UNIVERSIDAD
DE LOS ANDES
MERIDA VENEZUELA

Inteligencia Artificial Distribuida

La IAD

- En diferentes dominios se han hecho esfuerzos por plantearse formas para la **solución de problemas cooperativamente.**

- La Inteligencia Artificial Distribuida (IAD) es uno de los dominios que ha estudiado ese problema, en este caso, usando conceptos provenientes de:
  - La Inteligencia Artificial (IA)
  - Los Sistemas Distribuidos.

Por qué Distribuir a la IA?

- **Por las Ciencias humanas: capacidades intelectuales proviene**
  - disposiciones genéticas, interacciones con los semejantes y ambiente
- **Por la psicología: ser humano**
  - opinión o punto de vista sobre una cosa
- **Resolución de problemas**
  - problemas funcionalmente o espacialmente distribuidos y complejos
- **Industria**
  - trabajar con varios especialistas con puntos de vista diferentes
- **Informática**
  - extensión noción (multi)objetos

Generalidades

- La IAD involucra *comunidades de agentes* donde los agentes resuelven (sub)tareas

- **Áreas principales:**
  - Sistemas Multi-agentes (SMA)
  - Resolución Distribuida de Problemas (RDP)

- **La IAD tiene que ver con:**
  - Granularidad de los agentes
  - Heterogeneidad de los agentes
  - Métodos para distribuir las tareas, el control, etc., entre los agentes
  - Protocolos de comunicación entre los agentes

- **La IAD no tiene que ver con:**
  - Coordinación de procesos concurrentes
  - Arquitecturas paralelas, Lenguajes de programación paralela, Sistemas operativos distribuidos

La IAD

Técnicas de resolución de problemas que mezclan tanto la IA como los Sistemas Distribuidos.

- Los sistemas basados en IAD ***integran al menos dos agentes***, los cuales interactúan para la resolución de un problema dado.

- Los agentes son frecuentemente heterogéneos y deben tener cierto grado de ***autonomía, capacidades de razonamiento, planificación*, etc.**

- Los agentes deben ser capaces de ***interactuar, negociar, cooperar* y hasta *competir*** con otros para llevar a cabo sus tareas, las cuales pueden ser individuales o grupales.

La IAD

Objetivos IAD

Desarrollar mecanismos y métodos que permiten a los agentes interactuar, así como lo hacen los seres humanos (o incluso mejor),
comprender la interacción entre entidades inteligentes, ya sean computacionales, humanas, o ambas.

Estos objetivos plantean una serie de retos que se centran en torno a la cuestión elemental de:

cuándo y cómo interactuar y con quién.

La IAD

Dos razones principales de la IAD

1. Tiene un papel clave en las plataformas informáticas y entornos de información actuales: distribuidos, amplios, abiertos y heterogéneos.
   - Las computadoras están clásicamente conectadas.
   - La creciente complejidad de los sistemas informáticos y de información a menudo excede el nivel de la computación convencional, centralizada, ya que requieren, por ejemplo, el procesamiento de grandes cantidades de datos, o los datos están en lugares geográficamente distintas.

- Para hacer frente a estas situaciones, las computadoras tienen que actuar más como "individuos" o "agentes", en lugar de sólo "partes".

IAD proporciona una gestión de la interacción de alto nivel requeridas para las aplicaciones computacionales complejas modernas.

La IAD

Dos razones principales de la IAD

2. IAD tiene la capacidad de desempeñar un papel importante en el desarrollo y análisis de modelos y teorías de la interactividad en las sociedades humanas.
   - Los seres humanos interactúan de diversas maneras y en muchos niveles: por ejemplo, observan y modelan unos a otros, solicitan y proporcionan información, negocian y discuten, desarrollan puntos de vista compartidos de su entorno, detectan y resuelven conflictos, y se forman y se disuelven estructuras organizacionales, tales como equipos, comités y economías.
   - Muchos procesos interactivos entre los seres humanos son todavía poco conocidos, a pesar de que son una parte integral de nuestra vida cotidiana.

IAD permite explorar fundamentos sociológicos y psicológicos

La IAD

El papel que el concepto de SMAs juega en IAD es comparable al papel que el concepto de agente desempeña en la IA.

Tanto IAD e IA consideran aspectos computacionales de la inteligencia, pero lo hacen desde diferentes puntos de vista y bajo diferentes supuestos:

- IA se centra en agentes como "**sistemas autónomos inteligentes**" y la inteligencia como una propiedad del sistema que actúa aisladamente, la IAD se centra en agentes como *"**sistemas inteligentes conectados** " y la inteligencia como una propiedad de los sistemas que interactúan*.
- IA se centra en los "**procesos cognitivos**" de los individuos, la IAD se centra en los *"**procesos sociales**"* de grupos de individuos.

La IAD

El papel que el concepto de SMAs juega en IAD es comparable al papel que el concepto de agente desempeña en la IA.

Tanto IAD e IA consideran aspectos computacionales de la inteligencia, pero lo hacen desde diferentes puntos de vista y bajo diferentes supuestos:

- IA considera que los sistemas tienen un único foco de ***razonamiento y de control interno*** y requieren sólo una ayuda mínima de otros para actuar, la IAD considera sistemas en los que se ***distribuye el razonamiento, el control y las actividades***.
- IA usa ideas, inspiraciones y metáforas de la ***psicología y el conductismo*** , la IAD utiliza *la* ***sociología y la economía***.

IAD no es una especialización de la IA, sino una generalización de la misma.

