Computación Orientada a Agentes

Juan A. Botía

Departamento de Ingeniería de la Información y las Comunicaciones
Universidad de Murcia

5º Curso, Ing. Superior en Informática

1. Introducción
2. Jason
3. JADEX
4. INGENIAS IDK
   - La metodología INGENIAS
   - Modelado de conceptos típicos de agentes
5. INGENIAS
   - Meta-modelo de Tareas y Objetivos
   - Meta-modelo de Interacciones
   - Meta-modelo de agente
   - Análisis y diseño en INGENIAS

Computación orientada a agentes

Qué es AOC?

La computación orientada a agentes comprende una filosofía de resolver problemas computacionales basados en la idea de autonomía y la metáfora de agente para modelar soluciones a problemas complejos.

El concepto es tremendamente amplio por lo que vamos a ver dos filosofías muy similares aunque diferentes

- Jason
- 3APL

La plataforma Jason

¿Qué es Jason?

Es un intérprete del lenguaje AgentSpeak, mediante el cual podemos programar sistemas multi-agente y ejecutarlos en la red, de manera interpretada.

¿Y qué es AgentSpeak?

Es un lenguaje basado en lógica y orientado a la programación de agentes reactivos que hacen uso de planes y que hacen uso del modelo BDI para la arquitectura de los agentes y la lógica BDI.

Agentes BDI de nuevo

Los agentes BDI tienen tres diferentes tipos de estados (actitudes mentales)

- Informativos: creencias
- Motivacionales: deseos (en donde los *goals* son deseos alcanzables)
- Deliberativos: intenciones (siguiendo a Bratman, las intenciones son planes parciales que el agente se ha comprometido a ejecutar)

y estos son críticos para un correcto funcionamiento en condiciones de recursos limitados

¿Por qué esta aproximación al desarrollo de sistemas software?

Las siguientes condiciones las exhiben muchos sistemas complejos

1. En todo instante de tiempo, el entorno puede cambiar de muchas formas diferentes (indeterminismo)
2. En cualquier instante de tiempo, el sistema puede tomar diferentes actuaciones (indeterminismo en el sistema)
3. En cualquier instante de tiempo, el sistema puede tener múltiples objetivos que cumplir
4. Las acciones que el sistema debe seleccionar para ejecutar dependen solamente del entorno y no del estado del propio sistema
5. La frecuencia a la que el entorno cambia es comparable a la que se usa para realizar cálculos o ejecutar acciones (i.e. es posible que mientras se delibera, el entorno cambie)

¿Cómo instrumentar un mecanismo de decisión apropiado para estas condiciones?

La base fundamental es la lógica modal

- Estudia el comportamiento deductivo de expresiones que incluyen "necesariamente" y "posiblemente".
- Ejemplo [**?**]:

  "mientras escribía la tesis el partido que gobernaba en España era el PP"

  se refiere a una sentencia con una verdad circunstancial que, en otro momento, podría ser perfectamente falsa. Por otro lado

  "la raiz cuadrada de 2 no es un número racional"

  es una verdad necesaria, en el pasado, presente o futuro.

Una breve introducción a lógica modal

Articulada mediante la teoría de los mundos posibles

- El juego de las siete y media
  - Si nuestro agente tiene un Aa (i.e. el agente tiene esto como creencia), ¿cómo podríamos deducir qué cartas tiene el otro agente?
    - ★ Podríamos escribir cada combinación de cartas posible en un trozo de papel
    - ★ Usando la información de que dispone el agente, eliminaría aquellas en las que aparece el As que ya tiene
    - ★ Si llamamos mundo a cada uno de esos trozos de papel, cada uno es un posible estado de las cosas, dadas las creencias del agente
- Algo que es verdad en cada uno de los mundos posibles, entonces forma parte de las creencias del agente (i.e. que tiene un As)
- Algo que es posible (i.e. solo se encuentra en alguno de los mundos) se dice que es probable

Una breve introducción a lógica modal (y II)

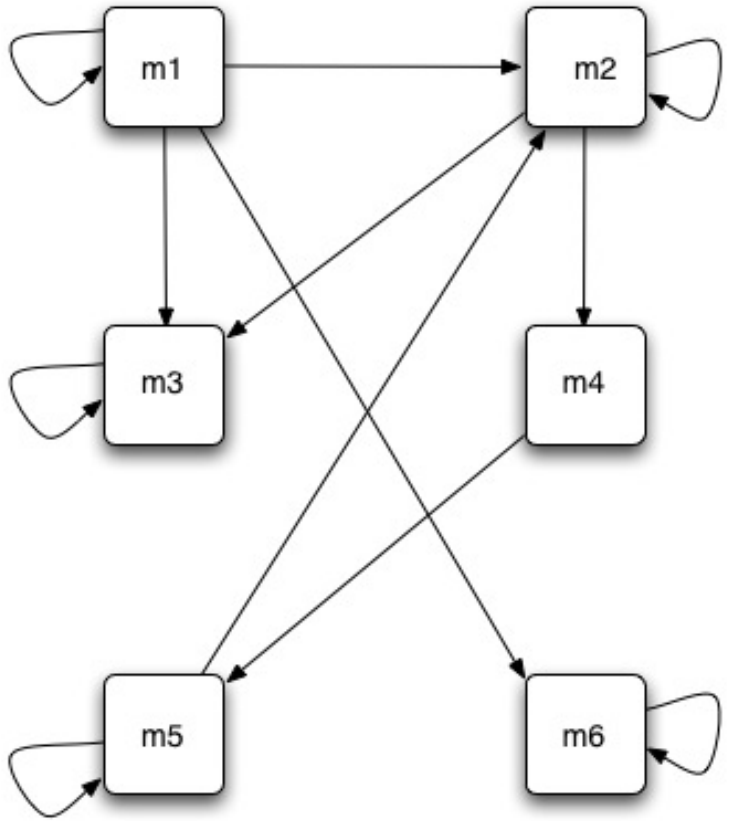
La lógica modal de Kripke

- Los símbolos incluyen $\neg$ para la negación, $\rightarrow$ para la implicación, $\Box$ para el operador de necesidad y $\Diamond$ para el de posibilidad
- Una lógica modal está asociada con valores de verdad en los mundos posibles
  - Si algo ocurren en todos los mundos, es necesario (e.g. "Siempre habrá cielo y tierra")
  - Si algo ocurre en alguno, es posible (e.g. "Es posible que volemos a Júpiter")
- Para interpretar fórmulas necesitamos
  - un conjunto de mundos W y una relación binaria $R_{W \times W}$ que indica los mundos que son posibles a partir de otros mundos y que se denomina de accesibilidad.
  - una interpretación que asigne significados a proposiciones y predicados básicos

Un ejemplo

Sea $W = \{m_1, m_2, m_3, m_4, m_5, m_6\}$ y las proposiciones básicas p, q, r, s con los significados y la función de accesibilidad representada en el grafo

Mundo	p	q	r	s
m_1	v	v	f	v
m_2	f	v	v	f
m_3	v	f	f	f
m_4	f	v	v	f
m_5	v	v	v	v
m_6	v	f	v	v



Un ejemplo (y II)

Ahora, si se tienen las siguientes fórmulas

$$A : \Box p \rightarrow \Diamond(q \rightarrow r),$$

$$B : \Diamond q \vee \Box s \rightarrow \Box(p \rightarrow r)$$

- En m_1
  - $\Box p$ es falsa ya que lo es en m_2, accesible desde m_1. Por tanto, A es verdadera
  - $\Diamond q$ es verdadera ya que q es verdadera en alguno de los mundos de W accesibles desde m_1 con lo que $\Diamond q \vee \Box s$ también lo es
  - Como $p \rightarrow s$ es falsa en m_1, el consecuente es falso con lo que B es falsa.
- En m_6, como solo m_6 es accesible, la interpretación se realiza como en cálculo proposicional
  - $\Box p$ es verdadera, $q \rightarrow r$ también, por tanto A también
  - Similar para B

Un ejemplo (y III)

Finalmente tenemos que

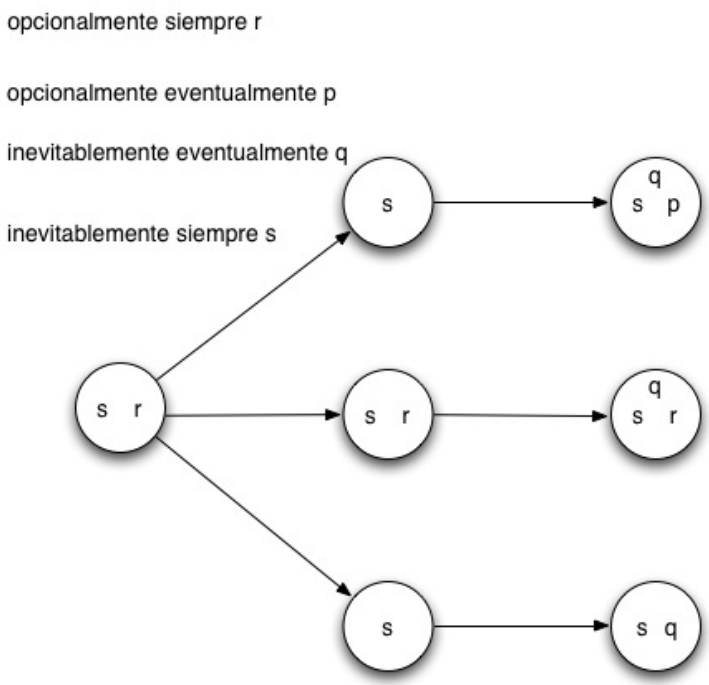
Mundo	$\Box p$	$\Diamond q$	$\Box s$	$\Diamond(q \rightarrow r)$	$\Box(p \rightarrow r)$	A	B
m_1	f	v	f	v	f	v	f
m_2	f	v	f	v	f	v	f
m_3	v	f	f	v	f	v	v
m_4	f	v	f	v	v	v	v
m_5	v	v	f	v	v	v	v
m_6	v	f	v	v	v	v	v

Ver ejemplo en el libro de Cuena [1]

¿Cómo modelar los $m_1, m_2, \ldots, m_n$?

Vamos a usar árboles temporales, en donde cada nodo es un instante de tiempo (i.e. una situación) que puede desdoblarse en el instante de tiempo siguiente mediante ramas a más nodos

- Dado un nodo, una acción ejecutada por el agente transformará una situación en otra
- Las ramificaciones en un nodo equivalen a posibles opciones del agente a ejecutar en ese instante



Ejemplo

- p: es *posible* que Juan visite Londrés *eventualmente*, i.e. $\Diamond\Diamond p$
- r: es *posible* que Sonia *siempre* viva en Madrid, i.e. $\Diamond\Box r$
- q: es *inevitable* que el Madrid *eventualmente* ganará la décima, i.e. $\Box\Diamond q$
- s: es *inevitable* que dos y dos serán *siempre* 4, i.e. $\Box\Box s$

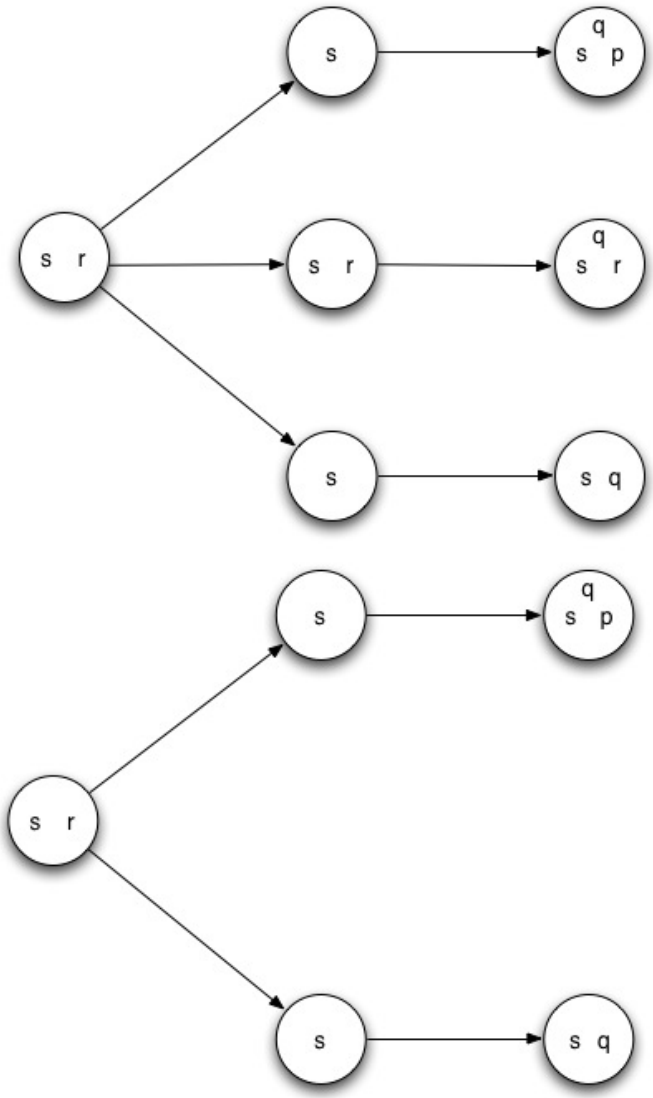
Tres tipos de mundos para tres nociones mentales

