



UNIVERSIDAD
COMPLUTENSE
MADRID

Desarrollo de Aplicaciones y Sistemas Inteligentes (DASI) Sistemas Multiagentes

Juan Pavón Mestras

Dep. Ingeniería del Software e Inteligencia Artificial UCM

Agentes

- Los agentes representan **actores heterogéneos** en **sistemas dinámicos**
- Los agentes se suelen concebir/modelar siguiendo una metáfora de sistemas naturales inteligentes
 - P.ej. personas, pero también animales, colonias de insectos, etc.
- Permiten representar **sistemas complejos**
 - Donde el comportamiento del sistema total no puede ser comprendido a partir del estudio separado de sus partes
 - Interacciones mutuas
 - En los cuales aparecen frecuentemente **fenómenos emergentes**
 - El fenómeno sólo puede ser descrito usando términos y medidas que no se aplican a sus componentes

Definición de agente

- Agente (Diccionario RAE):
 - *Que obra o tiene virtud de obrar*
 - El que realiza una acción
 - Persona o cosa que produce un efecto
 - *Persona que obra con poder de otra*
 - El que actúa en representación de otro (agente artístico, comercial, inmobiliario, de seguros, de bolsa, etc)
 - *Persona que tiene a su cargo una agencia para gestionar asuntos ajenos o prestar determinados servicios*
- **Agentes software:**
 - Aplicaciones informáticas con capacidad para decidir cómo deben actuar para alcanzar sus objetivos
- **Agentes inteligentes:**
 - Agentes software que pueden funcionar fiablemente en un entorno rápidamente cambiante e impredecible



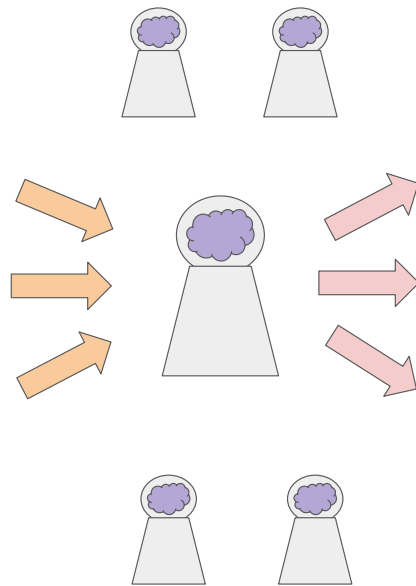
Agentes inteligentes

- El concepto de **Agente Inteligente** fue introducido en el tema 1 como una abstracción común en Inteligencia Artificial [Russell y Norvig, 2016]
- Un agente es una entidad **autónoma**
 - La autonomía implica que puede tomar sus propias decisiones y por tanto iniciar sus propias acciones
 - Capaz de operar sin supervisión humana directa
 - Persiguiendo una serie de **objetivos** → comportamiento racional
 - Embebido activamente en un **entorno** en el que interacciona
 - Entorno físico → Con componentes que no son agentes
 - Ej. sensores, actuadores, bases de datos, módulos de utilidades...
 - Entorno social → Con otros agentes



La teoría de las vocales

Autonomía
Entorno
Interacciones
Organización
Uusuario

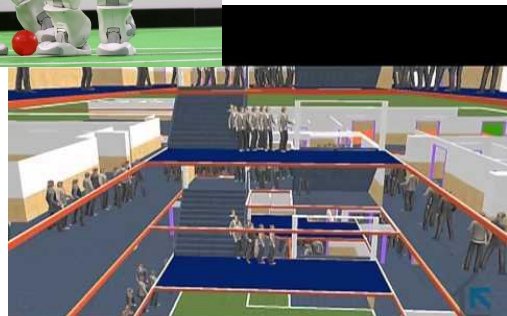
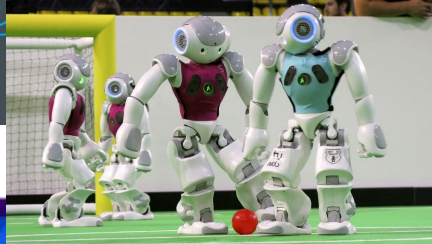


J. Pavón (UCM)

DASI - Sistemas Multiagentes

5

Ejemplos de sistemas multi-agentes



J. Pavón (UCM)

DASI - Sistemas Multiagentes

6

Autonomía

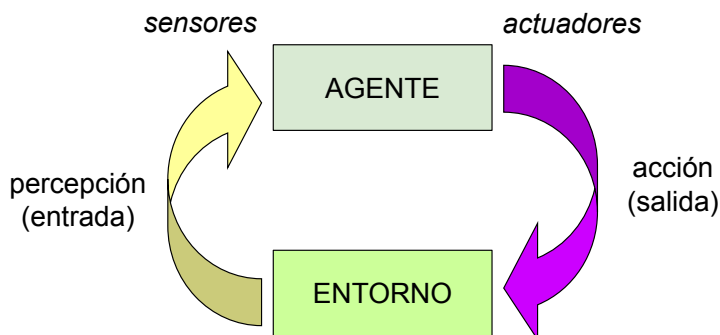
- Los agentes son:

Entidades autónomas

- **Autonomía**
 - Pueden trabajar sin la intervención directa del usuario y tienen cierto control sobre sus acciones y estado interno
- **Reactividad**
 - Pueden percibir su *entorno* (que puede ser el mundo físico, un usuario detrás de una interfaz gráfica o vocal, aplicaciones en la red, u otros agentes) y responder oportunamente a cambios que se produzcan en el mismo
- **Iniciativa**
 - El comportamiento de los agentes está determinado por los *objetivos* (metas) que persiguen y por tanto pueden producir acciones no sólo como respuesta al entorno

Entorno

- Los Agentes son entidades autónomas situadas en un entorno, con el que interaccionan mediante:
 - Percepción: sensores, mensajes, eventos
 - Acción: mediante actuadores, envío de mensajes, uso de recursos y servicios



¿Cómo influye el entorno en el agente?

- Agentes situados en entornos complejos
 - En la mayor parte de los dominios los entornos serán:
 - No deterministas
 - Sólo observables y controlables parcialmente por el agente
 - Una misma acción realizada por el agente en ocasiones diferentes puede tener efectos muy distintos
 - En general los entornos son no-deterministas
- ⇒ **Un agente debe estar preparado para fallar o para la incertidumbre de no saber si ha tenido éxito o no**
- El principal problema al que se enfrenta un agente es decidir qué acción realizar para alcanzar sus objetivos de diseño
- ⇒ **Las arquitecturas de agentes podrían verse como arquitecturas software para sistemas de toma de decisiones empotrados en un entorno**



Inteligencia

- Los agentes tienen:

Inteligencia

- **Razonamiento**
 - Un agente puede decidir:
 - qué objetivo perseguir o a qué evento reaccionar
 - cómo actuar para conseguir un objetivo
 - o suspender o abandonar un objetivo para dedicarse a otro
- **Aprendizaje**
 - El agente puede adaptarse progresivamente a cambios en entornos dinámicos mediante técnicas de aprendizaje



DOCTOR FUN

26 Sep 96



Copyright © 1996 David Farley, d-farley@tezcet.com
<http://sunsite.unc.edu/Dave/drfun.html>
This cartoon is made available on the Internet for personal viewing only.
Opinions expressed herein are solely those of the author.

In the world of the future, "intelligent agents" will waste hours watching television for us.



J. Pavón (UCM)

DASI - Sistemas Multiagentes

11

Agentes inteligentes

• Principio de Racionalidad

(Allen Newell (1982). *The knowledge level*. Artificial Intelligence **18**: 87-127):

If an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action.