La IAD

IAD como sistema distribuido, tiene las siguientes propiedades:

- **Velocidad y eficiencia** - Los agentes pueden funcionar de forma asíncrona y en paralelo, aumentando la velocidad en general (si la sobrecarga por la coordinación no penalice esa ganancia).
- **Robustez**- El fallo de un o varios agentes no significa necesariamente que el sistema en general quede inútil, ya que otros agentes disponibles en el sistema pueden hacerse cargo de su parte.
- **Escalabilidad y flexibilidad** - El sistema puede soportar el aumento del tamaño del problema mediante la adición de nuevos agentes, y esto no necesariamente afecta la operatividad del sistema.
- **Costos** - puede ser mucho más rentable que un sistema centralizado, ya que podría estar compuesto por subsistemas simples de bajo costo unitario.

La IAD

IAD como sistema distribuido, tiene las siguientes propiedades:

- **Desarrollo y reutilización** - agentes individuales pueden ser desarrollados por separado por especialistas (ya sea desde cero o basado en cosas ya está disponibles), el sistema global se puede probar y mantener con más facilidad, y se puede reconfigurar y reutilizar agentes en diferentes escenarios de aplicación.

- **La tecnología informática y de red disponible constituye una plataforma sólida para la realización de estos sistemas**. En particular, la evolución reciente de la computación paralela y distribuida, la computación móvil, etc.

IA Distribuída (IAD)

- No es IA paralelo o Sist.Distribuídos.
- Problema se resuelve mediante la cooperación (o colaboración).
- Grandes habilidades de interacción y comunicación.
- Organización: significa garantizar la convergencia:
- Ls estructuras de autoridad y control dividido.
- División de conocimientos y recursos.

Generalidades

- **En IAD:**
  - No es obligatorio pensar en un *control* central; el control puede ser distribuido
  - El *conocimiento* o las *fuente de información* pueden también estar distribuidos

- **Un RDP** considera la tarea de resolver un problema en particular, para lo cual **lo descompone en módulos** (división de tareas) que serán resueltos cooperativamente por un grupo de agentes.

- **Un SMA** tiene que ver con el **comportamiento de una comunidad de agentes autónomos tratando de resolver problemas**, por lo cual comparten conocimiento acerca de los problemas y sus soluciones.

RDP

Es un subcampo de la inteligencia artificial distribuida (IAD) en el que el énfasis está en conseguir que los agentes trabajen juntos para resolver problemas que requieren un esfuerzo colectivo.

Debido a una distribución inherente de los recursos como el conocimiento, la capacidad, la información y experiencia entre los agentes, un agente en es incapaz de cumplir sus propias tareas por sí solo.

RDP requiere

- **Coherencia** (es decir, los agentes necesitan querer trabajar juntos): se asume un cierto grado de coherencia, ya que los agentes son diseñados para trabajar juntos; o la ingeniería social introduce desincentivos para el individualismo de los agentes; etc.
- **Competencia** (es decir, los agentes necesitan saber cómo trabajar juntos): La tarea consiste en como distribuir las competencias; como cualquiera que haya jugado en un equipo, trabajado en un proyecto de grupo, o participado en una orquesta musical,

Simplemente tener el deseo de trabajar juntos no asegura un resultado colectivo exitoso

RDP

- Presume la **existencia de problemas** que necesitan ser resueltos y **expectativas acerca** de lo que constituyen las **soluciones**.
  - Por ejemplo, un problema a resolver podría ser para un equipo de agentes diseñar un artefacto (por ejemplo, un coche).
  - La solución que formulen deberá cumplir los requisitos generales (debe tener cuatro ruedas, el motor debe encajar dentro del compartimiento del motor y ser potente como para mover el coche, etc.),
  - Debe existir en una forma particular (documento de especificaciones para el montaje del carro).

- Los agentes **formulan soluciones** por cada subproblema (uno o más) y **sintetizan estas soluciones** de los subproblema en soluciones globales.

RDP

A veces el problema de los agentes es construir un plan.

- Incluso, si los agentes están resolviendo otro tipo de problemas, tienen que resolver problemas de planificación también. Es decir, los agentes deben planear cómo trabajar juntos:
    1. Descomponer problemas en subproblemas,
    2. Asignar estos subproblemas,
    3. Intercambiar soluciones de los sub-problemas, y
    4. Sintetizar soluciones globales-

Son en sí un problema que los agentes tienen que resolver!!

Planificación distribuida está así estrechamente entrelazada con la resolución de problemas distribuida, siendo a la vez un problema en sí mismo y un medio para resolver un problema.

RDP

Motivaciones para la resolución de problemas distribuida

Los conocimientos o capacidades de resolución de problemas se pueden distribuir

Por ejemplo, en la **ingeniería concurrente**, un problema podría implicar el diseño y fabricación de un artefacto (por ejemplo, un coche) con agentes especializados que hagan individualmente componentes y procesos, y la combinación de éstos dan una solución colectiva.

- Puede implicar i**nteracciones** para monitorear los eventos, evaluar la situación, generar la respuestas, etc.
- El problema es emplear diversas capacidades para **resolver problemas** que no sólo son **de gran tamaño**, pero también **multifacéticos**.

  Ejemplo, una **red de sensores distribuido** en un estacionamiento para monitorizar los movimientos de los vehículos.

- La tarea general de vigilancia no se puede hacer desde una única ubicación.
- El problema es descomponer la tarea de monitoreo en subtareas que se pueden asignar adecuadamente a agentes distribuidos geográficamente.

RDP

Motivaciones para la resolución de problemas distribuida.