Modelado de Creencias (B)

- A cada situación se asocia un conjunto de mundos accesibles mediante creencias (i.e. mundos que el agente cree que pueden ser posibles)

Modelado de Deseso (D) en forma de *goals*

- A cada situación se asocia un conjunto de mundos accesibles mediante objetivos
- Representan objetivos alcanzables y consistentes entre sí
- Se exige que por cada mundo posible accesible mediante creencias, exista uno contenido en él accesible mediante objetivos



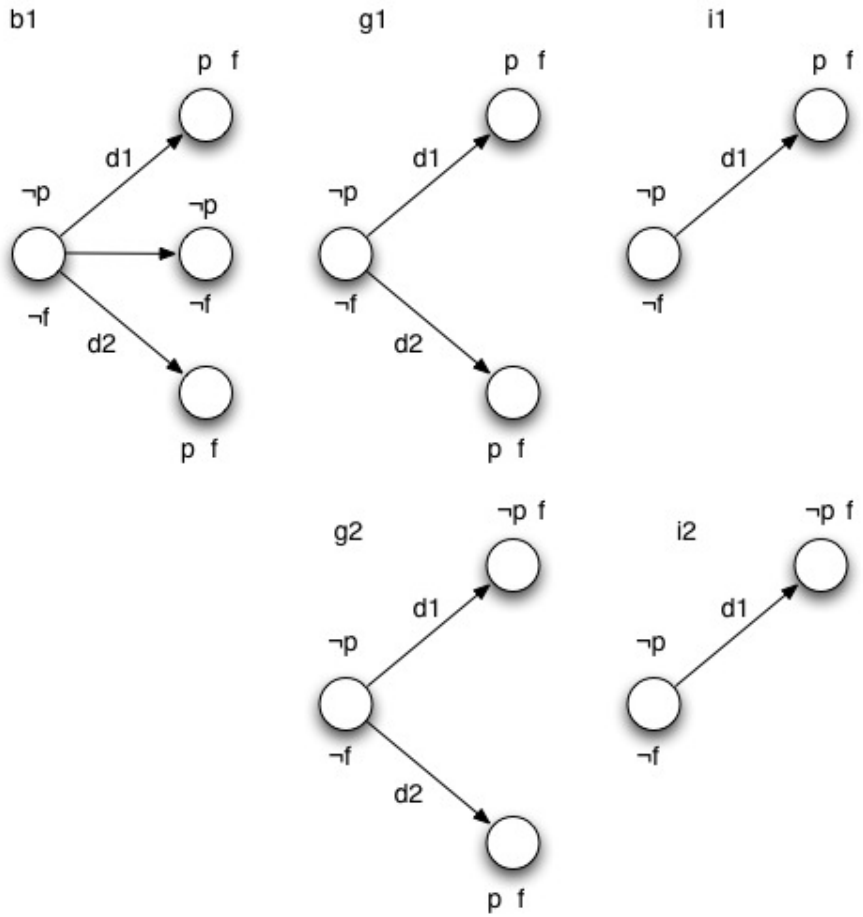
Tres tipos de mundos para tres nociones mentales (y II)

Modelado de Intenciones (I)

- Las intenciones son objetivos que el agente se ha comprometido a *intentar* cumplir
- Al igual que ocurren con creencias y goals, para cada mundo posible accesible mediante objetivos w en el instante t, existe un submundo de w en ese mismo instante, accesible por intenciones (i.e. el agente se compromete a ejecutar un curso de acciones)
- OJO: en la relación *submundo*, tanto los objetivos como las intenciones deben ser consistentes, sin embargo, no tienen por qué ajustarse exactamente a las creencias del mundo posible alcanzable por creencias del que derivan

Tres tipos de mundos para tres nociones mentales (y III)

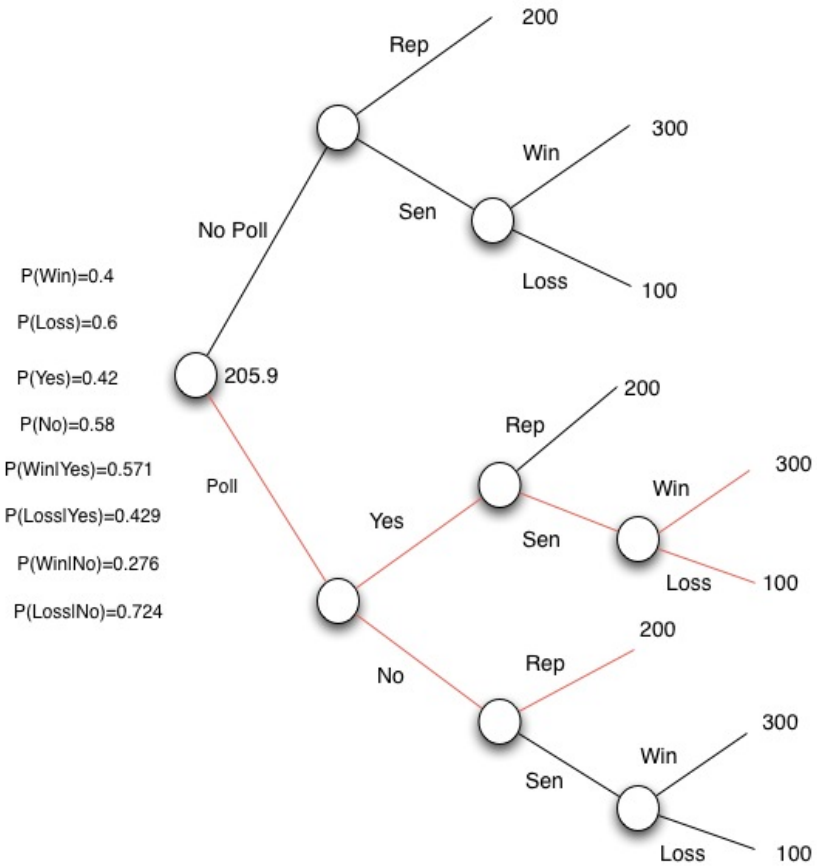
Recibir un empaste f, duele siempre, p. Así, si se quiere empastar una muela en un dentista d_1 u otro d_2, se experimente dolor (también podemos ir de compras, b). Aunque no necesariamente vamos a ir al dentista con la intención de sufrir dolor (ver [3]).



¿Cómo habilitar mecanismos de decisión?

Supongamos un ejemplo (un senador debe decidir si realizar consulta popular para dejar su ayuntamiento, Rep, y pasar a optar al senado, Sen o bien retirarse, Ret[1])

- Cada nodo refleja un estado del mundo y cada arco una acción tomada por el agente o un evento ocurrido
- Nodos de decisión: aquellos de los que parten arcos con acciones
- Nodos de posibilidad: aquellos de los que parten arcos de eventos



[1] Aunque retirarse no lo considera seriamente ya que confía en conservar su puesto (i.e. retirarse es una opción pero no tiene intención de hacerlo)

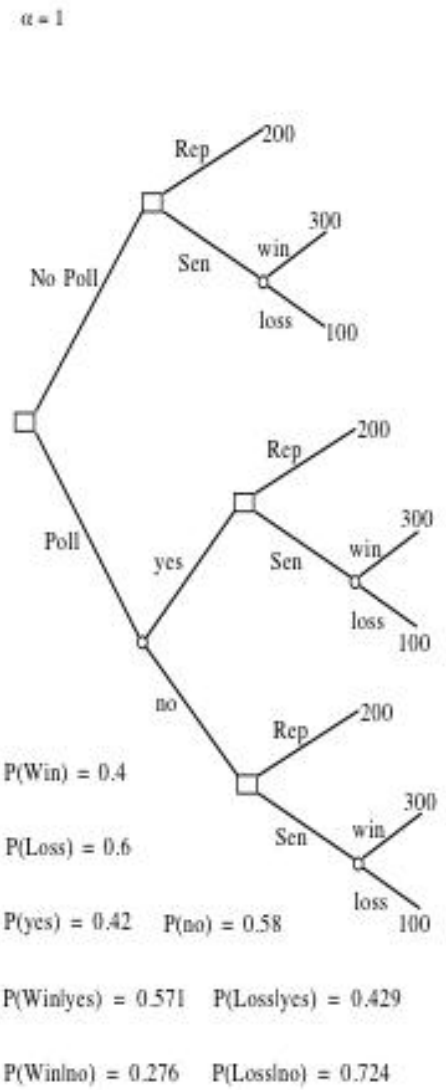
¿Cómo habilitar mecanismos de decisión?

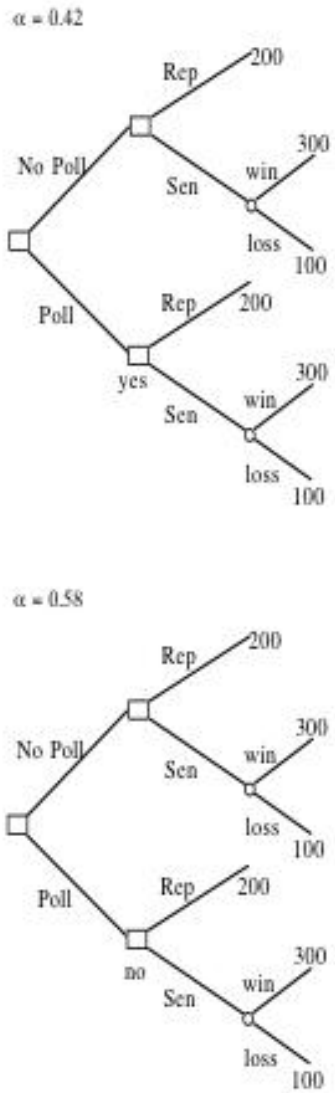
Deberíamos transforma ese árbol en una representación equivalente que

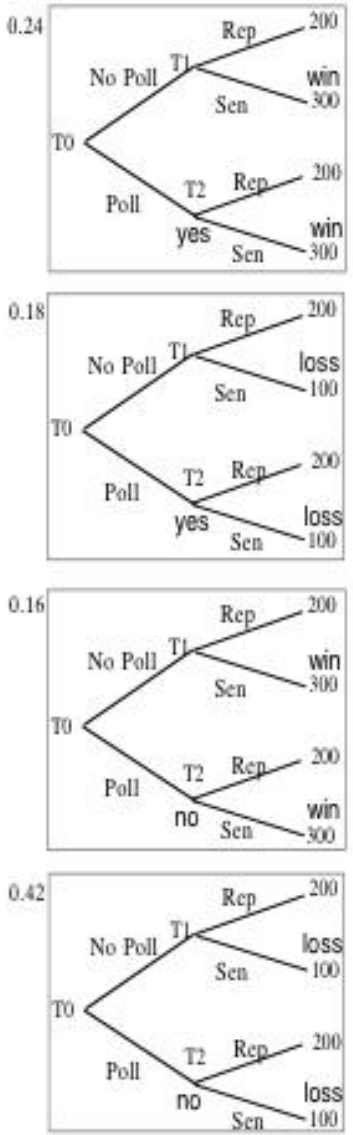
- Tuviera en cuenta, por separado, creencias, deseos e intenciones
  - Creencias: mediante un conjunto de mundos posibles que el agente cree posibles (se asocia una distribución de probabilidad al conjunto)
  - Deseos: mediante un conjunto de mundos posibles que el agente desearía alcanzar (se asocia un beneficio a cada opción)
  - Intenciones: mediante un conjunto de mundos posibles que el agente se compromete a alcanzar (se escogen aquellos cjtos. de acciones más prometedores)
- Para B, D e I, deberiamos poder representar relaciones de accesibilidad entre los mundos posibles
- Si es posible, tenga en cuenta las condiciones (1) y (2) del entorno

Procedimiento (consultar [2, 4])