... but this does not imply that it will take the best decision

⇒ El entorno es dinámico, complejo, incierto, y solo prácticamente conocido

• Razonamiento

- Un agente puede decidir:
 - qué objetivo perseguir o a qué evento reaccionar
 - qué acciones (plan) realizar para lograr un objetivo
 - suspender o abandonar un objetivo para perseguir otro

• Aprendizaje

- Un agente puede adaptarse progresivamente a cambios en entornos dinámicos



J. Pavón (UCM)

DASI - Sistemas Multiagentes

12

Componentes de los agentes

- Los agentes cuentan generalmente con:
 - Un **estado** interno
 - *Memoria, estado, estado mental...*
 - Medios para **observar** el estado de su entorno
 - *Perceptores, sensores...*
 - Medios para **actuar** sobre el entorno
 - *Actuadores*
 - Medios para **gestionar** y usar los anteriores
 - *Razonamiento, aprendizaje, comportamientos, tareas...*
- Los agentes interactúan con su entorno y otros agentes



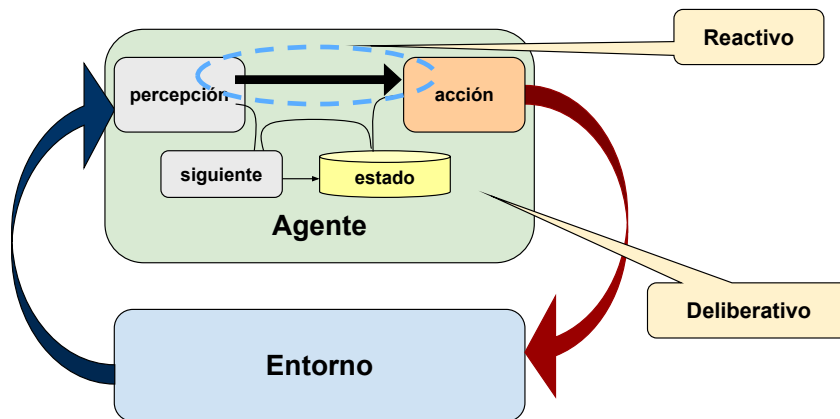
Componentes de los agentes

- Se suele decir que un agente...
 - Opera en un entorno → **situado**
 - Formado por otros agentes y todos los artefactos a que estos tienen acceso
 - Cuyo estado y eventos percibe y le afectan
 - Al menos parcialmente
 - Persigue unos objetivos → **intencional**
 - Estado del mundo (o parte de él) que el agente trata de lograr o evitar
 - Para lo cual cuenta con una serie de capacidades
 - Tareas que, dadas unas ciertas condiciones, pueden ejecutarse y potencialmente satisfacer ciertas condiciones
 - El agente decide si las ejecuta o no
 - Pueden fallar porque el agente no controla totalmente el entorno



Tipos de agentes

- Diferencia clave en el procesamiento interno
 - Agentes reactivos
 - Relación directa percepción → acción
 - Agentes deliberativos
 - Relación mediada percepción → estado mental → acción



Agentes reactivos

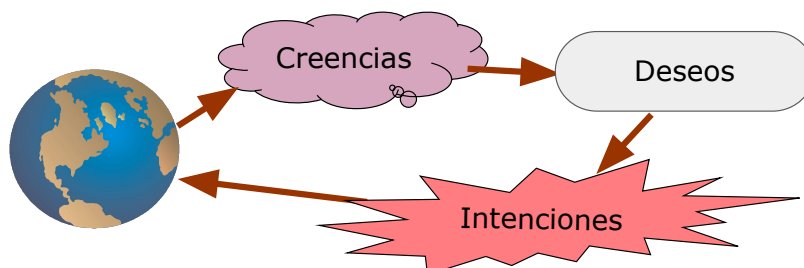
- Los agentes *reactivos* funcionan en **ciclos percepción → acción**
 - Percepción del entorno en un momento dado
 - Selección de la acción a realizar según la información recibida
 - Ejecución de la acción seleccionada
- No manejan los siguientes tipos de información:
 - Representación explícita del entorno
 - Sólo existe lo que se percibe → agente reactivo simple
 - Aunque a veces se guarda un histórico de percepciones → agente reactivo basado en modelo
 - Ej. no tienen un mapa del entorno ni saben que hay otros agentes hasta que los detectan, pero pueden recordar que han percibido un obstáculo
 - Conocimiento acerca del mundo y sí mismos
 - No elaboran nuevo conocimiento a partir de sus percepciones
 - Ej. no planifican para alcanzar sus objetivos en múltiples pasos

Agentes *deliberativos*

- Los agentes *deliberativos* introducen una **función de deliberación** para elegir las acciones a realizar
 - Dependiendo del diseño puede incluir, aunque no es necesario:
 - Objetivos → **agente basado en objetivos**
 - Funciones de utilidad → **agente basado en utilidad**
- La función depende del “**estado mental**” del agente
 - El estado mental contiene información para el razonamiento
 - Ej. recuerdos, leyes generales sobre el entorno, reglas de comportamiento, percepción, objetivos, etc.
- La selección de los objetivos a abordar y las tareas con que hacerlo depende de la información disponible
 - P.ej. el agente no intenta repetir un objetivo ya satisfecho, e intenta acciones que es posible ejecutar en el entorno actual

Agentes BDI

- Es el modelo más popular de agente deliberativo basado en objetivos
 - Basado en la teoría psicológica BDI = Creencia – Deseo – Intención (*Belief – Desire – Intention*) [Bratman, 1987; Rao y Georgeff, 1995]
 - Nuestra percepción del mundo nos permite crear un modelo de cómo es: las creencias
 - Pero el mundo no es ideal y tenemos una concepción idílica de cómo debería ser: los deseos
 - Para lograr hacer que el mundo sea como queremos, actuaremos sobre él, pero puede que no logremos los efectos deseados, por eso se habla de: las intenciones
 - Percibiendo el resultado de las acciones en el mundo se pueden revisar las creencias (estado mental) y realizar nuevas acciones en un proceso cíclico

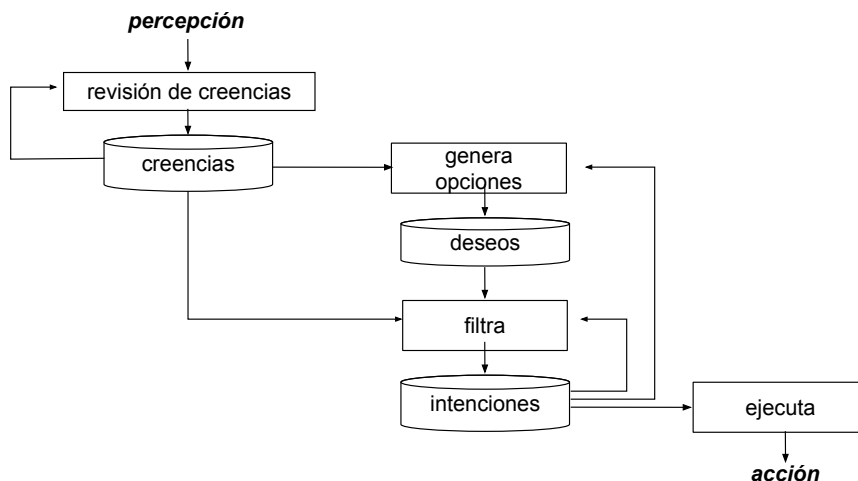


Agentes BDI

- Elementos de un agente BDI:
 - **Creencias** que representan el conocimiento sobre el entorno
 - **Deseos** son estados del mundo que el agente desea que se satisfagan
 - **Intenciones** son los *objetivos* que el agente intenta en un momento dado
 - **Planes** reflejan las acciones concretas a desarrollar para cumplir las *intenciones*
 - Esta información está sujeta a las vicisitudes del entorno real
 - Los deseos pueden ser inconsistentes entre sí
 - Ej. quiero llegar rápido al destino pero quiero consumir pocos recursos
 - Los objetivos de un momento dado (intenciones) sí han de ser consistentes y reflejar prioridades.
 - Los planes pueden fallar
 - Los agentes actúan en entornos con incertidumbre.
- ⇒ Esto es importante, hay que programar qué hará el agente, no solo cuándo tenga éxito su plan, también cuando falla

Ejemplo de arquitectura BDI

- El ciclo de comportamiento del agente es:
 - Percepción del entorno en un momento dado
 - Incorporación de la percepción al estado mental
 - Selección de la acción a realizar según el estado mental
 - Ejecución de la acción seleccionada




Agentes BDI: revisando los agentes inteligentes

- El modelo BDI puede ser visto como genérico
 - Puede modelar desde agentes puramente reactivos a deliberativos
- La orientación depende de ciertas elecciones.
 - ¿Cuándo se reacciona a los eventos del entorno?
 - ¿Cuándo se revisan las intenciones adoptadas?
 - ¿Cómo se prioriza la información?
- La selección adecuada depende de la aplicación
 - Un entorno altamente dinámico requiere una actualización rápida de creencias sobre el mismo
 - Agentes más reactivos
 - Planes de larga duración requieren un compromiso a largo plazo
 - Agentes basados en objetivos y en utilidad
- Hay que buscar un compromiso entre deliberar suficiente y actuar
 - Los recursos son limitados
 - Similar al compromiso entre exploración y explotación visto en el *Tema 3 – Aprendizaje por refuerzo*.