El uso de los recursos distribuidos simultáneamente puede permitir una aceleración de la resolución de problemas gracias a paralelismo.

Las posibles mejoras debido al paralelismo dependen, por supuesto, en el grado de paralelismo inherente en un problema.

Los datos pueden ser distribuidos.

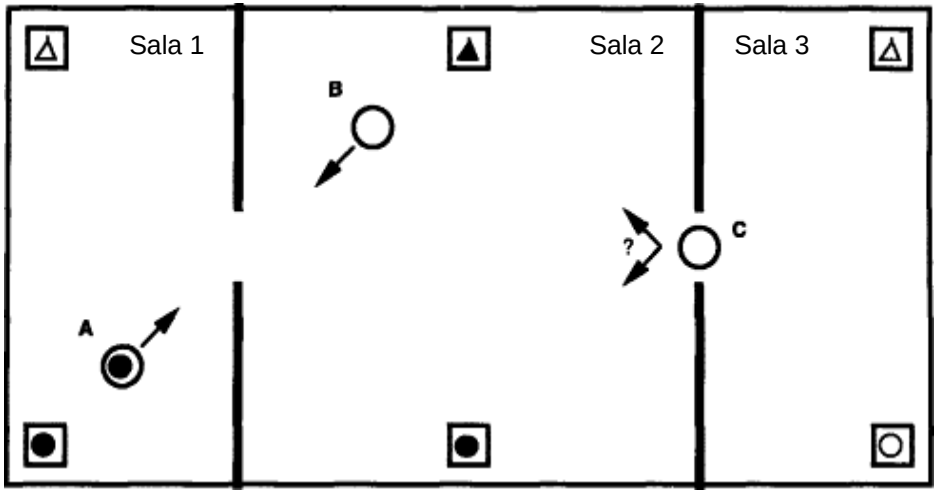
- Para el ejemplo de la red de sensores distribuidos, hacer el seguimiento de vehículos distribuido en principio, podría ser centralizado: cada uno de los agentes de sensores distribuidos podría transmitir los datos en bruto a un sitio central para su interpretación.
- Esta estrategia podría implicar enormes cantidades de comunicación innecesaria en comparación si se permite a los agentes sensores interpretar localmente qué debería ser transmitido de manera selectiva.

RDP

Motivaciones para la resolución de problemas distribuida.

Los resultados de la resolución de problemas o la planificación son distribuidos.

- Por ejemplo, en una tarea que implica la entrega de objetos en diferentes ubicaciones, agentes de entrega distribuidos pueden actuar en paralelo.
- La formulación de planes se podría hacer en un sitio centralizado o distribuirse.
- Durante la ejecución de sus planes, características del ambiente que no se conoce en tiempo de planificación, o que inesperadamente cambian, puede desencadenar cambios en lo que los agentes deben hacer.



Todas estas decisiones podrían ser realizadas por un coordinador central, pero por diversas razones (**explotar el paralelismo, la poca disponibilidad del coordinador, los canales de comunicación lentos**, etc.) podría ser preferible para los agentes modificar sus planes de manera unilateral o con comunicación limitada entre ellos

RDP

Estrategias de resolución de problemas distribuidos

Reparto de tareas: Cuando un agente tiene muchas tareas que hacer, se debe contar con la ayuda de agentes con pocas o ninguna tarea. Los pasos principales en el reparto de tareas son:

- *Descomposición de tareas*.
- *Asignación de tareas*.
- *Logro de Tareas* (puede implicar recursividad.
- *Síntesis de Resultado*:(solución global).

Los diferentes pasos pueden ser más o menos difíciles. Por ejemplo,

- A veces un agente sobrecargado comienza con un conjunto de tareas separadas, por lo que la descomposición es innecesaria;
- A veces, el agente puede pasar tareas a una serie de agentes idénticos, por lo que la asignación es trivial; y,
- A veces las tareas no dan ningún resultado que necesite ser sintetizado

RDP

Un problema clásico de la IA que muestra lo anterior es el problema de la Torre de Hanoi (ToH)

- ToH: El objetivo es mover los discos de la clavija de inicio a otra clavija ,moviendo sólo un disco a la vez, sin poder colocar un disco más grande en la parte superior de un disco más pequeño. El problema es encontrar una secuencia de movimientos que permitan alcanzar el estado final.