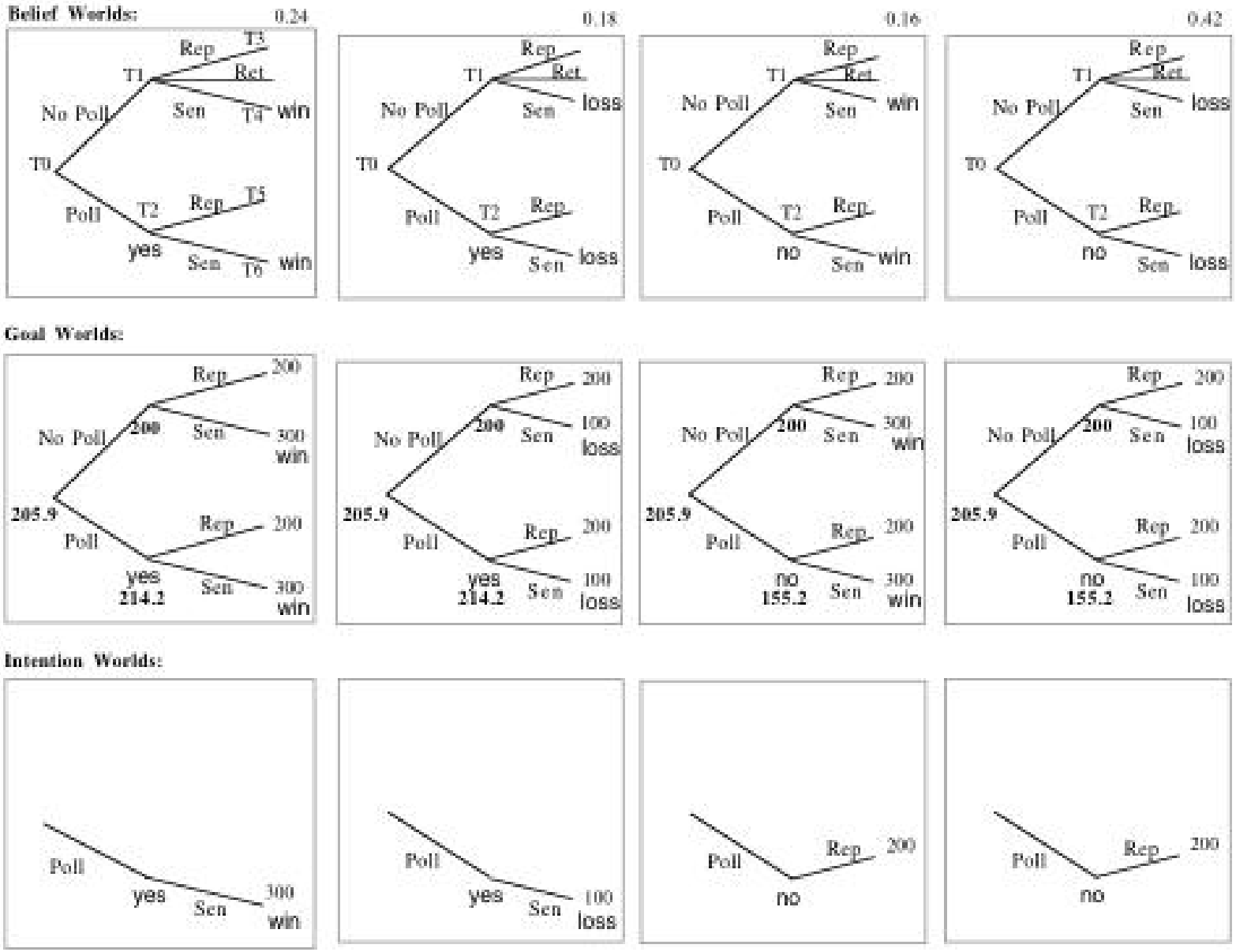
Mundos posibles para D







Mundos posibles para B, D e I



¿Cómo enlazar esto con la lógica modal?

Dependerá de la plataforma!

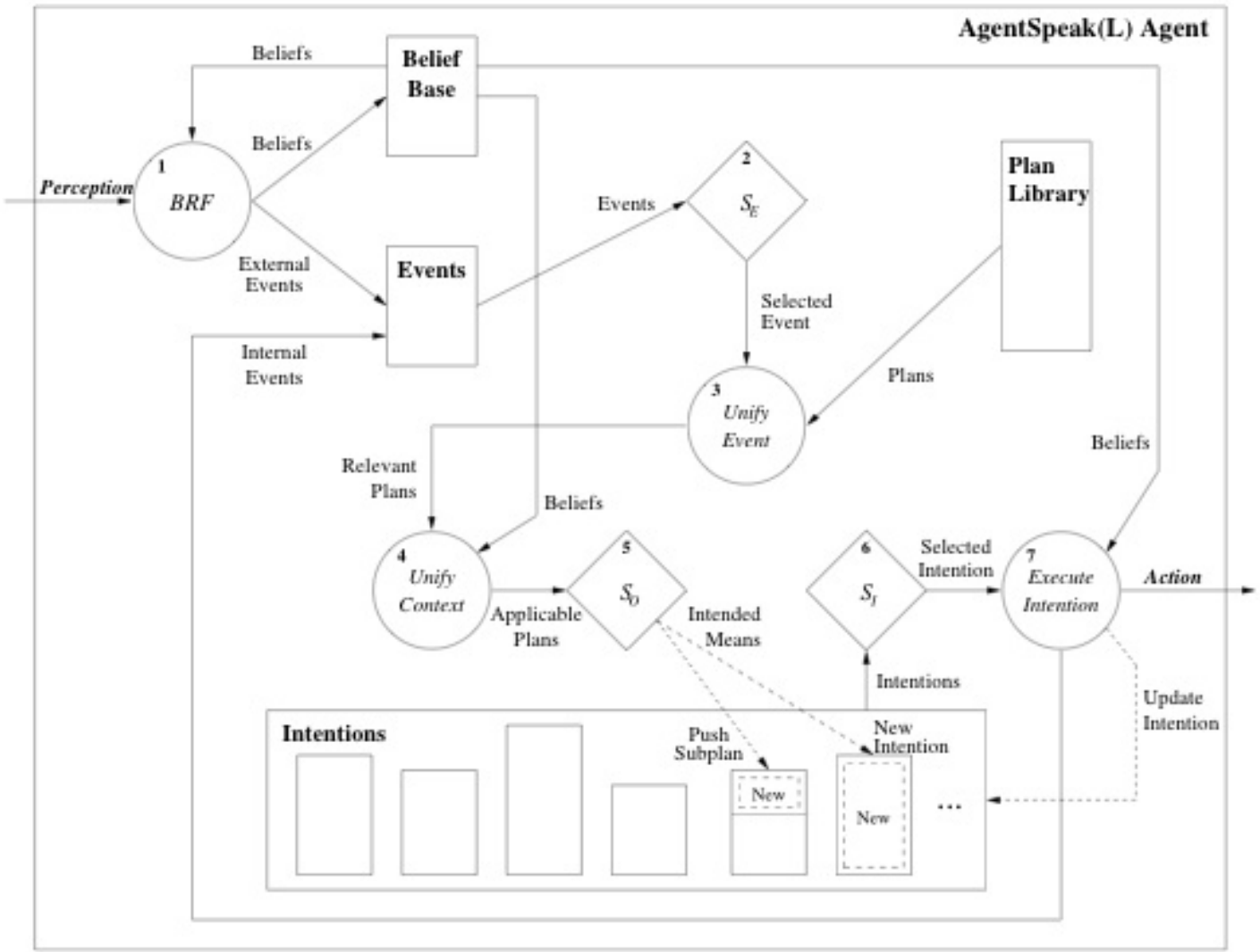
Modelo de agente en Jason

Para el programador, está compuesto de

- Un conjunto de creencias
  - Hechos básicos (i.e. *ground facts*) expresados en lógica de predicados de primer orden
- Un conjunto de objetivos (de cumplimiento, '!' y de test, '?')
- Una librería de planes
- Un conjunto de eventos (reflejan cambios en las creencias o *disparan un determinado plan*)
- Un conjunto de intenciones

Los tres últimos gestionados mediante funciones de selección, S_O, S_E S_I

La arquitectura de un agente Jason



Algunas nociones sobre la semántica formal del lenguaje de programación

Está expresada mediante semántica operacional

- Usada en el contexto de los intérpretes (vs. traductores)
- Basada en reglas que traducen unas expresiones del lenguaje en otras más sencillas (en Jason, se pasa de una configuración a otra)

Una configuración es una quíntupla $\langle ag, C, M, T, s \rangle$ donde

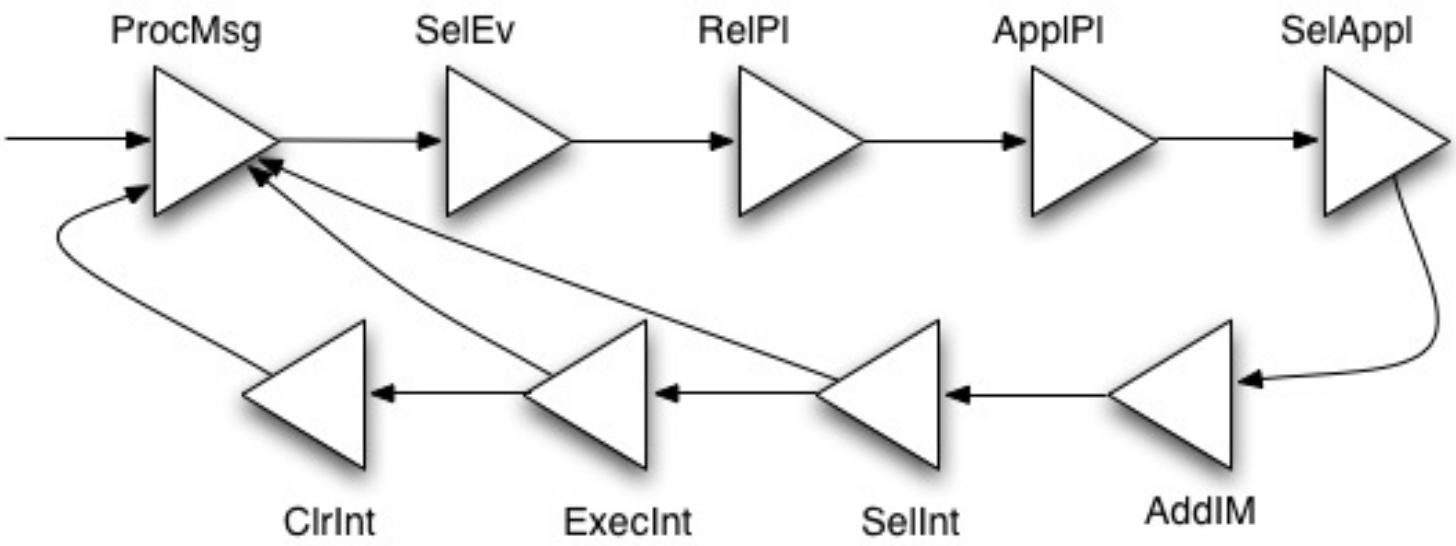
- ag es un agente compuesto por un cjto. de creencias y planes
- C es la circunstancia del agente, estructurado en $\langle I, E, A \rangle$
  - I es un cjto. de intenciones $\{i, i', \ldots\}$ en donde cada i es una pila de planes a ejecutar
  - E es un cjto. de eventos $\{(te, i), (te', i'), \ldots\}$ que se generan como resultado de las operaciones que la función de revisión de creencias realiza (añadir o quitar eventos, i para evento interno, $\top$ para externo)
  - A es un conjunto de acciones a ejecutar en el entorno

Algunas nociones sobre la semántica formal del lenguaje de programación (y II)

Seguimos con la definición de $\langle ag, C, M, T, s \rangle$

- M es $\langle In, Out, SI \rangle$ donde
  - In incluye todos los mensajes $\rangle mid, id, ilf, cnt \langle$ enviados a este agente
  - Out incluye todos los mensajes $\rangle mid, id, ilf, cnt \langle$ listos para enviar a otros agentes
  - SI almacena las intenciones suspendidas a la espera de contestación
- T es la tupla $\langle R, Ap, i, \epsilon, \rho \rangle$ donde
  - R es un conjunto de planes relevantes para el evento actual
  - Ap es el conjunto de planes aplicables en ese momento
  - i, ϵ y ρ almacenan la intención, evento y plan actuales
- s es el paso de ejecución actual, uno entre: procesar mensaje del buzón (*ProcMsg*), seleccionar evento (*SelEv*), recuperar planes relevantes (*RelPl*), chequear cuáles son aplicables (*AppPl*), seleccionar un plan aplicable (*SelAppl*), añadirlo al conjunto de intenciones (*AddIM*), seleccionar una intención (*SelInt*), ejecutarla (*ExecInt*) y eliminar la intención cumplida (*ClrInt*)

El ciclo de razonamiento de un agente Jason



Seleccionar intenciones

Si existen intenciones

$$\frac{C_i \neq \{\} \quad S_I(C_I) = i}{\langle ag, C, M, T, SelInt \rangle \rightarrow \langle ag, C, M, T', ExecInt \rangle}$$

$$\text{en donde } T'_i = i,$$

y hacemos uso de S_I para seleccionar la siguiente intención.

Si no existen intenciones que acometer

$$\frac{C_i = \{\}}{\langle ag, C, M, T, SelInt \rangle \rightarrow \langle ag, C, M, T, ProcMsg \rangle},$$

con lo que se reinicia el ciclo de razonamiento.

