por ejm. Crear un contenedor y crear agentes) desde un programa Java externo (servlets, applets, JSP..)
- Ejecutar un comportam. JADE normal en un *thread* dedicado
- Salvar y recargar el estado del agente un una BD relacional (persistencia basado en www.hibernate.org); add-on
- Integración con JESS (Java Expert System Shell)
  - Permite razonar sobre mensajes en JESS
  - Permite que un programa JESS controle el envío/recepción de mensajes y/o crear/destruir comportamientos JADE
- Seguridad distribuida, tolerancia a fallos, soporte para agentes replicados y servicios
- Protégé, XML, RDF y OWL

LEAP

- **Agentes JADE para PDA y teléfonos móviles**
- **JADE-LEAP (J2SE, PJava, MIDP): implementación interna diferente, la misma API para los agentes**
- **JADE-LEAP para MIDP es en sí mismo un MIDlet; la clase principal es jade.MicroBoot**
- **Un MIDlet debe ser empaquetado como un único fichero JAR ➔ mezclas clases de biblioteca y app.**
- **Los parámetros de configuración MIDP se especifican en el manifiesto/JAD del MIDlet**



Links

http://jade.cselt.it

http://www.fipa.org

http://www.hibernate.org

http://herzberg.ca.sandia.gov/jess/

http://protege.stanford.edu/

www.swi.psy.uva.nl/usr/aart/beangenerator

http://jadex.sourceforge.net