Interacciones

- Los agentes tienen:

Habilidad Social

- **Interacción**
 - Diálogo
 - **Delegación**
 - Asignar la realización de tareas
 - **Cooperación**
 - Trabajo en común para lograr un objetivo común
 - **Coordinación**
 - Organizar el proceso de solución del problema de forma que se eviten interacciones nocivas y que se exploten las beneficiosas
 - **Negociación**
 - Formular un acuerdo que sea aceptable por todas las partes implicadas)
- 



Interacciones

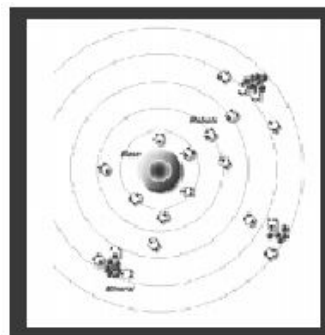
- La comunicación es la base para las interacciones y la organización social de los agentes
 - *Hay interacciones cuando la dinámica de un agente está perturbada por las influencias de otros* [O. Boissier, 2001]
 - Las interacciones son el motor de los SMA
- Distintas formas de interaccionar
 - Acciones sobre el entorno
 - Pizarra compartida
 - Inferencias
 - Paso de mensajes
 - ...



Interacciones a través del entorno

Ejemplo: robots distribuidos [Steels 89]

- Problema
 - Un conjunto de robots tienen que recoger piedras de determinado tipo de mineral (cuya localización no se conoce de antemano) y llevarlas a una nave nodriza
- Arquitectura de subsunción cooperativa
 - La cooperación no usa comunicación directa
 - La comunicación se realiza a través del entorno:
 - Campo gradiente de la señal generada por la nave nodriza
 - Permite saber cuán lejos están
 - Partículas radioemisoras que pueden recoger, echar y detectar los robots al pasar
 - Permiten indicar si una zona se ha explorado y si se hay minerales de interés que recoger o no



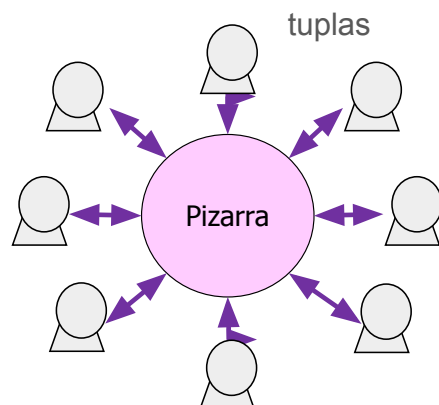
Interacciones a través del entorno

Ejemplo: robots distribuidos [Steels 89]

- Los robots ejecutan dos tipos de comportamiento en paralelo:
 - Comportamiento de manejo de objetos
 - Si detecta una piedra y no lleva ninguna -> recogerla
 - Si detecta la nave nodriza y lleva una piedra -> depositarla
 - Si llevo una piedra -> echar dos partículas
 - Si no lleva ninguna piedra y detecta partículas -> recoger una partícula
 - Comportamiento de movimiento
 - Organizados de acuerdo a una jerarquía de subsunción (ver tema de arquitecturas de agentes)

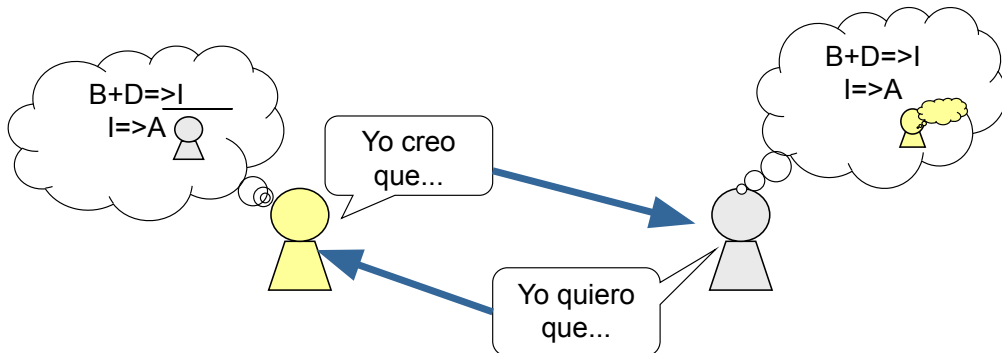
Sistemas de pizarra

- La *pizarra* es una zona de trabajo común que permite a los agentes compartir todo tipo de información
 - Puede haber varias pizarras
 - No hay comunicación directa entre agentes
- La comunicación basada en el entorno y los espacios de pueden verse como sistemas de pizarra
 - Ej. hormigas y feromonas o robots y gradientes de radiación



Comunicación en el nivel de conocimiento

- Se dice que los agentes se comunican en el nivel de conocimiento porque cuando se pasan mensajes lo que intentan es:
 - Mostrar a otros agentes su estado mental
 - Intentar modificar el estado mental de otros agentes
- Los lenguajes de comunicación de agentes mediante paso de mensajes siguen este planteamiento y se basan en teorías del lenguaje humano, en concreto en la teoría de los actos del habla



Teoría de actos del habla

• *Language as Action*

J.L. Austin (1962), *How to do things with words*, Clarendon Press

- La lingüística tradicional intentaba entender el significado de las frases indicando cómo es posible usar una combinación de palabras para hacer una declaración con significado
 - Interés en la función *denotativa* del lenguaje: determinar la verdad o falsedad de una frase
- Los actos del habla hacen referencia a la función *conativa*
 - *Conativo (def.): que intenta modificar la conducta de quien lo recibe*
 - Un acto del habla designa las acciones intencionales en el curso de una conversación

⇒ **Las interacciones son consecuencia del intento de los agentes por alcanzar sus objetivos**

Esto es, tienen una *intención* (BDI)

Teoría de actos del habla

- El lenguaje como acción:
 - Quien habla no declara solamente sentencias ciertas o falsas
 - Quien habla realiza actos de habla:
 - peticiones, sugerencias, promesas, amenazas, etc.
 - Cada declaración es un acto de habla
- Tipos de actos del habla
 - Actos asertivos: dan información sobre el mundo
Estoy de acuerdo 2 y 2 son 4 Estamos en clase
 - Actos directivos: para solicitar algo al destinatario
Siéntate ¿Cuántas pesetas son un euro?
 - Actos de promesa: comprometen al locutor a realizar ciertas acciones en el futuro
Mañana vuelvo a las 8 Te enviaré las fotos
 - Actos expresivos: dan indicaciones del estado mental del locutor
Estoy contento Gracias ¡Feliz cumpleaños!
 - Actos declarativos: el mero hecho de la declaración es la realización de un acto
Estás contratado Empezamos la clase



Teoría de actos del habla

- Componentes de los actos del habla
 - **Locución**: modo de producción de frases utilizando una gramática y un léxico
 - **Ilocución**: acto realizado por el locutor sobre el destinatario mediante la declaración (*utterance*)
 - Fuerza ilocutoria (F): afirmación, pregunta, petición, promesa, orden => PERFORMATIVA
 - Contenido proposicional (P), objeto de la fuerza ilocutoria
 - Se puede representar como F(P) (o performativa(contenido))
aserta(está nevando) responde(está nevando)
 - **Perlocución**: efectos que pueden tener los actos ilocutorios en el estado del destinatario y en sus acciones, creencias y juicios
 - Ejemplos: convencer, inspirar, persuadir, atemorizar



Teoría de actos del habla

- Ejemplo:

Cierra la puerta

- *locución*: declaración física con contexto y referencia: quién habla y quién escucha, qué puerta, etc.
- *ilocución*: acto de llevar intenciones: el que habla quiere que el que escucha cierre la puerta
- *perlocución*: acciones que ocurren como resultado de la ilocución: el que escucha cierra la puerta

Lenguajes de comunicación de agentes

- Los lenguajes de comunicación entre agentes (en inglés, ACL, Agent Communication Language) tratan de implementar la teoría de los actos del habla
 - Dos estándares
 - KSE (Knowledge Sharing Effort)
 - Knowledge Querying and Manipulation Language (**KQML**)
 - Knowledge Interchange Format (KIF)
 - **FIPA** (Foundation for Intelligent Physical Agents)
 - Especificaciones de arquitectura, infraestructura, ACL y aplicaciones
- ⇒ FIPA ACL y KQML son similares
- ⇒ KSE era una iniciativa norteamericana y al final es más usado FIPA ACL, adoptado por IEEE

FIPA (*Foundation for Intelligent Physical Agents*)

- FIPA (<http://www.fipa.org>) define un lenguaje y protocolos:
 - **Agent Communication Language (ACL)**
 - Basado en actos del habla
 - Los mensajes son acciones comunicativas
 - Es posible definir nuevas primitivas a partir de un núcleo de primitivas mediante composición
 - **Protocolos de conversación**
 - Protocolos de interacción de alto nivel
 - Patrones, secuencia típicas de mensajes
 - Protocolos básicos definidos por FIPA:
 - FIPA-request
 - FIPA-query
 - FIPA-request-when
 - FIPA-contract-net □ protocolos de cooperación (red de contratos)
 - FIPA-iterated-contract-net
 - FIPA-auction-english □ protocolos de negociación
 - FIPA-auction-ducth (subasta inglesa y holandesa)