Ejecución del cuerpo de un plan

Existen varias formas de ejecutar planes. El plan a ejecutar siempre está en la cima de la pila de la intención actual

- Un goal de cumplimiento

$$\frac{T_i = i[head \rightarrow !at; h]}{\langle ag, C, M, T, ExecInt\rangle \rightarrow \langle ag, C', M, T, ProcMsg\rangle}$$

$$\begin{array}{lrcl} donde & C'_E & = & C_E \cup \{\langle +!at, i[head \rightarrow h]\rangle\} \\ & C'_I & = & C_I\{T_i\} \end{array}$$

Ejecución del cuerpo de un plan (y II)

- Actualización de una creencia

$$\frac{T_i = i[head \rightarrow +b; h]}{\langle ag, C, M, T, ExecInt \rangle \rightarrow \langle ag', C', M, T, s \rangle}$$

$$\begin{array}{lrcl} donde & ag'_{bs} & = & ag_{bs} + b[source(self)] \\ & C'_E & = & C_E \cup \{\langle +b[source(self)], \top \rangle\} \\ & C'_I & = & (C_I\{T_i\}) \cup \{i[head \rightarrow h]\} \\ & s & = & \begin{cases} ClrInt & \text{if } h = \top \\ ProcMsg & sino \end{cases} \end{array}$$

¿Qué es JADEX

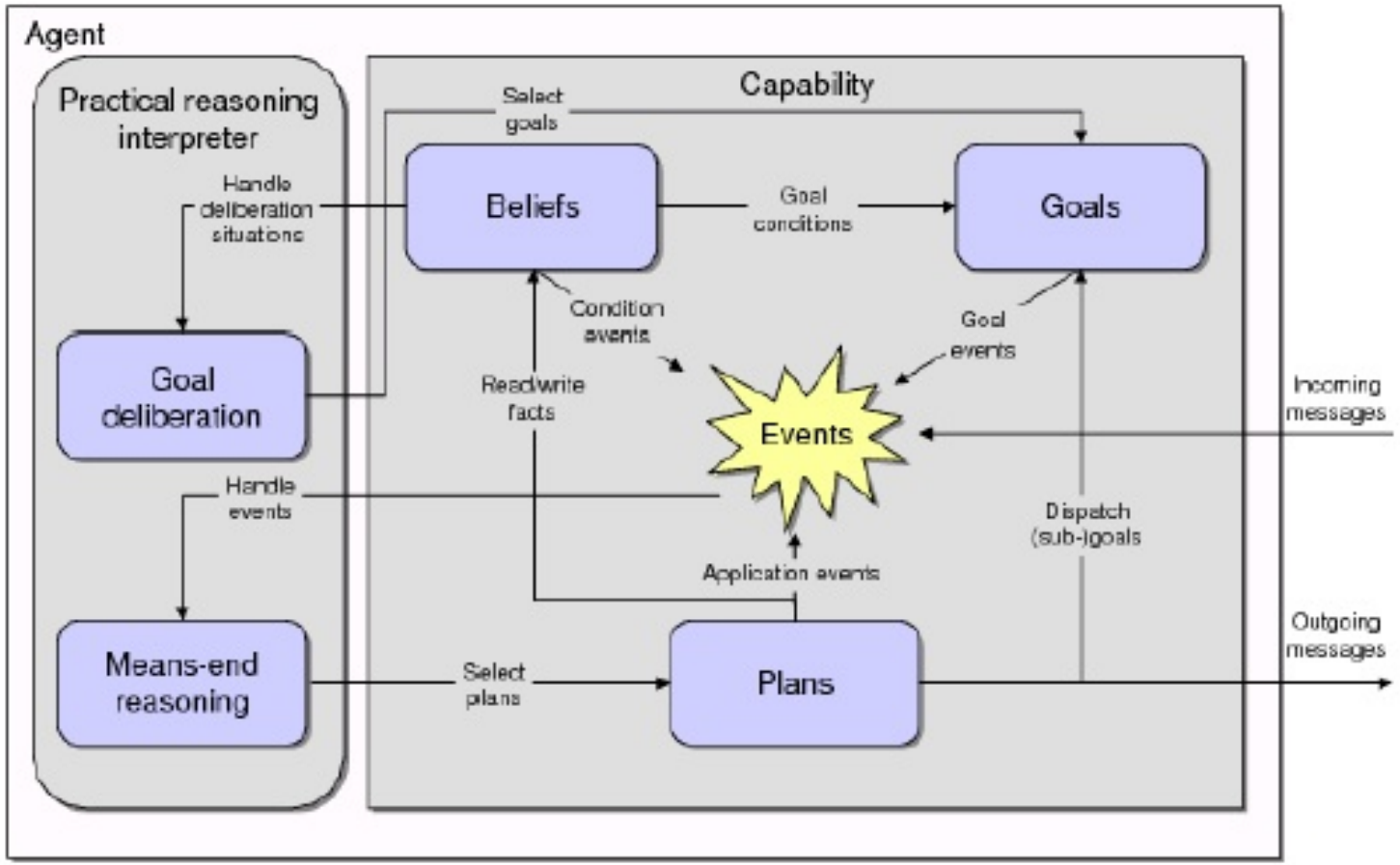
JADEX

Es un motor de razonamiento automático orientado a agentes, el cual se programa usando Java y XML y se asienta sobre middlewares como pueden ser J2EE o JADE.

Característicias principales

- Los agentes son BDI
- Utiliza razonamiento práctico (*means-ends reasoning*) para actuar frente a mensajes entrantes, eventos internos y objetivos
- En paralelo, lleva a cabo un proceso deliberativo contínuo sobre sus deseos para generar intenciones

Arquitectura abstracta de un agente JADEX



La base de creencias

La base de datos de creencias es el elemento que almacena las creencias del agente y a través del cual se acceden a las mismas

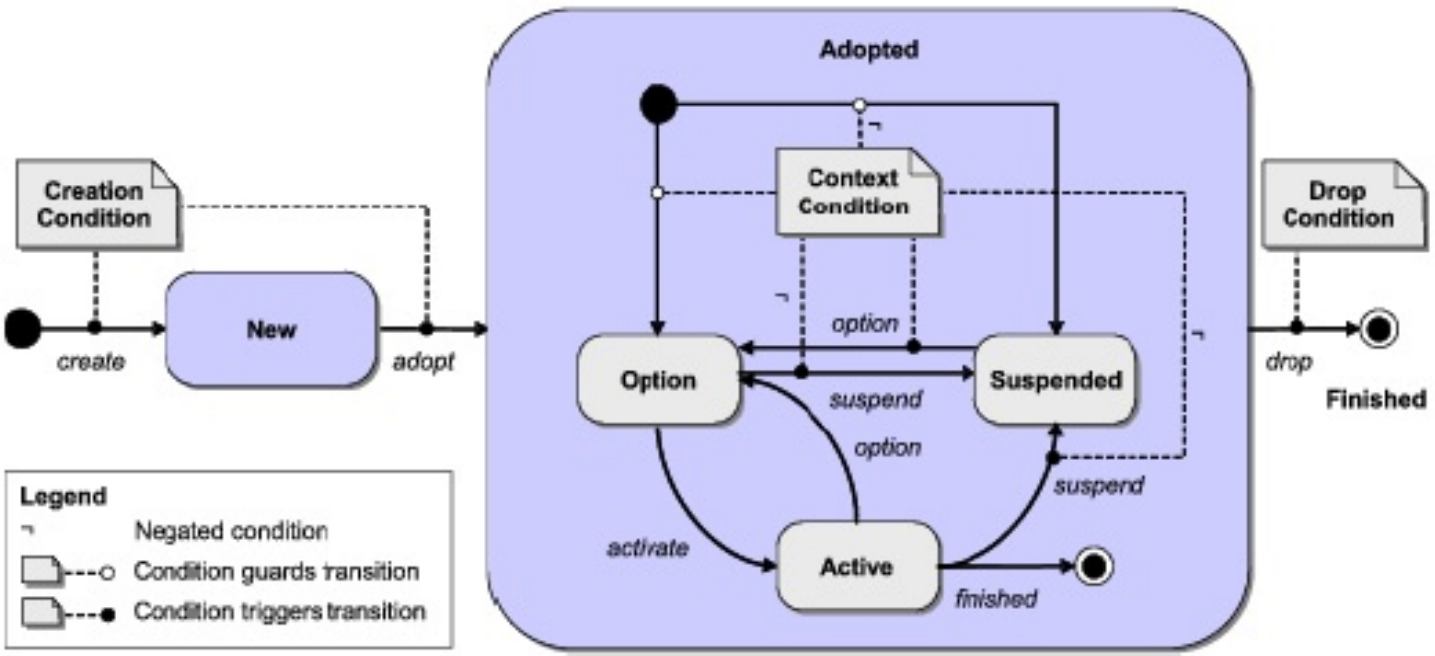
- Las creencias del agente se acceden mediante *strings*, i.e. claves primarias en una base de datos relacionas
- Cada creencia es un hecho, que al final se convierte en un objeto Java
- Hay dos tipos de creencias: hechos simples y cjtos. de hechos
- Sobre la base de creencias existen facilidades de consulta (OQL like)
- Es posible definir triggers para activar planes u objetivos cuando una creencia cambia

Objetivos en JADEX

En JADEX, los *goals* son elementos principales.

- Para cada objetivo, el agente ejecutará acciones apropiadas, orientadas a su consecución y solo dejará de hacerlo cuando bien se ha cumplido bien se detecta que no puede cumplirse o bien cuando se pierde el interés
- Cuatro tipos de goals
  - Perform goal: algo que debe hacerse, sin resultado necesario
  - Achieve goal: descrita mediante un estado a alcanzar sin decir cómo (un procedo deliberativo lo decidirá)
  - Query goal: usada para adquirir información necesaria
  - Maintain goal: un estado que debería mantenerse una vez que se llega a él

Ciclo de vida de los objetivos



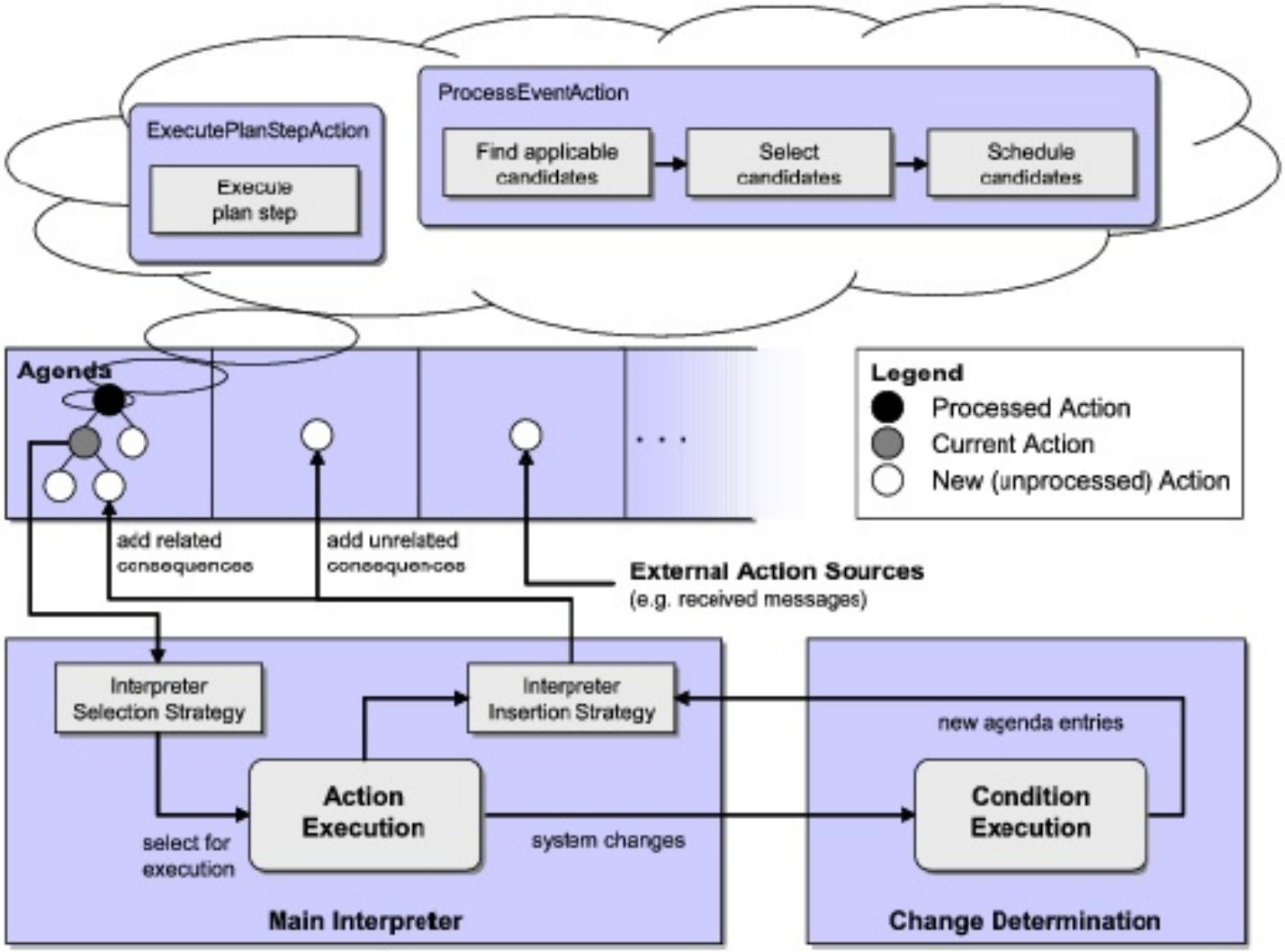
Planes en JADEX

Los planes en JADEX se usan para responder a eventos y cumplir objetivos.