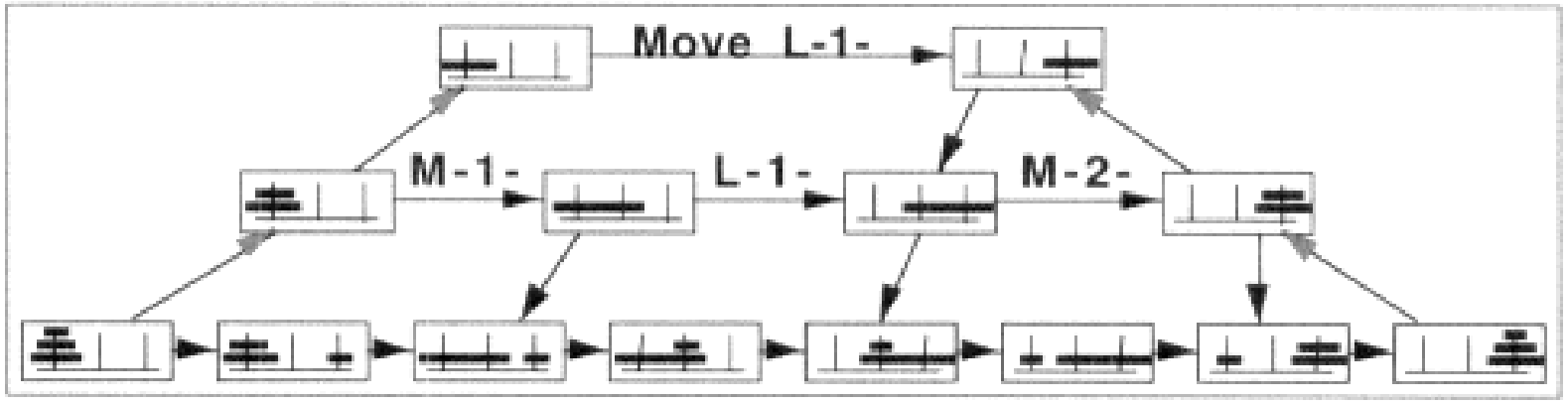
RDP

Reparto de tareas en ese problema

1. *Tarea de descomposición*: mover el disco más grande que no está en su clavija de destino conduce a una descomposición recursiva: resolver el problema de llegar al estado en el que más grande de disco se puede mover. Este subproblema se puede descomponer aún más en el problema de mover el segundo disco mayor, etc.
2. *Asignación de tareas:* Si suponemos un número indeterminado de agentes desocupados idénticos capaces de resolver el problema, la asignación se reduce a sólo asignar una tarea al azar a uno de estos agentes.
3. *Logro de Tareas*: En general, un agente debe encontrar la diferencia más significativa entre los estados de inicio y el objetivo y descomponer el problema en base a éstos. Si los estados de inicio y el objetivo son los mismos, entonces la descomposición recursiva termina.
4. *Síntesis de los Resultados*: Cuando un agente ha solucionado su problema, pasa la solución de nuevo hacia arriba. Cuando un agente ha recibido soluciones a todos los subproblemas que paso hacia abajo, los puede componer en una secuencia más completa de movimientos, y luego pasar esto como su solución.

RDP

Reparto de tareas en ese problema



RDP

Reparto de tareas en sistemas heterogéneos

Una de las motivaciones para la RDP es que es difícil construir artefactos para ser competente en cada tarea posible. Además, incluso si es factible construir un agente capaz a menudo es excesivo porque la mayoría de esas capacidades se desperdicia.

- La estrategia en los sistemas humanos es **combinar especialistas** en diferentes áreas para resolver problemas.
- Es posible que un agente tenga una **tabla que identifica las capacidades de los agentes**, por lo que simplemente selecciona un agente apropiado y envía el subproblema

RDP

Reparto de tareas en sistemas heterogéneos

Pero por lo general la información es más dinámica. Por ejemplo, si hay varios agentes candidatos , algunos ya se han comprometido,

Cómo se descubre eso? Una forma es utilizar el protocolo Net-Contrat

- Un gerente difunde su anuncio para llegar a los agentes de que es actualmente desconocen también.
- Una estrategia muy simple es volver a intentar el anuncio periódicamente, en el supuesto de que, finalmente, un contratista va a liberar.
- Podría ser que el gerente estaba siendo demasiado exclusiva en los que sería entretener a las ofertas de. Por lo tanto, el gerente podría participar en revisión iterativa de su anuncio, se relajan los requisitos de elegibilidad hasta que comience a recibir ofertas.
- El administrador puede intentar descomponer el problema de tal manera que los contratistas están disponibles para los subproblemas alternativas.

RDP

Compartiendo Resultado

Los resultados de la tarea de un solucionador del problema podría diferir de los resultados de la misma tarea por otro solucionador de problemas. Por ejemplo, los alumnos de una clase tienen a menudo la misma tarea (problema), pero sus soluciones no lo son (no deberían ser idénticas). Al compartir los resultados, se puede mejorar el desempeño del grupo:

- **Confianza:** generación independientemente de resultados para la misma tarea pueden utilizarse para corroborar el uno al otro, produciendo un resultado colectivo con mayor confianza de ser correcta.
- **Integridad:** Cada agente formula resultado para la subtarea que puede cumplir, y estos resultados cubren por completo una parte más completa de la tarea global. .

RDP

Compartiendo Resultado

- **Precisión:** Para refinar su propia solución, un agente necesita saber más acerca de las soluciones que otros han formulado para mejorar su propia respuesta.
- **Puntualidad:** Incluso si un agente podría resolver, en principio, una gran tarea por sí sola, la solución de subtareas en paralelo puede dar una solución global más rápido.

Pero hacer este trabajo es más difícil de lo que piensas!:

- Los agentes deben saber qué hacer con los resultados compartidos:
- Los agentes deben tratar de ser lo más selectivos posible acerca de lo que intercambien porque
  - la comunicación de grandes volúmenes de resultados puede ser costoso,
  - la gestión de muchos resultados incurre en gastos , etc.

RDP

Compartiendo Resultado: Repositorios compartidos y búsqueda negociada

Una de las estrategias para reducir el potencial aluvión de mensajes de multidifusión es en concentrar resultados parciales en un único repositorio compartido.

Arquitectura de pizarra

En esencia, el uso de un repositorio compartido puede soportar estrategias de búsquedas alternativas.

Por ejemplo, una ***estrategia de búsqueda negociada*** para la RDP sería:

- Iniciar solución (proponer un punto de partida para una solución);
- Extender-solución (revisar una solución parcial ya existente);
- Criticar-solución (dar feedback sobre la viabilidad de una solución parcial ya existente);
- Relajar-requisito -solución (cambiar los requisitos locales para aumentar aceptabilidad de la solución).