FIPA ACL

- FIPA Agent Communication Language (ACL)

Parámetros del mensaje

:sender
:receiver
:content
:reply_with
:in_reply_to
:envelope
:language
:ontology
:reply_by
:protocol
:conversation-id

Tipos de mensaje

accept-proposal agree cancel
cfp
confirm disconfirm failure
inform inform-if inform-ref
not-understood
propose
query-if query-ref
refuse reject-proposal
request request-when
request-whenever
subscribe



FIPA

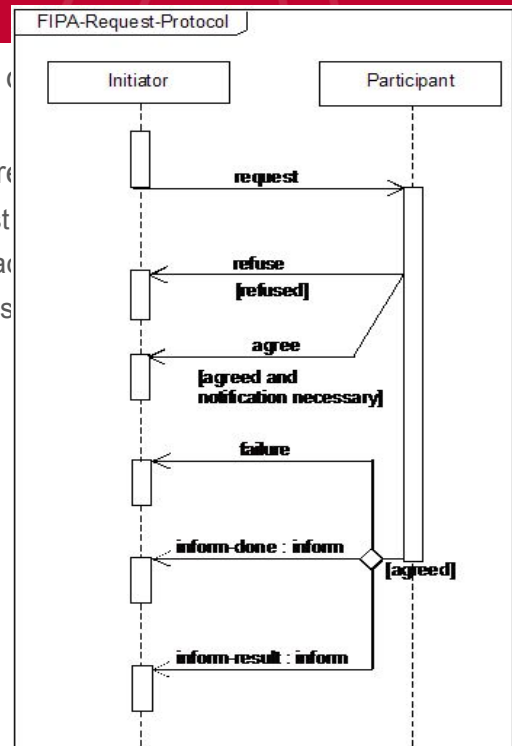
- Ejemplos de mensajes ACL
 - El request se usa con frecuencia para pedir algo
 - Suele devolverse el resultado con inform

```
( request
  :sender Usuario123
  :receiver ServidorBolsa1
  :content (PRECIO TEL ?precio)
  :reply-with accion-telefonica
  :language sl
  :ontology IBEX35
)
```

```
( inform
  :sender ServidorBolsa1
  :receiver Usuario123
  :content (PRECIO TEL 11.30)
  :in-reply-to accion-telefonica
  :language sl
  :ontology IBEX35
)
```

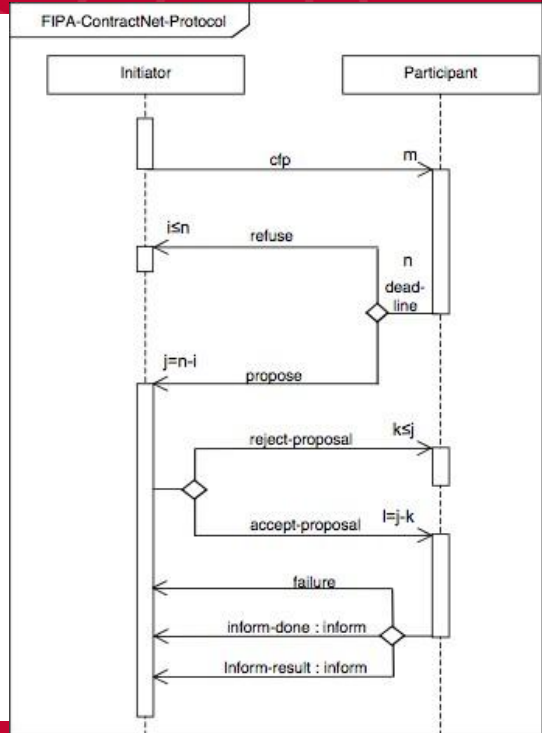
FIPA

- Los protocolos FIPA definen patrones de comportamiento habituales
- El más utilizado es el protocolo FIPA-request
 - Se solicita algo con un mensaje request
 - El agente puede (agree) o no (refuse) aceptar la solicitud
 - Se devuelve un resultado con inform o success o failure



FIPA

- Ejemplo de protocolo FIPA: **Contract net**
 - Se solicitan m propuestas para realizar una tarea, bajo unas condiciones
 - Los receptores se consideran contratistas potenciales y generarán n respuestas
 - De éstas, j son propuestas para realizar la tarea, que indican:
 - Precondiciones para realizarla (precio, cuándo se puede acabar, etc.)
 - Al cabo de un tiempo (deadline) el Iniciador evalúa las j propuestas recibidas y elige uno, varios o ningún agente para la realización de la tarea
 - Los l agentes elegidos reciben un accept-proposal y el resto k agentes un reject-proposal
 - Las propuestas son un contrato entre Iniciador y participante
 - El participante enviará un inform con el resultado de ejecutar la tarea



Ontologías

- Para dar semántica al contenido de los mensajes los agentes utilizan **ontologías**: vocabulario común en el que se han acordado significados para describir un dominio
 - Es una conceptualización del mundo, en función de objetos, cualidades, distinciones, relaciones
 - Una ontología define un conjunto de clases, funciones y constantes para un dominio de discurso, e incluye una axiomatización para restringir su interpretación
- Ejemplos de ontologías:
 - Cyc (<http://www.cyc.com/>), ontología de propósito general
 - WordNet, sistema de referencia léxica accesible por internet
 - CIA World Fact Book (<https://www.cia.gov/library/publications/the-world-factbook/>)
 - UMLS (Unified Medical Language System)

Organizaciones de agentes

- Los agentes no actúan solos, sino en grupos

Sistemas Multi-Agente

- Resolución de problemas mediante la estrategia *divide y vencerás*
 - Reparto de responsabilidades
- *Heterogeneidad*
 - Especialización
- *Concurrencia y Distribución*
 - Flexibilidad, escalabilidad, tolerancia a fallos, gestión de recursos
 - Distribución del conocimiento



Necesidad de múltiples agentes

- Los agentes individuales son limitados en entornos complejos
 - No pueden manejar toda la información necesaria
 - Complica el diseño
 - No se tienen recursos suficientes para gestionarla
 - Ej. un agente para gestionar una red con miles de sensores distribuidos
 - No tienen acceso a toda la información necesaria
 - La capacidad de percepción del agente es limitada
 - Ej. un agente no controla todas las fuentes de información
 - Tiene capacidades limitadas
 - Un agente no cuenta con toda la funcionalidad para resolver un problema complejo
- La necesidad de resolver problemas complejos lleva al uso de múltiples agentes organizados en Sistemas Multi-Agente



Organizaciones de agentes

- Una organización de agentes se define por:
 - Estructura
 - Grupos, estructura organizacional, topología
 - Roles, relaciones de autoridad, pares, etc.
 - Funcionalidad
 - Objetivos, estrategia global
 - Tareas, planes
 - Interacciones
 - Normas
 - Obligaciones, permisos, prohibiciones
 - Comportamiento dinámico
 - Entrada/salida agentes, adopción de roles
 - Workflows
 - Entorno
 - Recursos, clientes, proveedores



Organizaciones de agentes

- Dentro de la organización los agentes juegan **roles**
- Un rol es un elemento de modelado que
 - Incluye
 - Objetivos, capacidades, recursos, conocimiento, protocolos, normas...
 - Pero no describe
 - El comportamiento interno que usa los anteriores elementos
 - ¿Cómo toma las decisiones?
 - ¿Cómo implementa sus capacidades?
 - Restringe los comportamientos posibles
- Roles y agentes especifican diferentes elementos de la organización
 - El rol describe el comportamiento externo
 - Ej. cómo actuar en un cierto protocolo de comunicación
 - El agente implementa dicho comportamiento
 - Ej. cuenta con las capacidades (algoritmos) para participar en el protocolo

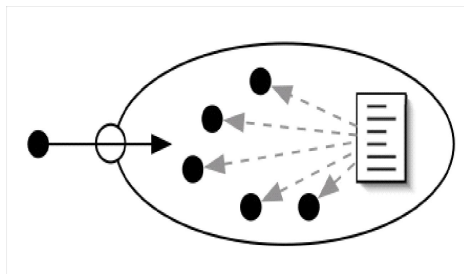


Organizaciones de agentes

- Una organización tiene **normas**
 - Definen consecuencias de las acciones de los agentes:
 - Restricciones sobre la organización
 - Obligaciones
 - Sanciones a aplicar en caso de su incumplimiento
 - Control de accesos externos
- La especificación de las normas requiere establecer:
 - Las acciones o elementos que provocan la activación de la norma
 - El conjunto de obligaciones que adquiere el agente
 - Las acciones que se deben llevar a cabo para subsanar su violación
- Hay distintas formas de implementar las normas. Algunos ejemplos:
 - OperA
 - Contratos de interacción entre agentes
 - Instituciones electrónicas
 - Capa social que garantiza interacciones conforme a las normas
 - Sociedades de agentes civiles
 - Agentes centinela que controlan los contratos
 - Ante una excepción, lanzan agentes correctores para devolver la sociedad a un estado aceptable