Compuestos por

- Head: el conjunto de expresiones que se han de hacer ciertas para ejecutar el plan
- Body: las acciones que el plan ha de ejecutar. 100% Java
  - Puede lanzar más objetivos a cumplir
  - Esperar a que ocurra determinado evento

Modelo de ejecución de agente (ver [?])



Modos de funcionamiento

JADEX puede funcionar con varios middlewares. En su versión 0.96, puede funcionar

- De manera independiente (i.e. standalone)
  - Ejecución rápida
  - Uso de memoria optimizado
  - Código 100% + JADEX
- Haciendo uso de JADE para las comunicaciones y servicios básicos (i.e. a modo de middleware)
  - Agentes FIPA
  - podemos reusar código existente
  - Uso de ontologías mediante Protégé
  - Lenguajes de contenido

Herramientas disponibles

JADEX viene con varias herramientas que ayudarán en el desarrollo

Conclusiones iniciales

Hasta ahora, hemos visto como construir sistemas de agentes

- artesanalmente
  - Poca productividad
  - Curva de aprendizaje muy elevada
- ¿Podemos poner el énfasis en la productividad manteniendo la metáfora?
  - Podemos hacer uso de metodologías
  - Podemos diseñar con meta-modelos
  - Hasta disponemos de IDEs con esos meta-modelos integrados (IDK)

Construyendo sistemas multi-agente

Una metodología de desarrollo de software está compuesta de

- Un lenguaje de modelado para realizar el diseño
- Un proceso software que define las actividades de desarrollo y sus interrelaciones

¿Qué podemos encontrar para desarrollo de SMA?

- Metodologías de análisis y diseño (e.g. GAIA)
- Metodologías de análisis, diseño e implementación (Tropos, MASE, INGENIAS, MAS-CommonKADS)
- Lenguajes de modelado per se (AUML, UML 2.0)

INGENIAS

INGENIAS extiende la ingeniería OO con conceptos del área de los agentes software

- Diseño basado en la especificación de models
  1. Agente
  2. Organización
  3. Dominio
  4. Tareas & objetivos
  5. Interacciones
- Método de desarrollo: RUP (*Rational Unified Process*)

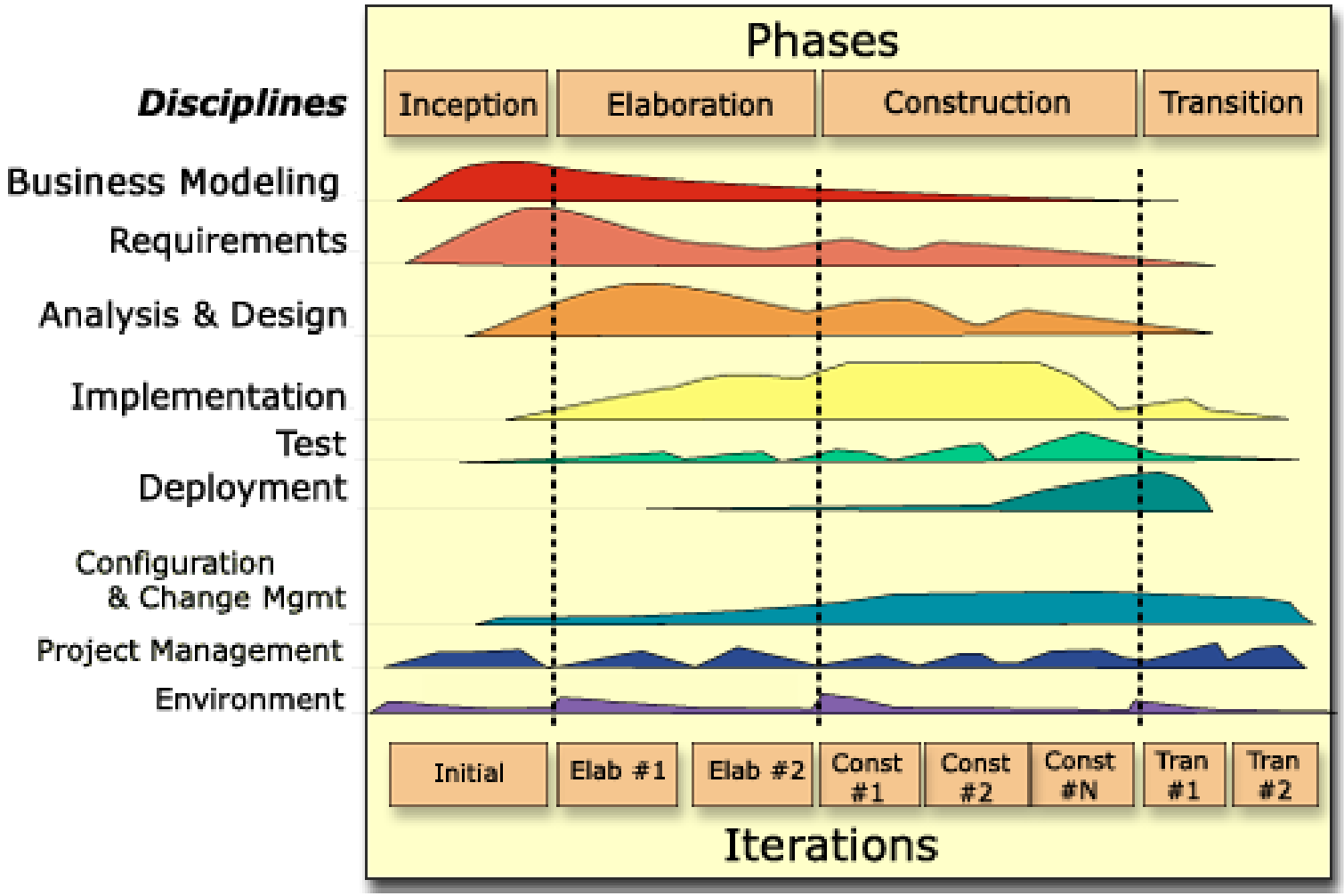
Recordatorio sobre RUP[2]

Los principales objetivos que se pretendían conseguir al crear el RUP eran

- Adaptar el proceso de desarrollo (i.e. se introducen plantillas)
- Colaboración entre diferentes equipos
- Medir el progreso en cada release (i.e. demonstrar el valor en cada iteración)
- Elevar el nivel de abstracción (de el código propio a los patrones de software)
- Tener en cuenta la calidad durante todo el proyecto (i.e. testing)

[2] www.wikipedia.org

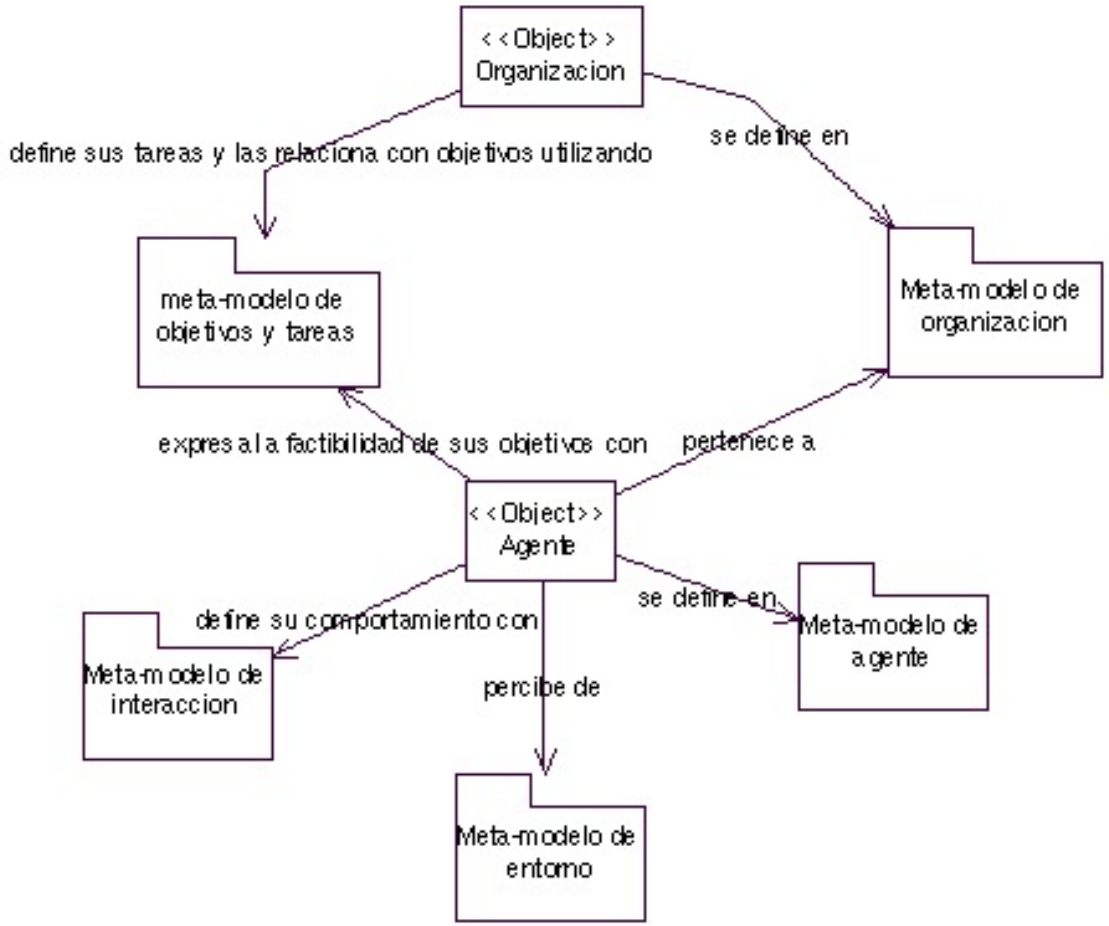
Recordatorio sobre RUP (y II)



INGENIAS, metamodelado

En INGENIAS,

$$SMA = M_{agente} + M_{interaccion} + M_{entorno} + M_{objetivos} + M_{organizacion}$$



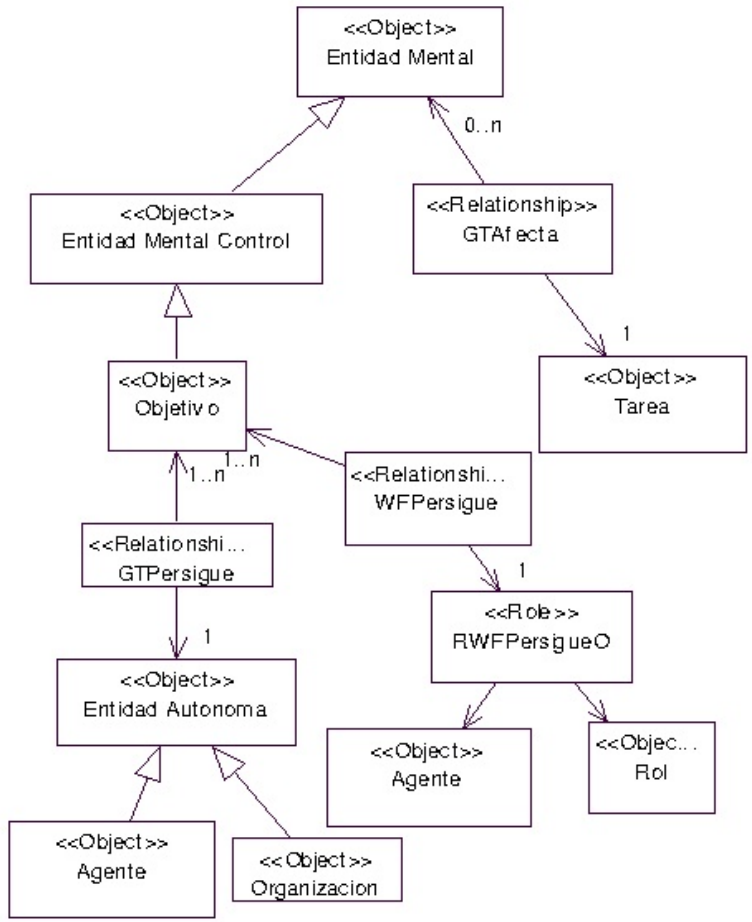
Meta-modelo de Tareas y Objetivos[3]

Este modelo expresa la **motivación** tras las tareas y qué alternativas de actuación tiene cada agente

Las organizaciones y los agentes, como entidades autónomas, persiguen objetivos

Los roles también, aunque mediante otro tipo de relación derivada de los flujos de trabajo (WF)

Las tareas afectan a entidades mentales (del agente que ejecuta la tarea) luego pueden hacer ciertos a los objetivos