Planificación Distribuida

Puede pensarse como una especialización de la RDP, donde el problema es alrededor de un plan y su ejecución .

- La planificación distribuida es un término ambiguo, porque no está claro lo que se "distribuye".
  - **El plan** está formulado y se distribuye su ejecución.
  - **El proceso** de planificación es distribuido.
  - **Ambos** son distribuidos.

no se considera el caso donde ninguno lo es

(ya que es la planificación centralizada tradicional)

Planificación Distribuida

- Planificación Centralizada para Planes Distribuidos
- Planificación Distribuida para Planes Centralizados
- Planificación Distribuida para Planes Distribuidos

Planificación Distribuida para Planes Centralizados

La formulación de un plan complejo podría requerir de la colaboración de varios especialistas, al igual que generar la solución a cualquier problema complejo.

- Por lo tanto, el proceso de planificación distribuida **podría ser realizado por varios agente**s, cada uno contribuye al plan, hasta crear un plan global.
- La formulación del problema puede ser pensado en **cómo descomponer la planificación y distribuirla entre varios especialistas**, cada uno genera partes del plan
- Tanto las **estrategias de compartir resultados o compartir tareas** se pueden usar en este caso.

Planificación Centralizada para Planes Distribuidos

Planes que se ejecutan de manera distribuida, no obstante se formulan de manera centralizada.

- Un **planificador de orden parcial** puede generar planes que no tienen un orden estricto.
- Esas acciones se pueden ejecutar en **paralelo**.
- Un **agente coordinador** centralizado un plan de este tipo y lo descompone en hilos separados, posiblemente con algunas acciones de sincronización.
- Estas **partes planas separadas se pueden pasar a los agentes** que pueden ejecutarlas.
- Los **agentes operan en paralelo** y logran alcanzar un estado del mundo consistente con los objetivos del plan.

Planificación Centralizada para Planes Distribuidos

- **Esquema:**
  - **Planificación en orden parcial** que puede ser ejecutada en paralelo (diferentes agentes hacen ramas diferentes).
  - Se debe garantizar no condiciones de dependencia.

- **Algoritmo:**
  1. Diseñar un plan general teniendo en cuenta una descripción meta, un conjunto de operadores y un inicial estado (el planificador)
  2. identificación ramificaciones que se pueden realizar en paralelo. (POP).
  3. Descomponer el plan en tales ramas-subplanes (ramificaciones)
  4. Insertar acciones de sincronización entre los subplanes
  5. Asignar subplanes a los agentes (asignacion (estatica o dinamica) a los agentes)
  6. Iniciar el plan de ejecución y seguimiento de los avances

Planificación Distribuida para Planes Centralizados

Formular planes requiere colaboración de especialistas

1. **Distribuir la tarea de construir planes** entre los especialistas
2. **Mezclar los planes** obtenidos en paralelos verificando la coherencia entre ellos

Debe haber una **coordinación centralizada para unir los planes** parciales

- CADA AGENTE HACE SU PLAN PARCIAL
- EL COORDINADOR FUSIONA LOS PLANES (RELACION ENTRE LAS ACCIONES)

Planificación Distribuida para Planes Distribuidos

La más desafiante en planificación es cuando se distribuye tanto el proceso de planificación como sus resultados.

- Podría ser innecesario tener un plan total en el sistema, y sin embargo, las partes del plan deben ser coherentes entre si

no debe haber conflictos entre sí cuando se ejecutan los planes

Pero además, se **pueden ayudar** para alcanzar sus Planes, unos a otros cuando sea racional

Planificación Distribuida para Planes Distribuidos

- **Caso General**
  1. Asignar objetivos a cada agente
  2. C/u formula planes locales
  3. C/u ejecuta sus planes

- Problemas
  - Resolución de Conflictos: **negociación**
  - Suposiciones de **tiempo de ejecución de tareas**

Planificación Distribuida para Planes Distribuidos

- **Otro Algoritmo:**
  1. **Intercambiar** entre agentes descripción de los planes
  2. **Aprobar** planes sin conflictos (si todos lo cumplen, resuelto el problema)
  3. **Refinar** planes con conflictos (cambios en los planes locales para eliminarlos)
  4. **Ejecutar localmente** planes sin conflicto

Planificación Distribuida para Planes Distribuidos

Fusión del Plan

- **Múltiples agentes formulan planes para individuos, y deben asegurar que sus planes se pueden ejecutar sin conflictos**.

- **El reto es identificar y resolver los posibles conflictos**. Hay varios enfoques:
  - **Coordinación plan centralizado**: un agente recoge planes individuales y los analiza para descubrir secuencias de acciones que podrían dar conflictos, y para modificarlos para eliminar los conflictos.
  - **Sincronizar durante la ejecución**, tal que se determina si el siguiente paso es seguro (no genera conflicto), siguiendo las ideas de estados seguros aplicados en evitar bloqueos de procesos

Planificación Distribuida para Planes Distribuidos

Plan de Formación iterativo

Es una fusión de gran alcance para el Incremento de paralelismo en el proceso de planificación, así como durante la ejecución.

- Los algoritmos de sincronización y programación pueden llevarse a cabo centralizada y descentralizada

- **El flujo es generalmente**
  1. Asignar objetivos a los agentes;
  2. Agentes locales formulan planes;
  3. Se cambian y combinan los planes locales;
  4. Se imponen compromisos para resolver interacciones negativas.