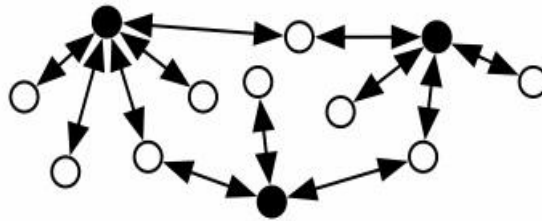
Ejemplo de organización: Sociedades de agentes

- Estructuras de largo plazo
 - Espacio común para la interacción
 - Restringido por normas sociales
- Agentes heterogéneos
 - Diferentes objetivos, nivel de racionalidad, capacidad...
- Se habla de sociedades abiertas (Open MAS) si
 - Los agentes pueden entrar (y salir) en la sociedad



Ejemplo de organización: Mercados

- Conjunto de agentes organizado según las reglas de mercado
 - Agentes competitivos
 - Los agentes compradores hacen peticiones u ofertas por un conjunto de mercancías
 - Los agentes vendedores ofrecen mercancías o eligen ofertas



Usuario

- Los SMA trabajan para los seres humanos
- Interacción hombre-máquina
 - Los agentes pueden trabajar por su cuenta (delegación)
- Ejemplo: *Chatbots* (chat robots)
 - Interacción mediante lenguaje natural
 - Alicebots
 - A.L.I.C.E. (Artificial Linguistic Internet Computer Entity)
 - Basado en AIML (Artificial Intelligence Markup Language)
 - Cada día salen nuevas herramientas para construir chatbots
 - Ejemplos:
 - <http://www.chatbots.org/chatbot/>
 - <https://botsociety.io/blog/2018/03/chatbot-examples/>

DASI

Desarrollo de SMA

SMA como paradigma de software

- En el diseño de sistemas distribuidos los agentes ofrecen:
 - Aspectos sociales
 - Lenguajes y protocolos de comunicación de agentes
 - Distribución de datos, control, conocimiento, recursos
- En el análisis de un sistema los agentes tienen un mayor grado de abstracción que los objetos o componentes:
 - Mayor autonomía y capacidad de decisión
 - Varios componentes heterogéneos que mantienen relaciones entre ellos y con escalas de tiempo diferentes
 - Modelado de sistemas naturales y sociales
- Facilitan la evolución:
 - Adaptación a modificaciones y al entorno
 - Escalabilidad
 - Añadir/quitar funcionalidad en tiempo de ejecución
 - Desarrollo incremental
 - Sistemas abiertos: capacidad de aceptar nuevos elementos

¿Cuándo usar agentes?

- Antes de desarrollar una aplicación como un SMA habría que plantearse varias preguntas:
 - ¿Se trata de un sistema distribuido abierto?
 - ¿Pueden incorporarse dinámicamente nuevos tipos de entidades en el sistema?
 - ¿Pueden cambiar las existentes?
 - ¿Es necesario considerar una evolución del comportamiento independiente para cada uno de los componentes del sistema o para una parte significativa?
 - ¿Hay incertidumbre?
 - ¿Es posible para una entidad del sistema conocer su contexto suficientemente para poder decidir con certeza el efecto de las acciones que puede realizar?
 - ¿Hay personalización?
 - ¿Un mismo servicio se puede ofrecer simultáneamente de manera distinta según las características de cada usuario?
 - ¿Hace falta definir una organización de entidades que interactúan para resolver conjuntamente problemas globales?

SMA vs. Orientación a Objetos

Objetos

- Ejecuta los métodos invocados
- Flujo de control del llamante
- Encapsula estado y comportamiento
- Estado: valor de variables
- Comportamiento: salida a partir de una entrada
- Mensajes invocan procedimiento
- Asociaciones entre objetos

Agentes

- Autonomía de decisión
- Flujo de control propio
- Encapsula la activación del comportamiento
- Estado mental: objetivos, creencias, ...
- Comportamiento: cómo decidir lo que hacer
- Interacciones: actos de habla (intencionalidad)
- Organización: relaciones sociales entre agentes

Desarrollo de SMA

- Lenguajes y plataformas de desarrollo de agentes
 - Homogeneizan los agentes proporcionando componentes y servicios básicos
 - Lenguajes de programación de agentes: Agent0, ConGolog
 - Plataformas (frameworks) de SMA:
 - Sobre JAVA: JADE, Jadex, Jack
 - Sobre Python: PADE
- Metodologías
 - Proporcionan guía para desarrollo de SMA
 - Adelfe, INGENIAS, PASSI, Prometheus, Tropos
 - [Henderson-Sellers y Giorgini, 2005] describe las más importantes
 - Algunas disponen de herramientas de desarrollo
 - INGENIAS Development Kit (IDK), agentTool III (aT3)



DASI

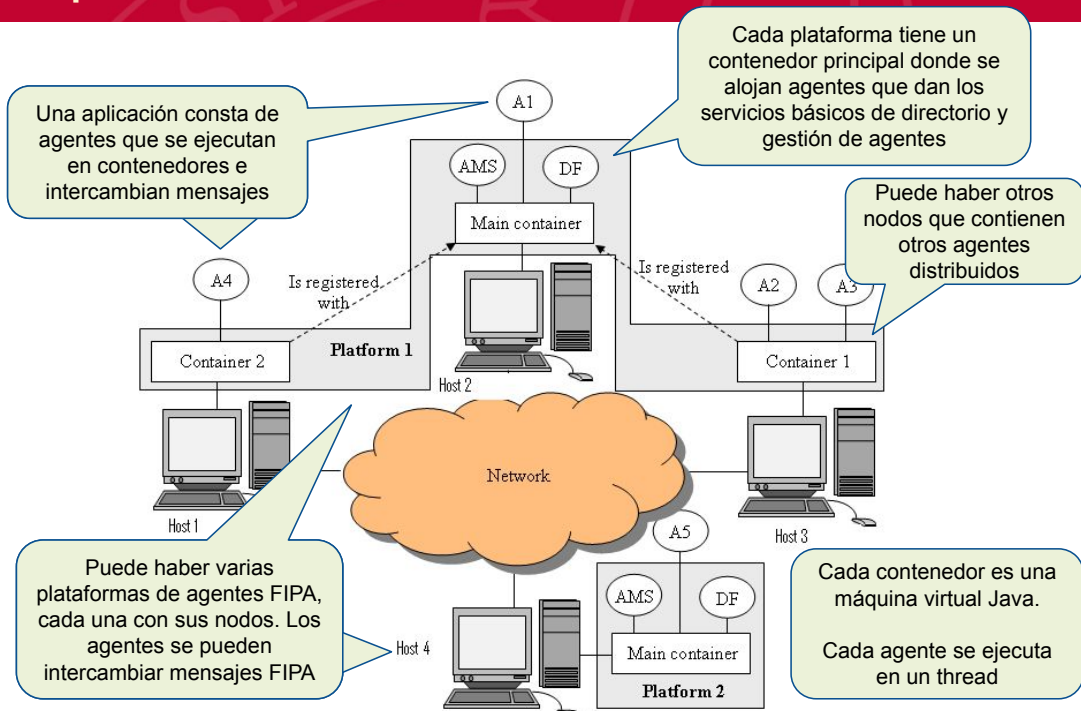
La plataforma JADE

Plataformas de agentes: JADE

- JADE (*Java Agent DEvelopment Framework*) [Bellifemine, et al., 2008] es probablemente la plataforma de desarrollo de SMA más popular
- Es una implementación en Java de los estándares propuestos por FIPA
- JADE proporciona:
 - Servicios para plataformas de ejecución de agentes
 - Ciclo de vida: creación, migración, destrucción de agentes
 - Servicios de localización de agentes (directorio) por nombre o por características
 - Comunicaciones: envío de mensajes entre agentes con protocolos FIPA
 - Librerías Java para la construcción de agentes.
 - Clases para agentes, su comportamiento, implementación de protocolos, tratamiento básico de ontologías para representación de conocimiento de dominio, etc.