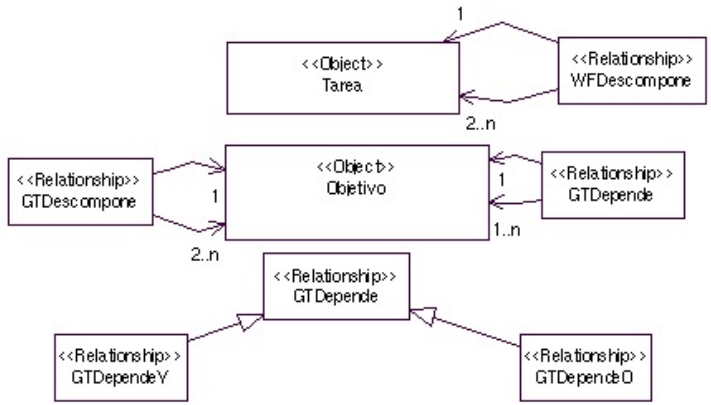


[3] http://grasia.fdi.ucm.es/ingenias/Spain/lenguaje/mtareas.htm

Las tareas y los objetivos se descomponen

Tanto los objetivos como las tareas se descomponen en subobjetivos y subtareas, respectivamente

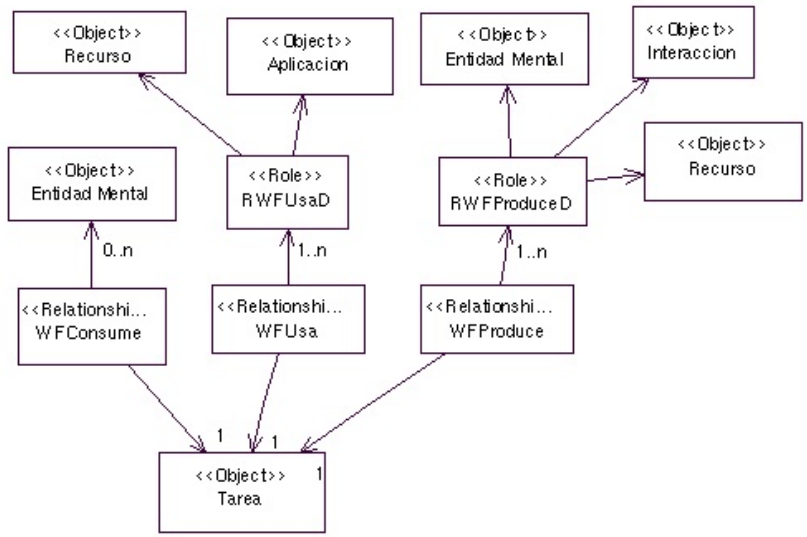
Las relaciones de dependencia entre objetivos forman árboles Y/O



Descripción de tareas

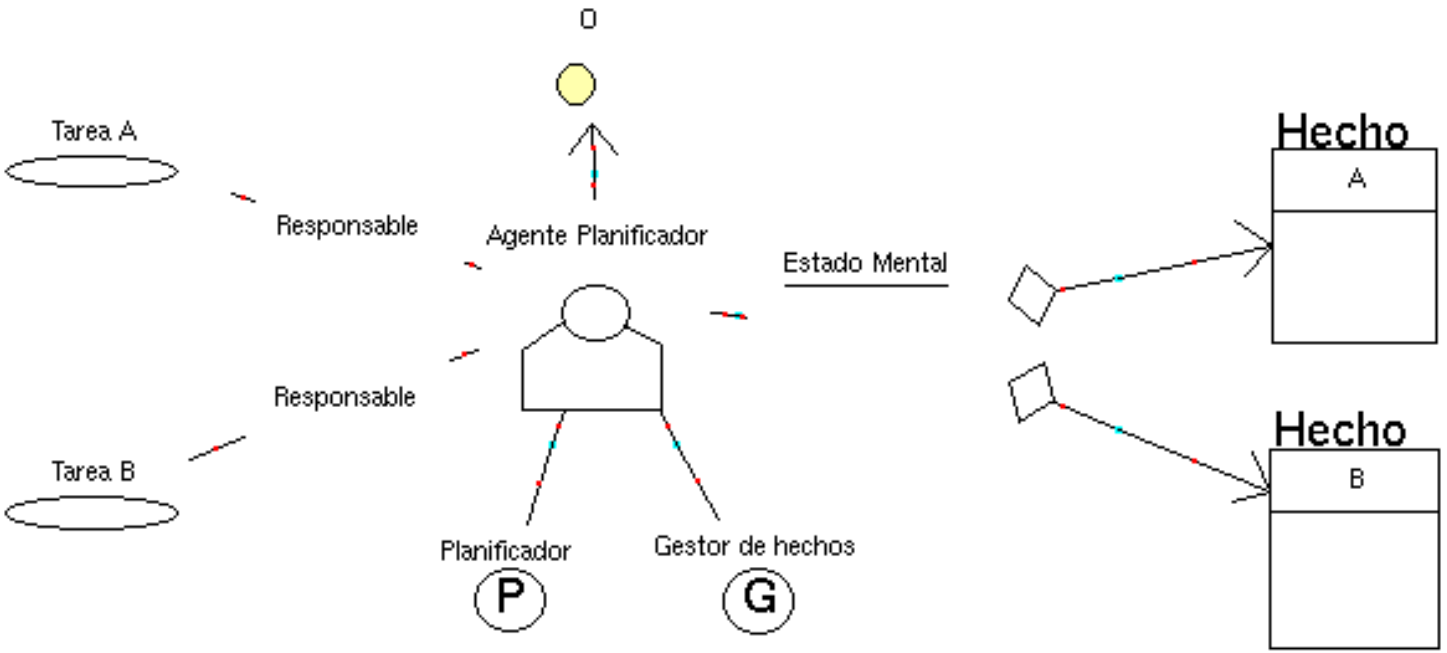
Las tareas se describen mediante precondiciones (i.e. WFConsume, WFUsa, GTAfecta) y postcondiciones (i.e. WFProduce, GTAfecta)

No se ejecutarán aquellas tareas que no satisfagan las precondiciones



Un ejemplo

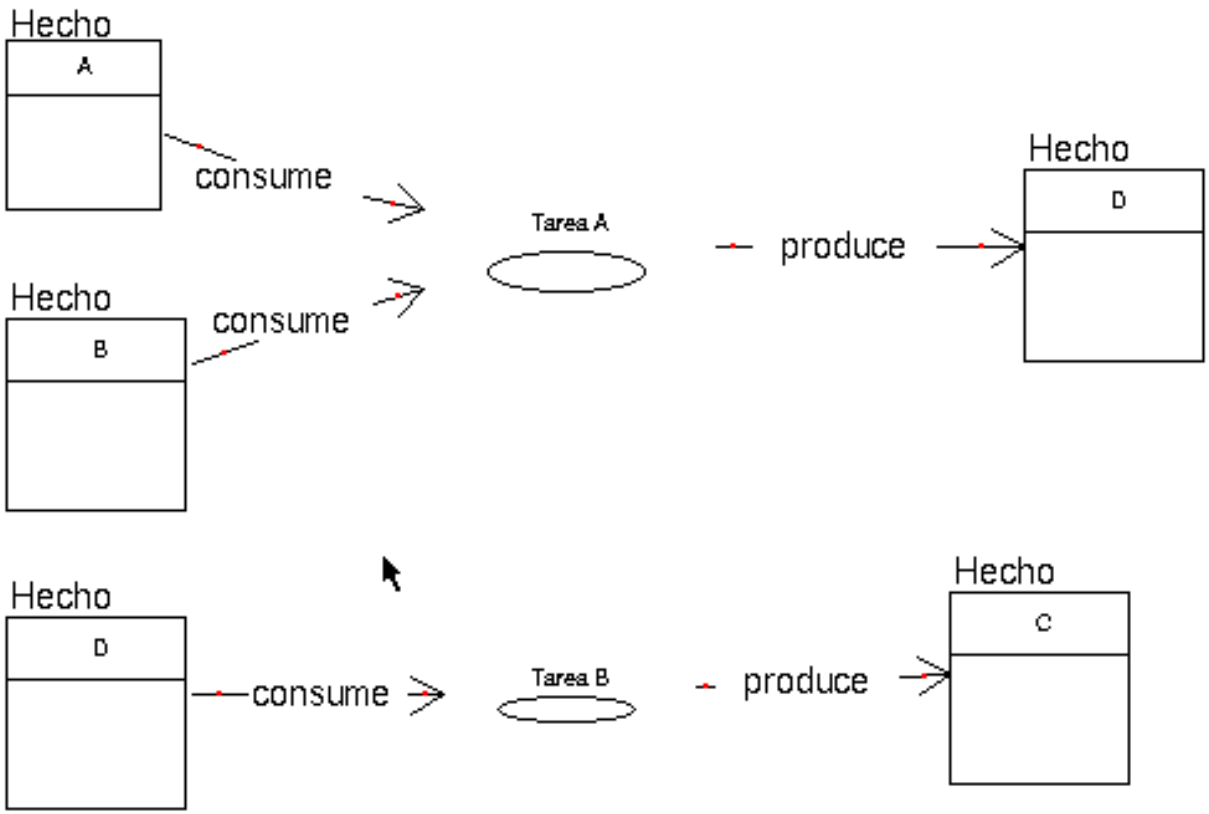
El ejemplo utilizado es el modelado de un agente planificador de tareas. El agente sabe ejecutar dos tipos de tareas: tarea A y tarea B. De la utilización de éstas depende que se alcance el objetivo O



Modelo de agente

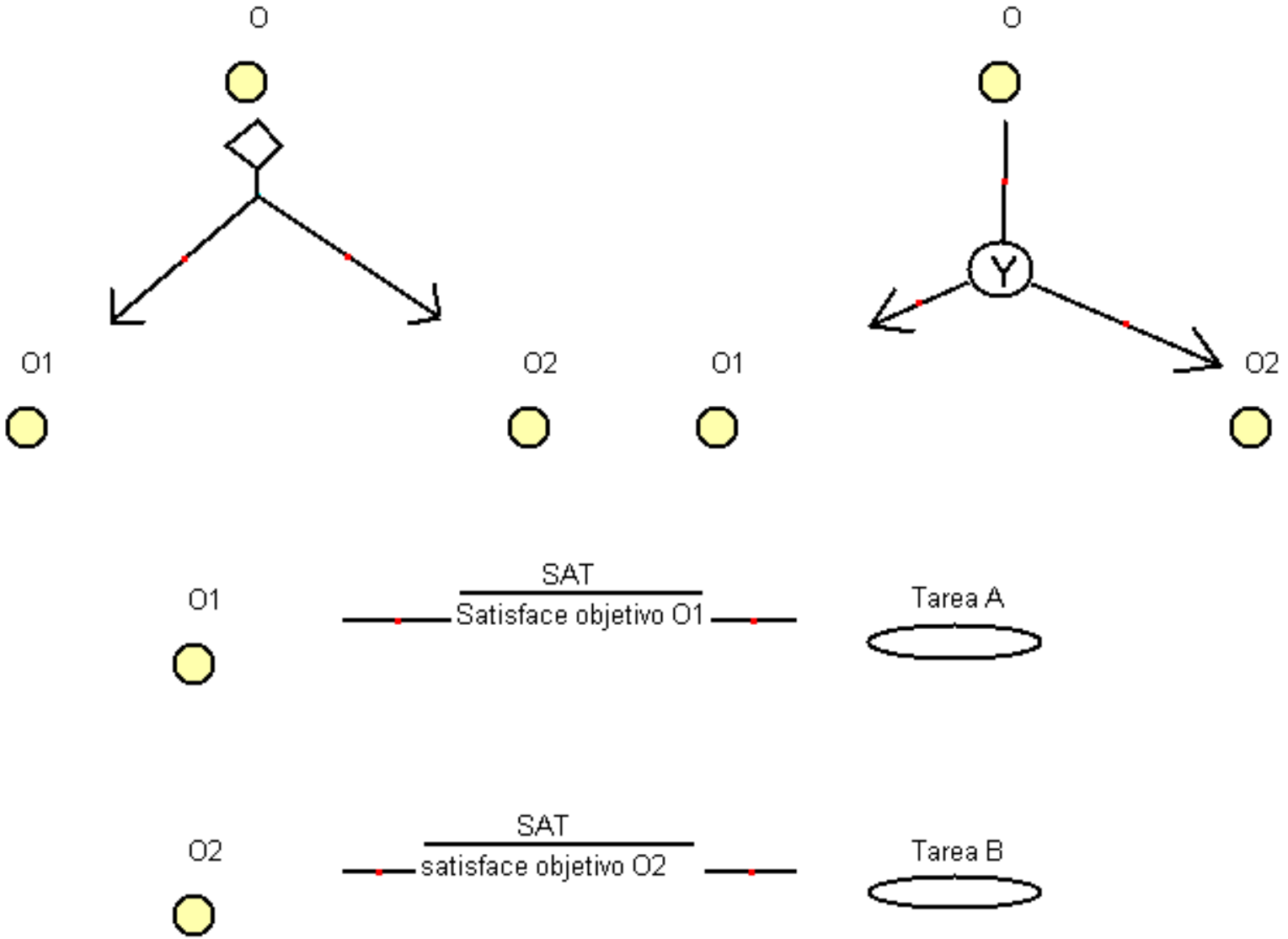
Un ejemplo (y II)