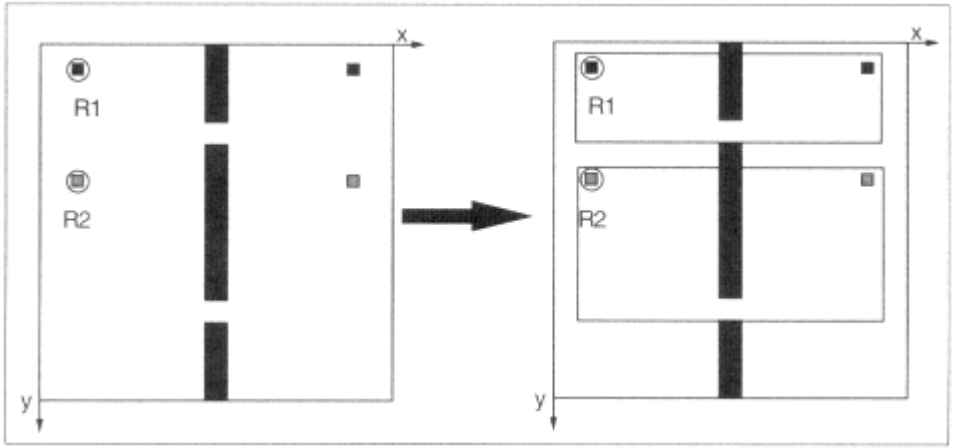
Planificación Distribuida para Planes Distribuidos

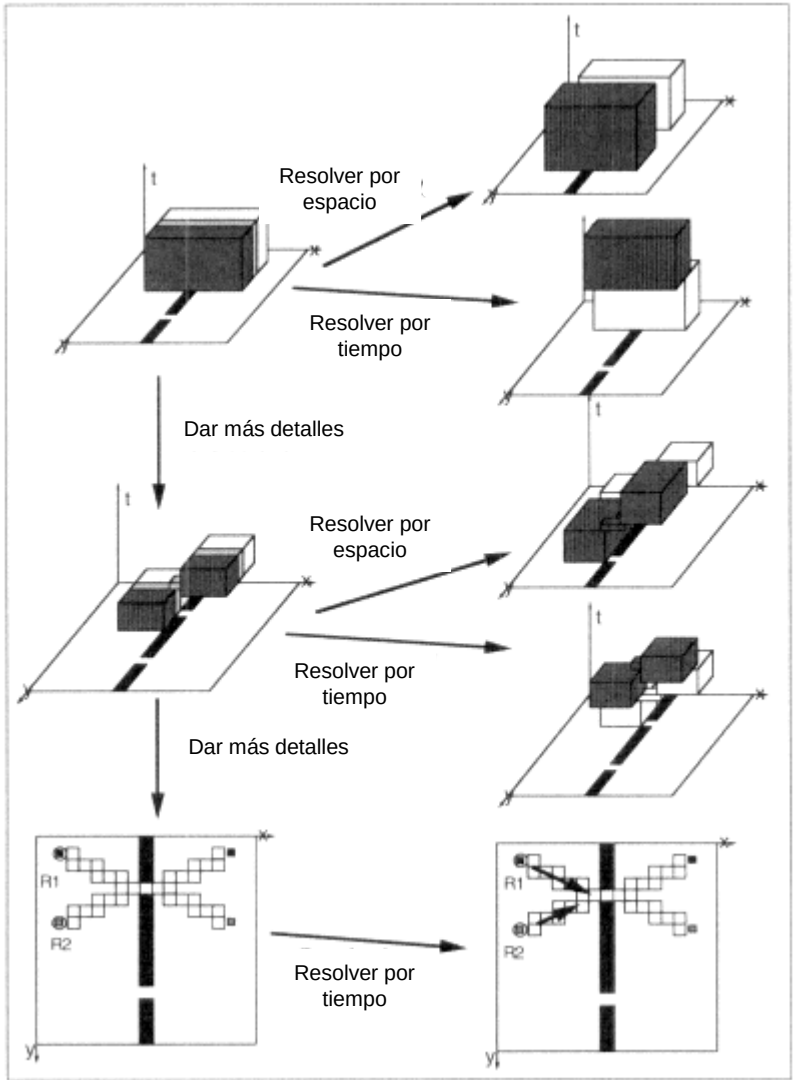
Planificación distribuida jerárquica

- Agente representa cada uno de sus planes locales **en múltiples niveles de abstracción**, para resolver todos los conflictos por cada nivel.
- **El bucle externo** del protocolo identifica un nivel de abstracción especial para trabajar, y si los conflictos deben resolverse a este nivel o se pasa a los niveles más detallados.
- **El bucle interno** del protocolo lleva a cabo lo que se puede considerar como una búsqueda de satisfacción de restricciones distribuido para resolver los conflictos.

1. *Inicialice el nivel de abstracción actual*
2. *Agentes intercambian descripciones* del plan y los objetivos de interés en el nivel actual
3. *Retire los planes sin conflictos*. Si el conjunto está vacío, entonces listo; De lo contrario, determinar si se debe resolver los conflictos en el nivel actual o en un nivel más profundo.
4. *Si se resuelven los conflictos en un nivel más profundo, especifique el nivel profundo y establezca los planes/objetivos que interesan*. Vaya al paso 2.
5. Si no, se procede a *resolver los conflictos en este* nivel

Planificación Distribuida para Planes Distribuidos





Planificación Distribuida y Ejecución

La planificación distribuida no ocurre en un vacío.

El plan distribuido debe ser ejecutado.