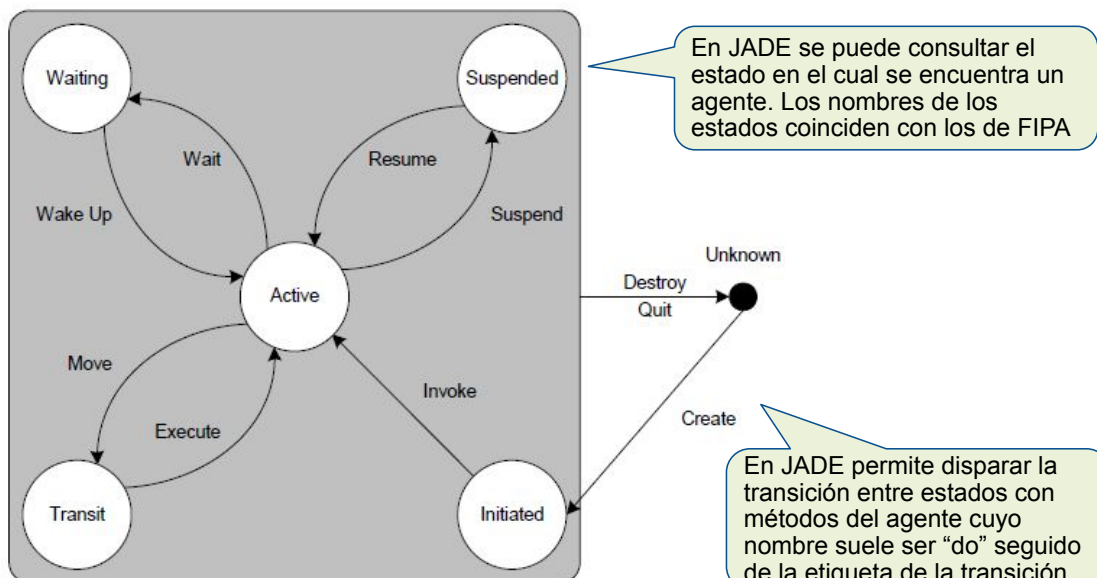
Arquitectura de JADE



JADE: Agente

- Un agente en JADE es una subclase de la clase `jade.core.Agent`
- Las tareas del agente se definen mediante comportamientos, que son subclases de `jade.core.behaviour.Behaviours`
- La clase `Agent` implementa por defecto funcionalidad básica:
 - Interacción con la plataforma
 - Ej. registro, configuración o gestión remota
 - Comportamiento del agente
 - Ej. envío y recepción de mensajes
 - Ciclo de vida definido por FIPA para los agentes

FIPA: ciclo de vida del agente

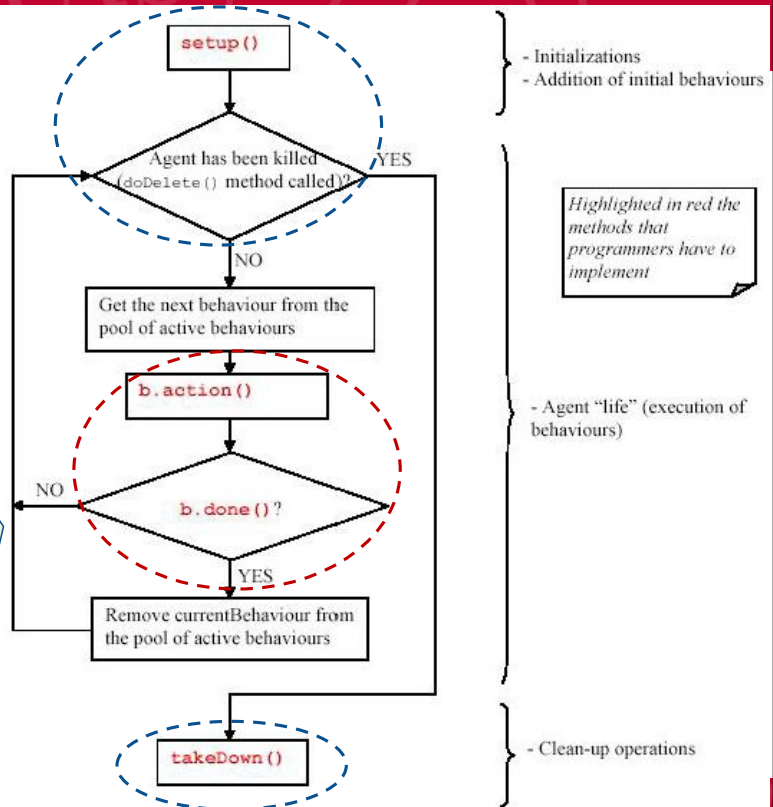


JADE: activo

Agente en estado activo

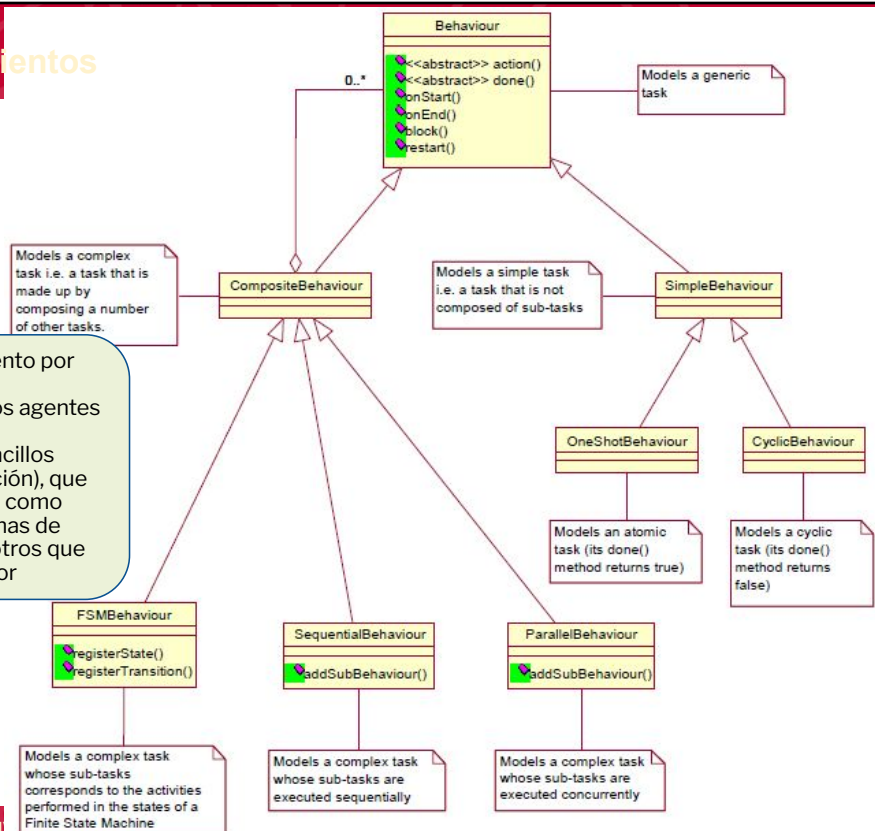
Rodeados en azul los métodos de *Agent* y en rojo los de *Behaviour*.

Una plataforma puede ejecutar varios agentes, cada uno como un *Thread* Java. Un agente puede tener programados varios comportamientos, pero sólo podrá ejecutar uno (*action*) a la vez.



JADE Comportamientos

Hay un comportamiento por defecto, Behaviour. Pero normalmente los agentes siguen patrones de comportamiento sencillos (cíclicos o de una acción), que pueden componerse como secuenciales, máquinas de estado, paralelos, u otros que defina el desarrollador



JADE - Comportamiento

- Un agente tiene comportamientos
- Un comportamiento es una subclase de `jade.core.behaviour.Behaviour`
- Un comportamiento debe implementar los métodos:
 - `action()`
 - El código de la acción del comportamiento propiamente dicha
 - No puede ser interrumpido hasta que él mismo para
 - Cuidado con bloquear la ejecución del agente en uno
 - `done()`
 - Devuelve un booleano indicando si el comportamiento ha terminado de ejecutarse
- El gestor de comportamientos del agente sólo invoca al método `done` cuando se sale del método `action`
 - Ver [JADE: activo](#)



JADE: agente básico

```
import jade.core.Agent;
import jade.core.behaviours.*;
public class miAgente extends Agent {
    protected void setup() { addBehaviour( new comportamiento(
        this ) ); }

    class comportamiento extends SimpleBehaviour {
        public myBehaviour(Agent a) { super(a); }
        public void action() {
            //...Esto es lo que hay que programar
        }
        private boolean finished = false;
        public boolean done() { return finished; }
    }
}
```

Clases básicas para agentes y sus comportamientos

Al arrancar se añade un comportamiento al agente

Comportamiento que implementará el agente

Acciones del comportamiento

Todo comportamiento ha de indicar si ha terminado o no según su estado interno.



JADE: comunicación (1/2)

- A través del AMS.
- Usando comportamientos de envío de mensajes.