Las tareas simplemente generan hechos nuevos



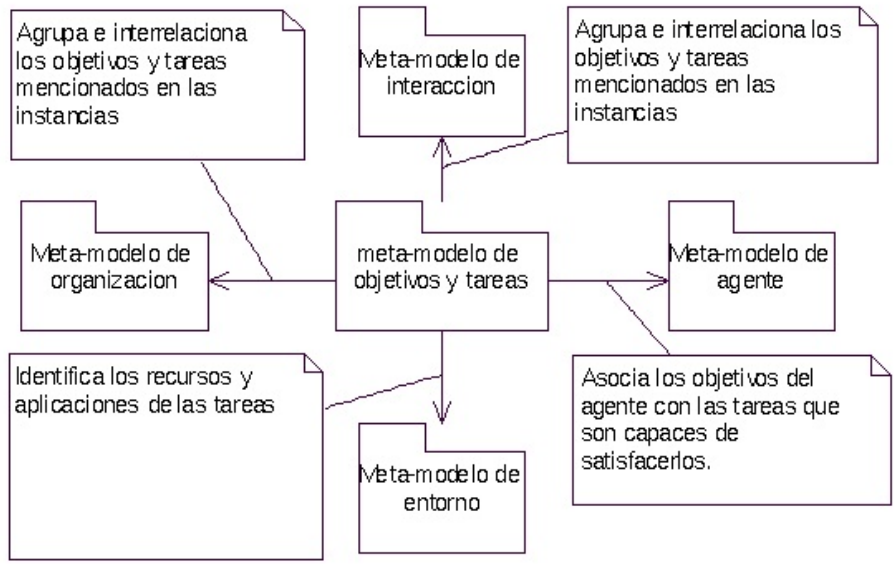
Modelo de tareas

Para alcanzar el objetivo O, se ejecutan A y B



Interdependencias con otros modelos

- Todas las tareas que aparecen en un modelo de tareas y objetivos también deben aparecer en algún modelo de agente o en algún modelo de organización.
- Todo objetivo que aparezca en un modelo de tareas y objetivos debe aparecer en un modelo de agente o en un modelo de organización.
- Si una tarea produce una interacción, debe existir un modelo de organización donde se enmarque esta tarea dentro de un flujo de trabajo
- Cuando los resultados de una tarea se necesiten en otra, se entiende que se tiene un flujo de trabajo. Por lo tanto, debe crearse una nueva entidad flujo de trabajo en un modelo de organización y especificar allí cómo se conectan las tareas.
- Todo recurso que aparece en este modelo debe aparecer en un modelo de entorno.
- Las entidades mentales consumidas, producidas, modificadas o destruidas deben pertenecer al estado mental del agente ejecutor



Las Interacciones en INGENIAS

¿Qué es una interacción?

Es una especificación sobre conversaciones que pueden mantener un grupo de agentes o un agente con un elemento del entorno, destinada a la realización de una tarea concreta.

A destacar sobre las interacciones

- No se tratan de la misma forma en todas las metodologías (GAIA, MAS-CommonKADS, etc)
- Independientemente de la metodología, disponemos de estándares tipo KQML o ACL-FIPA
- INGENIAS modela las interacciones a más alto nivel, pudiéndose instanciar posteriormente a ACLs concretos

La especificación de una interacción debe cubrir

las siguientes informaciones

- Los actores que participan (mostrar por qué está participando en la interacción, i.e. agente racional)
- La definición de unidades de interacción (e.g. puede ser tan simple como un paso de mensaje o tan compleja como un mensaje activo)
- Un orden sobre las unidades de interacción (i.e. un protocolo estándar)
- Acciones ejecutadas en la interacción mediante
  - Criterios para decidir cuándo ejecutar una tarea (no es suficiente con que alguien lo solicite)
  - Consecuencias de la ejecución de una tarea (se esperan cambios en el estado mental del agente)

Materialización de interacciones dependiendo de la tecnología subyacente

Al definir las unidades de interacción

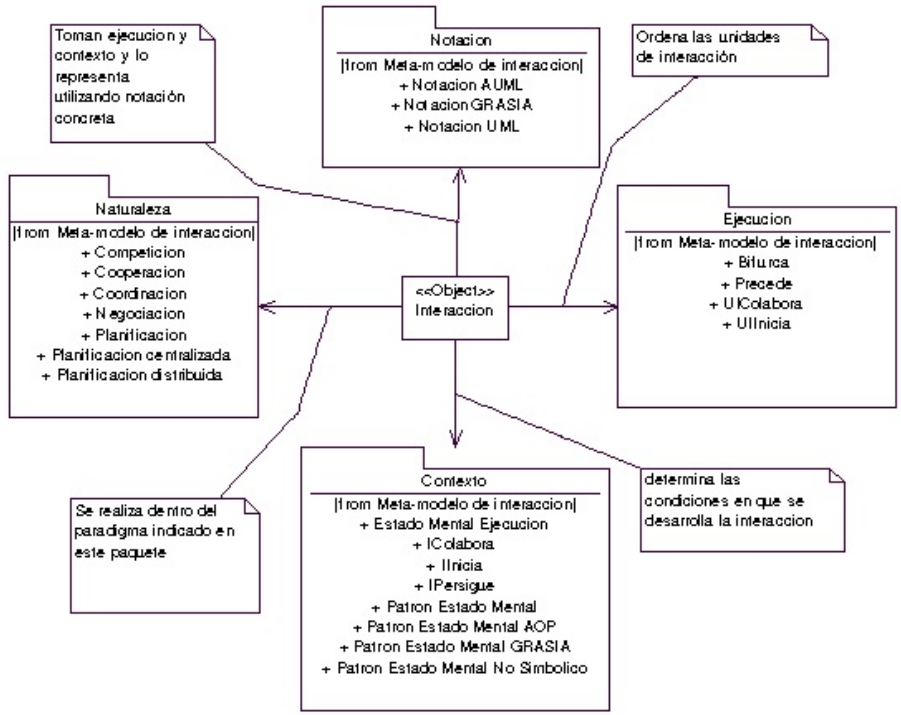
- Si usamos mensajes, se definen el número y contenido de sus parámetros
- Si usamos objetos, definimos tipo y valor de los argumentos de la llamada
- Si tenemos un espacio de tuplas (e.g. Linda), qué información se debe dejar en el espacio compartido y cómo esta información debe ser leída por los colaboradores

Al construir los protocolos

- Arquitectura de pizarra
- En la práctica se usa más el modelo FIPA (paso asíncrono de mensajes ACL)
- Paradigmas cliente servidor (CORBA, DCOM o RMI)

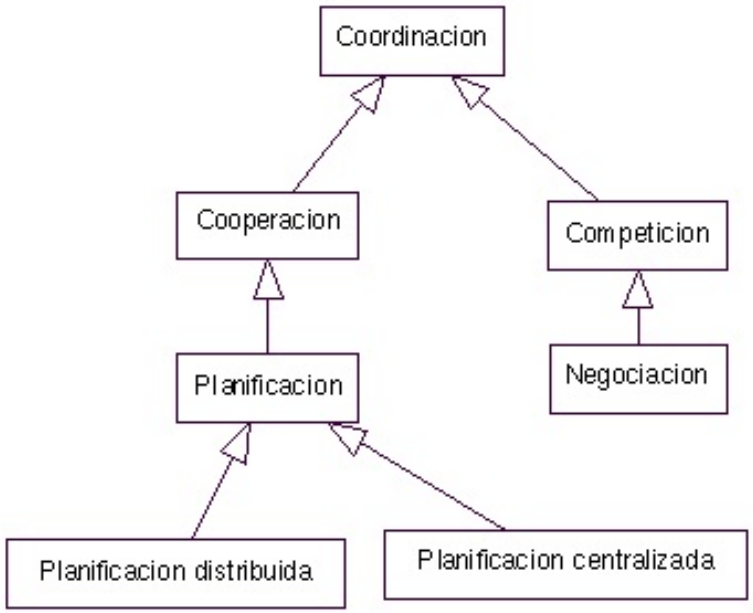
Panorama general

- Se construye sobre agentes, roles, objetivos, interacciones y unidades de interacción
- Los agentes y roles son los actores de las interacciones
- En las interacciones se ejecutan unidades de interacción (pasos de mensaje, lectura y escritura en un espacio de tuplas)
- Hay un iniciador y colaboradores
- La participación de los actores en la interacción y la existencia de la interacción en sí se justifica mediante objetivos



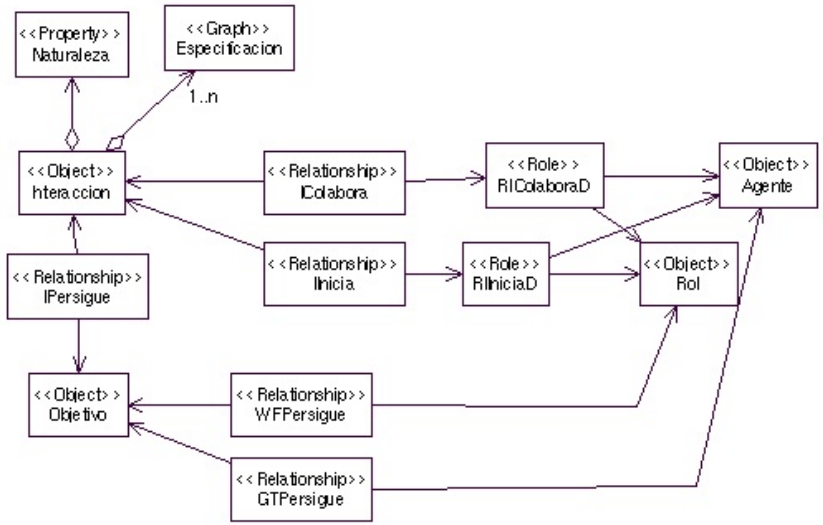
Naturaleza de las interacciones

- La naturaleza de la interacción define qué proceso coordinador se está llevando a cabo entre los agentes participantes
- Según su naturaleza, deberán aparecer unos elementos u otros en el modelo (e.g. si se está negociando, el bien sobre el que se negocia)



Meta-modelo de interacción

- La relación `IPersigue` representa la motivación de la interacción
- Los actores se indican mediante `WFPersigue` y `GTPersigue`
- La naturaleza se incluye como una propiedad y la especificación es un conjunto de relaciones, objetos y roles (i.e. tipo `Graph`)



Análisis en INGENIAS

Un ejemplo ilustrativo basado en un sistema recomendador de documentos

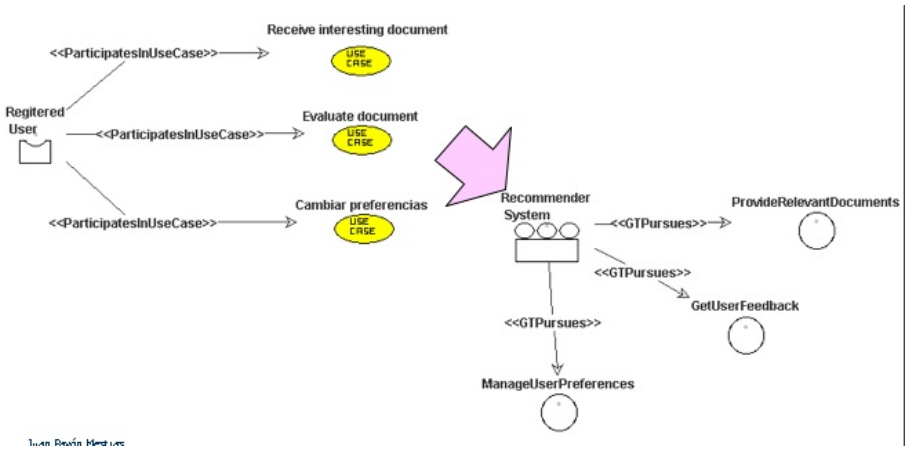
- Los documentos se evalúan según los gustos de los usuarios
- Los documentos llegan a la comunidad de usuarios desde fuera y se evalúan

En una primera tarea de análisis identificaremos requisitos mediante casos de uso y luego pasamos al diseño de

- Objetivos, identificados por los requisitos
- Tareas, que son procedimientos para satisfacer objetivos
- Roles, que definen servicios determinados por las tareas y responsabilidades
- Asignamos objetivos de la organización a los roles definidos
- Definimos workflows: relaciones entre tareas, roles y recursos
- Interacciones, para modelar cómo se comunican los roles
- Agentes, que desempeñan unos roles determinados