La **complejidad de coordinar planes**

- *Coordinación Post-Planificación*: El enfoque de planificación distribuida permite que los agentes planifiquen, pero luego debe hacerse la coordinación de la ejecución, y puede ser:
  - *Planificación de contingencia*. Cada agente que hizo plan responde a las contingencias que se pueden surgir en tiempo de ejecución. Esto implica planes más grandes con ramas condicionales coordinadas.
  - *Supervisión y la replanificación*: Un agente supervisa planes de ejecución y si hay una desviación detiene el progreso de todos los agentes, y regresa al ciclo de los planes donde la coordinación debe rehacerse.

Planificación Distribuida y Ejecución

Coordinación Pre-Planificación

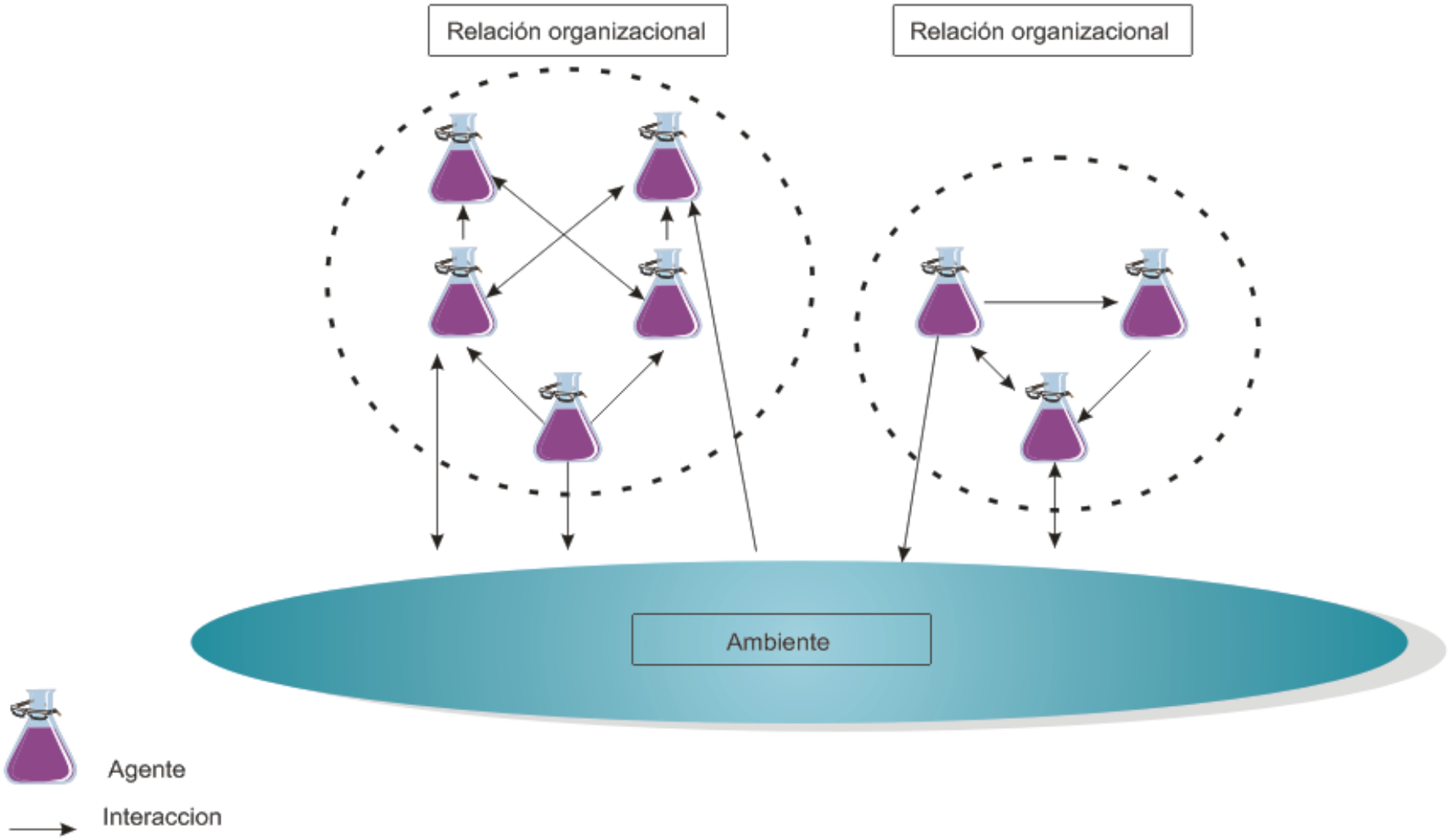
Antes de que un agente de planificación comience, el agente se coordinará con los demás (para definir restricciones a cumplir).

- Un ejemplo es usando leyes sociales

  Una ley social es una prohibición de acciones en contextos particulares

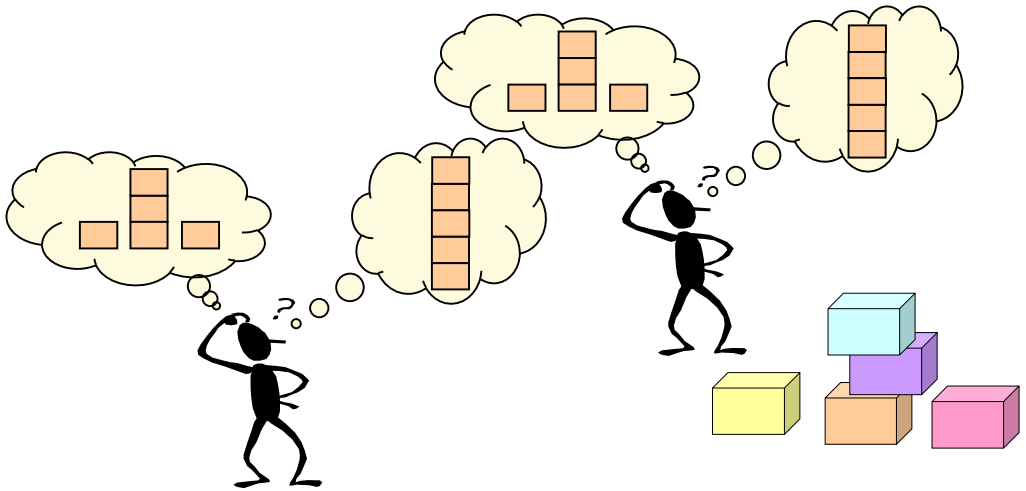
- Estas Leyes se pueden derivar trabajando desde estados indeseables y encontrando combinaciones de esos estados, para imponer las restricciones sobre acciones de manera de evitarlos (prevenir)