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);  
msg.addReceiver(new AID("Peter", AID.ISLOCALNAME));  
msg.setLanguage("English");  
msg.setOntology("Weather-forecast-ontology");  
msg.setContent("Today it s raining");  
send(msg);
```

Creación del mensaje indicando su performativa (tipo).

"Peter" es el nombre del agente receptor del mensaje. Es un nombre local porque está en la misma plataforma del emisor. Si no fuera así, el nombre incluiría la plataforma del destinatario.

Contenido del mensaje

Acción de envío del mensaje.

Acción de recepción del mensaje.

- Y recepción de mensajes.

```
ACLMessage msg = receive();  
if (msg != null) {  
    // Process the message  
}
```

Puede haberse vuelto del método *receive* sin mensaje, típicamente por un *timeout*.



JADE: comunicación (2/2)

- La comunicación puede ser mucho más sofisticada.
 - `jade.lang.acl.MessageTemplate`
 - Para establecer filtros sobre los mensajes que se quiere recibir.
 - Protocolos
 - Secuencias de intercambios predefinidos de mensajes.
 - JADE las soporta con comportamientos específicos para el iniciador y el colaborador del protocolo.
 - Ej. `ContractNetInitiator` y `ContractNetResponder`
 - Ontologías
 - Descripciones del conocimiento del dominio.
 - Ver método `setOntology` del `ACLMessage`.
 - Lenguajes de contenido
 - Específicos según las necesidades de la comunicación.
 - Ej. inglés, objetos Java serializados o Prolog.
 - Ver método `setLanguage` del `ACLMessage`.



JADE: plataforma de desarrollo

<http://jade.tilab.com>

RMA para crear y destruir agentes en contenedores Jade

Agente dummy para enviar mensajes arbitrarios en el SMA

Sniffer para analizar comunicaciones

JADE: RMA

- El Agente de Gestión Remoto (*Remote Management Agent*, RMA) proporciona una interfaz gráfica para la gestión de la plataforma de agentes y el ciclo de vida de los agentes.
- Dentro de la plataforma de agentes existen servidores en los cuales hay *contenedores*.
 - Los contenedores son meramente lógicos y para facilitar la administración.
 - No son parte del estándar FIPA.
 - Siempre hay al menos un *Main-Container* donde están el AMS, el DF y el propio RMA.

Clase de gestión del arranque.

Opcional, es la interfaz gráfica del RMA.

- ¿Cómo arrancar la plataforma?

- `java -classpath <JARS JADE> jade.Boot -gui agent1:CLASE_AGENTE1;agent2:CLASE_AGENTE2;...`

Agentes a arrancar. El AMS, DF y el RMA (si indicado “-gui”) se arrancan por defecto.

JADE: conclusiones

- JADE es una plataforma de desarrollo y ejecución de agentes que soporta el estándar de FIPA
 - Agentes
 - Definidos en base a comportamientos
 - Ciclo de vida de agentes
 - Creación, registro, ejecución y destrucción
 - Comunicaciones
 - Envío y recepción de mensajes
 - No cubre:
 - Los modelos deliberativos de agente
 - En particular el BDI □ Usar Jadex (<https://sourceforge.net/projects/jadex/>)
 - La organización del SMA
 - Solo proporciona las comunicaciones entre agentes
- ⇒ Más información sobre cómo instalar y programar con Jade en tema anexo

JADEX: BDI sobre JADE

- JADEX [DSISG, 2011] es una plataforma de desarrollo de agentes BDI sobre múltiples *middlewares*.
 - En particular, puede usarse sobre JADE, Android y sólo.
- Se basa en el uso de librerías y ficheros Java y ficheros XML.