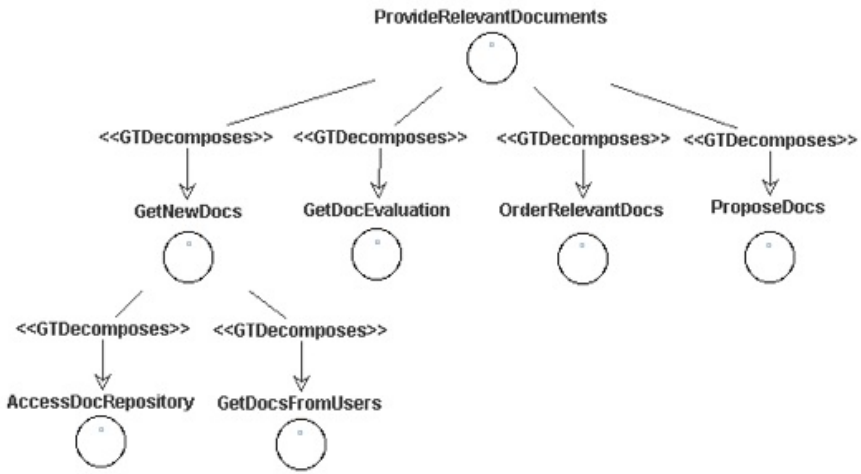
De los casos de uso a los objetivos en la organización

- Los casos de uso se emplean como usualmente, los actores serán roles posteriormente
- El sistema multi-agente es una organización
- Por cada caso de uso se organizan objetivos a cumplir en la organización que es el SMA



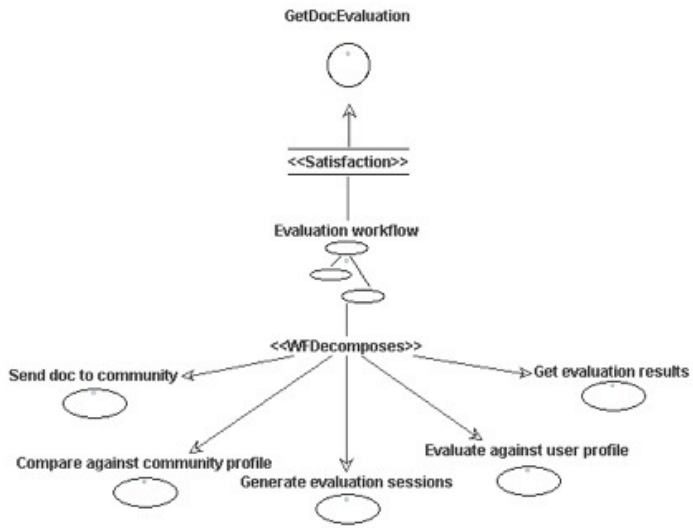
Modelado de objetivos jerárquicamente

- Los objetivos se descomponen hasta que son realizables mediante una tarea
- Así, cada objetivo más pequeño tendrá una tarea asociada dentro del modelo (o un flujo de tareas como en este caso)



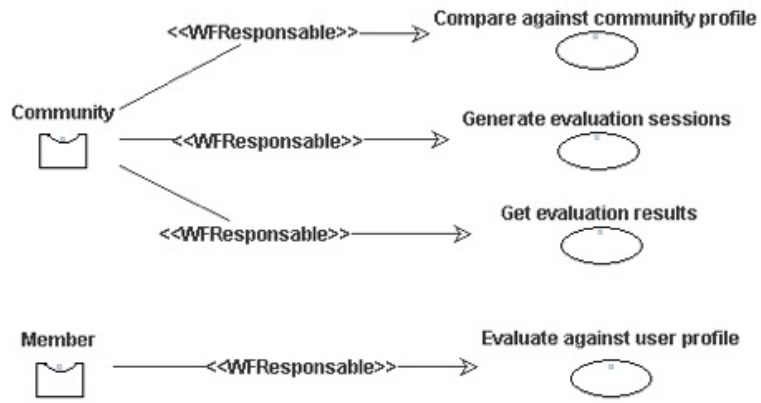
Modelo de objetivos para satisfacer tareas

- Un objetivo se puede satisfacer mediante un flujo de tareas complejo, o bien directamente por una tarea
- Un workflow no es más que la descomposición de una tarea en tareas más simples
- En el workflow se deberá indicar cómo se van modificando los datos de tarea en tarea



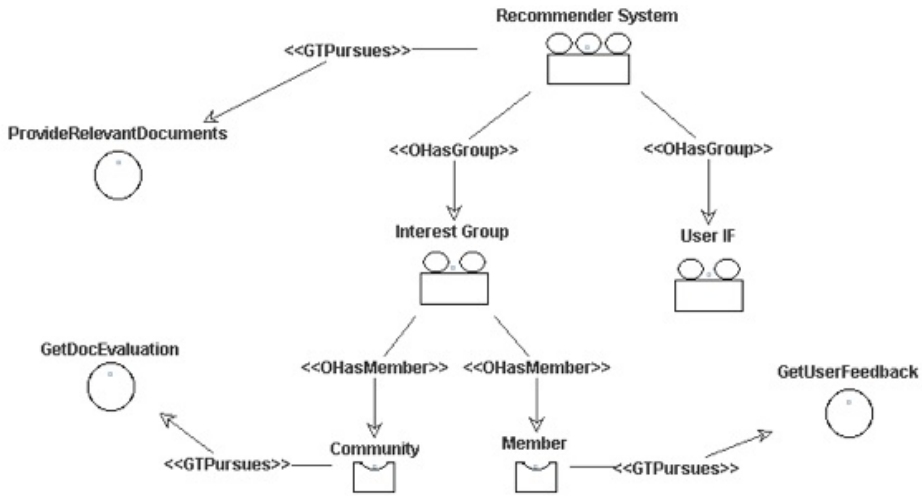
Asignación de roles a las tareas

- Un rol puede verse como un conjunto de servicios relacionados, junto con un conjunto de responsibilidades
- Indicamos los servicios de esta forma



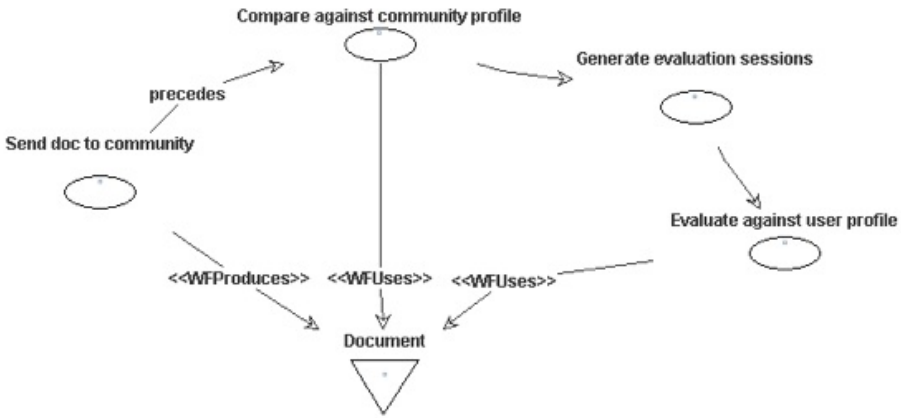
Asignación de roles a objetivos, dentro de la organización

- En el modelado puede resultar útil crear grupos dentro de la organización
- Luego los grupos tendrán sus respectivos roles
- También los roles perseguirán objetivos, mediante el desempeño de tareas



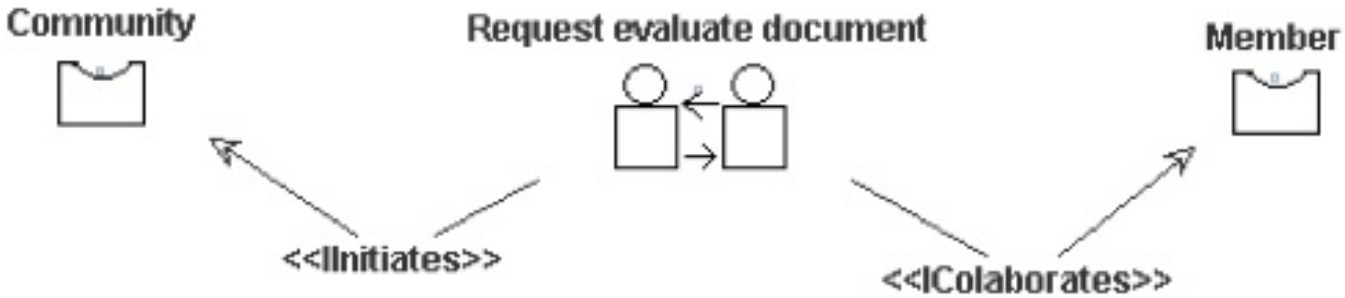
Workflows

- Definen relaciones entre tareas, roles y recursos
- En este ejemplo tenemos un recursos y varias tareas que se disponen a modo de flujo
- También podemos indicar cómo evolucionan los datos de una tarea a otra, qué genera y consume cada tarea



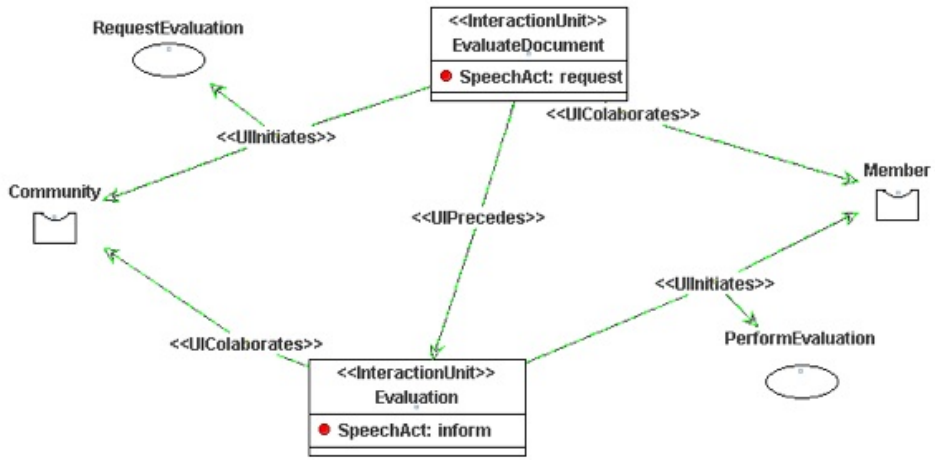
Las interacciones en el ejemplo

- En un modelo INGENIAS, las interacciones definen el procedimiento mediante el cual los roles interaccionan (i.e. negocian, cooperan, etc.)
- Uno de los roles inicia la conversación (`IInitiates`) y el otro u otros responden (*IColaborates*) y siguen interaccionando (si procede)
- La naturaleza se incluye como una propiedad y la especificación es un conjunto de relaciones, objetos y roles (i.e. tipo `Graph`)



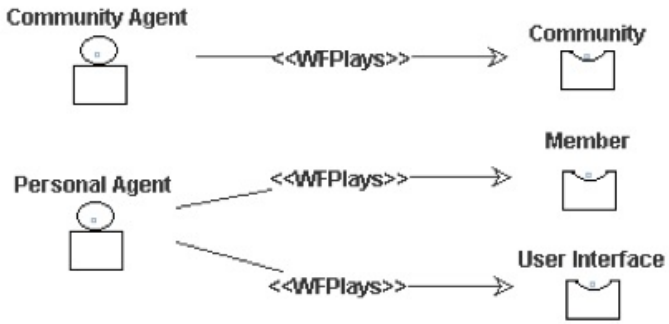
Luego hay que definir cada interaccion por separado

- Una unidad de interacción es una comunicación puntual dentro del diálogo (i.e. en FIPA es un acto comunicativo)
- Cada unidad de interacción es generada por un rol, recibida por otro y quizás, motivada por una tarea
- Además, existe un orden (`UIPrecedes`) entre las unidades de interacción
- Podemos usar modelado AUML



Un modelo de agente simple

- Los agentes aglutinan roles que desempeñan concurrentemente
- Con el IDK veremos que hay varias maneras de implementar agentes
- Cada uno de ellos puede tener un conjunto de creencias inicial



En este caso, el diseño era dirigido por objetivos

Hay otras posibilidades, dependiendo del problema

- Centrarse y comenzar con los workflows
  - Cuando la organización está orientada a procesos
- Centrarse en la coordinación y las interacciones
  - Cuando el problema es la definición de un algoritmo distribuido
  - Sistemas cooperativos
- Centrarse en el entorno
  - Sistemas empotrados
  - Robótica
- Centrarse en los actores
  - Simulación social


Jose Cuena.
Lógica Informática.
Alianza Editorial, 1985.


A. Rao and M. Georgeff.
Deliberation and intentions.
In *Proceedings of 7th Conference on Uncertainity in Artificial Intelligence*, Los Angeles, 1991.


Anand S. Rao and Michael P. Georgeff.
Modeling rational agents within a BDI-architecture.
In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991.


Anand S. Rao and Michael P. Georgeff.
Decision procedures for bdi logics.
Journal of Logic and Computation 1998 8(3):293-343, 8(3):293–343, 1998.