```
<achievegoal name="eat_food">  
  <parameter name="food" class="ISpaceObject">  
    <value>$food</value>  
  </parameter> <unique/>  
  <creationcondition language="jcl">  
    $beliefbase.eating_allowed  
  </creationcondition>  
  <dropcondition language="jcl">  
    !Arrays.asList($beliefbase.seen_food).contains($goal.food)  
  </dropcondition>  
</achievegoal>
```

Definición de un objetivo

Parámetro para la creación del objetivo

Condición de creación del objetivo

Condición de abandono del objetivo

Parámetros y condiciones hacen referencia a elementos del estado mental del agente.

JADEX: implementación de agentes

- Los comportamientos estandarizados de los agentes se pueden describir vía ficheros XML
 - Ej. gestión de objetivos, nuevas creencias, relaciones entre objetivos y creencias
- Los aspectos no estandarizados se definen vía clases Java que extienden la clase `jadex.runtime.Plan`
 - Reflejan el algoritmo de los planes
 - `body()` → acciones del plan
 - `passed()` → acciones tras la ejecución con éxito del plan
 - `failure()` → acciones tras la ejecución con fallo del plan
 - `aborted()` → acciones tras abortar la ejecución del plan
 - Otros elementos del plan como las condiciones para iniciarlo o sus efectos sobre el estado mental pueden recogerse parcialmente en los ficheros XML

JADEX: conclusiones

- JADEX es una plataforma de desarrollo y ejecución de agentes BDI
 - Definición de los elementos BDI y sus relaciones
 - Centrada en la gestión del estado mental de agentes individuales
- No cubre:
 - Los SMA
 - Salvo por las comunicaciones entre agentes
 - Y la definición de flujos de trabajo con la Notación y Modelo de Procesos de Negocio (*Business Process Model and Notation*, BPMN) [OMG, 2011]

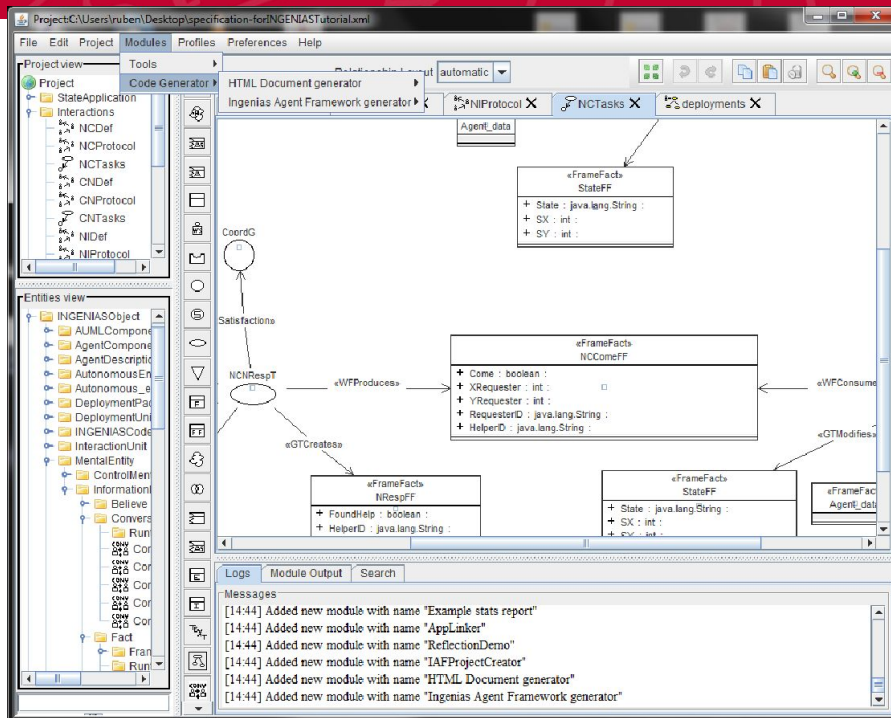
DASI

Entornos de desarrollo de SMA: INGENIAS Development Kit (IDK)

Entornos de desarrollo: IDK

- El IDK (*INGENIAS Development Kit*) [Pavón, et al., 2005] es la herramienta de desarrollo de la metodología INGENIAS
- Soporta un desarrollo dirigido por modelos de SMA
- El IDK proporciona:
 - Un entorno de modelado
 - Una API con la que desarrollar módulos que transformen especificaciones
 - Ej. generación de código, documentación o nuevos modelos
 - Módulos predefinidos
 - Ej. generación de código para el IAF (*INGENIAS Agent Framework*) desarrollado sobre JADE y documentación en HTML

IDK: interfaz de usuario

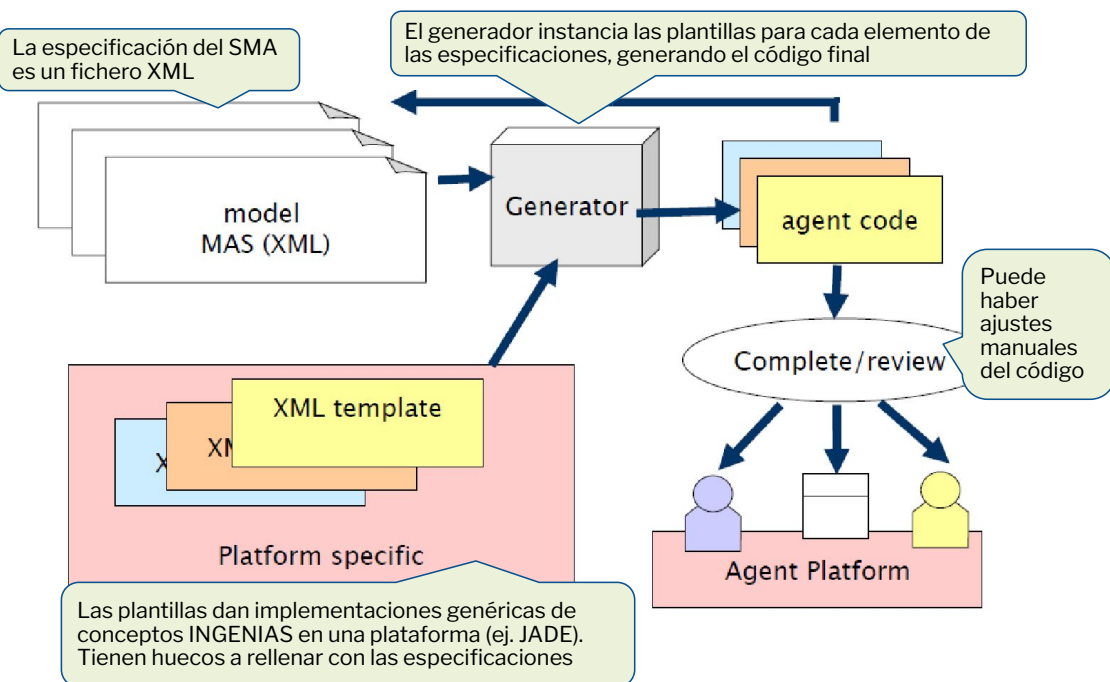


J. Pavón (UCM)

DASI - Sistemas Multiagentes

73

IDK: generación de código



J. Pavón (UCM)

DASI - Sistemas Multiagentes

74






Metodologías: INGENIAS

- INGENIAS (INGENiería de Agentes Software) [Pavón, et al., 2005] es una metodología de desarrollo de SMA
- Sus principales características son:
 - Desarrollo dirigido por modelos
 - Los desarrolladores crean especificaciones del sistema
 - Lenguaje de modelado gráfico (como puede ser UML para objetos)
 - A través de sucesivas transformaciones se genera el código final
 - Parte son automáticas
 - Ej. generar el esqueleto JADE de un agente
 - En algunas intervienen los desarrolladores para dar detalles
 - Ej. dar el código del algoritmo de decisión
 - Cubre el ciclo de desarrollo completo
 - Desde la captura de requisitos a las pruebas y despliegue
 - Con notación y procesos de desarrollo







Desarrollo dirigido por modelos

- El desarrollo dirigido por modelos (*Model-Driven Development*, MDD) es una aproximación al desarrollo de software
- Sus dos características principales son:
 - El foco en los modelos (especificaciones) para proporcionar el grueso de la información del desarrollo ⇒ Nivel de abstracción mayor que el código
 - El uso de transformaciones automatizadas de los modelos para realizar buena parte de las tareas del desarrollo
 - Ej. añadir información de diseño o generar código
- Para posibilitar esta forma de trabajo usa:
 - Metamodelos para definir los lenguajes de modelado
 - El metamodelo define los nodos, relaciones, propiedades y restricciones que han de satisfacer los grafos que representan los modelos
 - Lenguajes de transformaciones capaces de procesar metamodelos y modelos
- El hecho de usar modelos y transformaciones minimiza la labor manual repetitiva
 - El desarrollador sólo añade en los modelos los datos que no producen las transformaciones
 - Se crean nuevas transformaciones cuando es necesario

INGENIAS: marco conceptual y notación (1/2)

Concepto	Significado	Icono
Agente	Elemento activo con objetivos explícitos capaz de iniciar acciones involucrando a otros componentes	
Rol	Agrupación de objetivos y tareas a nivel de especificación. Cuando un agente juega un rol adquiere sus objetivos y ha de ser capaz de realizar sus tareas.	
Aplicación del entorno	Elemento del entorno. Los agentes actúan sobre el entorno usando sus acciones y lo perciben a través de sus eventos.	
Objetivo	Estado que el rol/agente trata de alcanzar. Los objetivos tienen éxito o fallan cuando ciertos elementos (evidencias) están presentes en el entorno o estado mental.	
Tarea	Capacidad de un rol/agente. Su ejecución requiere la presencia de ciertos elementos en el entorno o estado mental y produce/destruye otros.	

INGENIAS: marco conceptual y notación (2/2)

Concepto	Significado	Icono
Hecho	Elemento de información producido por una tarea.	
Evento	Elemento de información producido por una aplicación del entorno.	
Estado mental	Parte del estado internos de un rol/agente. Puede contener objetivos, hechos y eventos, más condiciones sobre ellos.	
Interacción	Actividad social que involucra múltiples roles/agentes.	
Grupo	Conjuntos de roles/agentes que comparten objetivos comunes y acceso a aplicaciones.	
Sociedad	Conjunto de roles/agentes, aplicaciones y grupos, juntos con las reglas que los gobiernan.	

INGENIAS: procesos de desarrollo

- INGENIAS cuenta con varios procesos de desarrollo
 - Adaptados a proyectos de diferentes características
- El proceso tradicional está inspirado en el RUP (Rational Unified Process)
- También cuenta con un proceso ágil basado en Scrum
- En ambos casos se trata de aproximaciones de ingeniería dirigida por modelos con un uso intensivo del IDK
 - Los ingenieros especifican modelos
 - Empiezan en los más abstractos
 - Refinan hacia los más concretos, próximos al código
 - El refinamiento (añadir información) está basado en transformaciones automatizadas
 - Algunas de ellas son estándar
 - Ej. generación de código para una plataforma dada
 - El refinamiento manual también es necesario



Bibliografía (1/2)

- [Austin, 1962] Austin, J.L.: *How to do things with words*. Harvard University Press, 1962.
- [Bellifemine, et al., 2008] Bellifemine, F., Caire, G., Poggi, A., y Rimassa, G.: JADE: A software framework for developing multi-agent applications. Lessons learned. *Information and Software Technology* 50(1-2), pp. 10-21, Elsevier, 2008.
- [Bellifemine et al., 2010] Bellifemine, F., Caire, G., Trucco, T., Rimassa, G.: *JADE Programmers Guide*. JADE 4.0, Tilab, 2010.
- [Brooks, 1991] Brooks, R.A.: Intelligence without representation. *Artificial Intelligence* 47(1-3), pp. 139-159, Elsevier, 1991.
- [DSISG, 2011] DSISG: Jadex – BDI Agent System. Publicación en línea disponible en <http://jadex-agents.informatik.uni-hamburg.de/>, accedido el 15/01/2012.
- [Gómez Sanz, 2003] Gómez Sanz, J. J.: Development with the INGENIAS Development Kit: A Hello World Agent. 2003. Publicación en línea disponible en <http://grasia.fdi.ucm.es/>, accedido el 01/02/2011.
- [Henderson-Sellers y Giorgini, 2005] Henderson-Sellers, B., y Giorgini, P. (eds.): *Agent-Oriented Methodologies*. IGI Global, 2005.



Bibliografía (2/2)

- [Newell, 1982] Newell, A.: The knowledge level. *Artificial Intelligence* 18, pp. 87-127, 1982.
- [OMG, 2011] OMG: *Business Process Model and Notation (BPMN)*. OMG, 2011.
- [Pavón et al., 2005] Pavón, J., Gómez-Sanz, J., y Fuentes, R.: The INGENIAS Methodology and Tools. En *Agent-Oriented Methodologies*, Henderson-Sellers, B., and Giorgini, P. (eds.), pp. 236-276, IGI Global, 2005.
- [Rao y Georgeff, 1995] Rao, A. S., y Georgeff, M. P.: BDI Agents: From Theory to Practice. Actas de la *1st International Conference on Multi-Agent Systems (ICMAS-95)*, pp. 312-319, AAAI, 1995.
- [Russell y Norvig, 2016] Russell, S. J., y Norvig, P.: *Inteligencia artificial. Un enfoque moderno*. Prentice Hall, 2ªed., 2016.
- [Shoham, 1991] Shoham, Y.: AGENT0: A Simple Agent Language and Its Interpreter. Actas de la *9th National Conference on Artificial Intelligence (AAAI 1991)*, vol. 2, pp. 704-709, AAAI, 1991.
- [Vaucher y Ncho, 2003] Vaucher, J., y Ncho, A.: Jade Tutorial and Primer. 2003. Publicación en línea disponible en <http://jade.tilab.com/>, accedido el 01/02/2011.
- [Weiss, 1999] Weiss, G. (ed.): *Multiagent systems: a modern approach to distributed artificial intelligence*. The MIT press, 1999.