Sistemas Multiagentes



Entornos multiagente

- Varios agentes **interactuando** en el mismo entorno
- Las **acciones** de un agentes influencian las acciones del resto
- **Autonomía**: un agente NO puede controlar las acciones de los otros
- **Racionalidad**: un agente SÍ puede predecir las acciones de los otros



SMA

Definición

Un ***Sistema MultiAgente*** es un sistema informático formado por un ***grupo de agentes*** que ***interactúan*** entre sí utilizando protocolos y lenguajes de comunicación de alto nivel, para ***resolver problemas*** que pueden estar más allá de las capacidades o del conocimiento de cada uno.

Es una ***red de entidades*** que ***trabajan conjuntamente*** para ***encontrar respuesta a problemas*** que pueden estar más allá de la capacidad y el conocimiento individual de cada entidad.

SMA

- Son sistemas compuestos por múltiples componentes *autónomos* que poseen las siguientes características:
  - Cada agente tiene ciertas capacidades.
  - Necesitan de formas de coordinarse
  - Los datos no están centralizados
  - La computación es asíncrona.

- Buscan responder a preguntas, tales como:
  - ¿Por qué y cómo cooperan los agentes?
  - ¿Cómo los agentes pueden reconocer y resolver conflictos?
  - ¿Cómo los agentes pueden negociar?.

SMA

sistemas en los que varios agentes inteligentes interactúan, siguiendo un conjunto de metas o realizar un conjunto de tareas.

- ***"Agentes"*** son entidades computacionales autónomas que perciben su entorno a través de sensores y actúan sobre él a través de efectores.
- ***"Inteligente"*** indica que los agentes persiguen sus objetivos y ejecutan sus tareas optimizando algunas medidas de rendimiento. Funcionan flexible y racionalmente en una variedad de circunstancias. Para ello, en IAD los procesos de resolución de problemas, planificación, toma de decisiones, y aprendizaje que hacen posible que los agentes sean flexibles y racionales en su comportamiento, se llevan a cabo en escenarios multiagente.

SMA

sistemas en los que varios agentes inteligentes interactúan, *siguiendo un conjunto de metas* o *realizar un conjunto de tareas*.

- ***"Interacción"*** indica que los agentes pueden verse afectados por otros agentes en la consecución de sus objetivos y ejecución de sus tareas. Interacción puede ser:
  - Indirecta a través del entorno (por ejemplo, mediante la observación de los otros o llevando a cabo una acción que modifica el ambiente)
  - Directa a través de un lenguaje compartido (por ejemplo, intercambiando información en la que otros agentes están interesados o que confunde otros agentes).
- IAD se centra en la *coordinación* como una forma de interacción para la consecución de objetivos y la realización de tareas.
- La finalidad de la coordinación es lograr estados deseables o evitar estados indeseables para varios agentes.
- Dos patrones básicos de coordinación: cooperación y competencia

Sistemas Multiagente: ¿Qué es nuevo?

- ¿Es lo mismo que Sistemas Distribuidos/Ingeniería del SW?
  - Sí, pero **añade** autonomía + racionalidad
  - **Coordinación** no precompilada
  - **No se asume** benevolencia
- ¿Es lo mismo que Sistemas Expertos/Inteligencia Artificial?
  - Sí, pero **añade** interoperatividad + sociabilidad
    - Percepción, planificación, razonamiento, aprendizaje, …
    - **Interacciones** sociales: Negociación, Compromisos, Trust
- ¿Es lo mismo que Economía/Teoría de Juegos?
  - Sí, pero **añade** computación y racionalidad

Por qué Distribuir a la IA?

Los SMA son, por definición, una subclase de los sistemas concurrentes.

- Por consiguiente, deben preocuparse de aspectos tales como la exclusión mutua, los abrazos mortales, y la hambruna, problemas clásicos en los sistemas concurrentes.

- Sin embargo, hay tres aspectos importantes propios de los SMA:
  - los agentes son ***autónomos*** (capaces de tomar sus decisiones para satisfacer sus objetivos de diseño);
  - las ***técnicas de sincronización y de coordinación*** requeridas por un SMA están implantadas en la plataforma computacional que les da soporte;
  - los encuentros que ocurren entre los componentes del SMA es porque ellos están ***interesados*** (no se asume que todos los componentes comparten un objetivo común, como en los sistemas concurrentes).

- En ese sentido, aspectos propios a los SMA tienen que ver con las formas para alcanzar acuerdos (**negociación**), y cómo pueden los agentes **coordinarse dinámicamente** sin conocerse.

Diferencia entre RDP y un SMA

- Algunos consideran que RDP es una sub-área de SMA, y otros lo inverso.

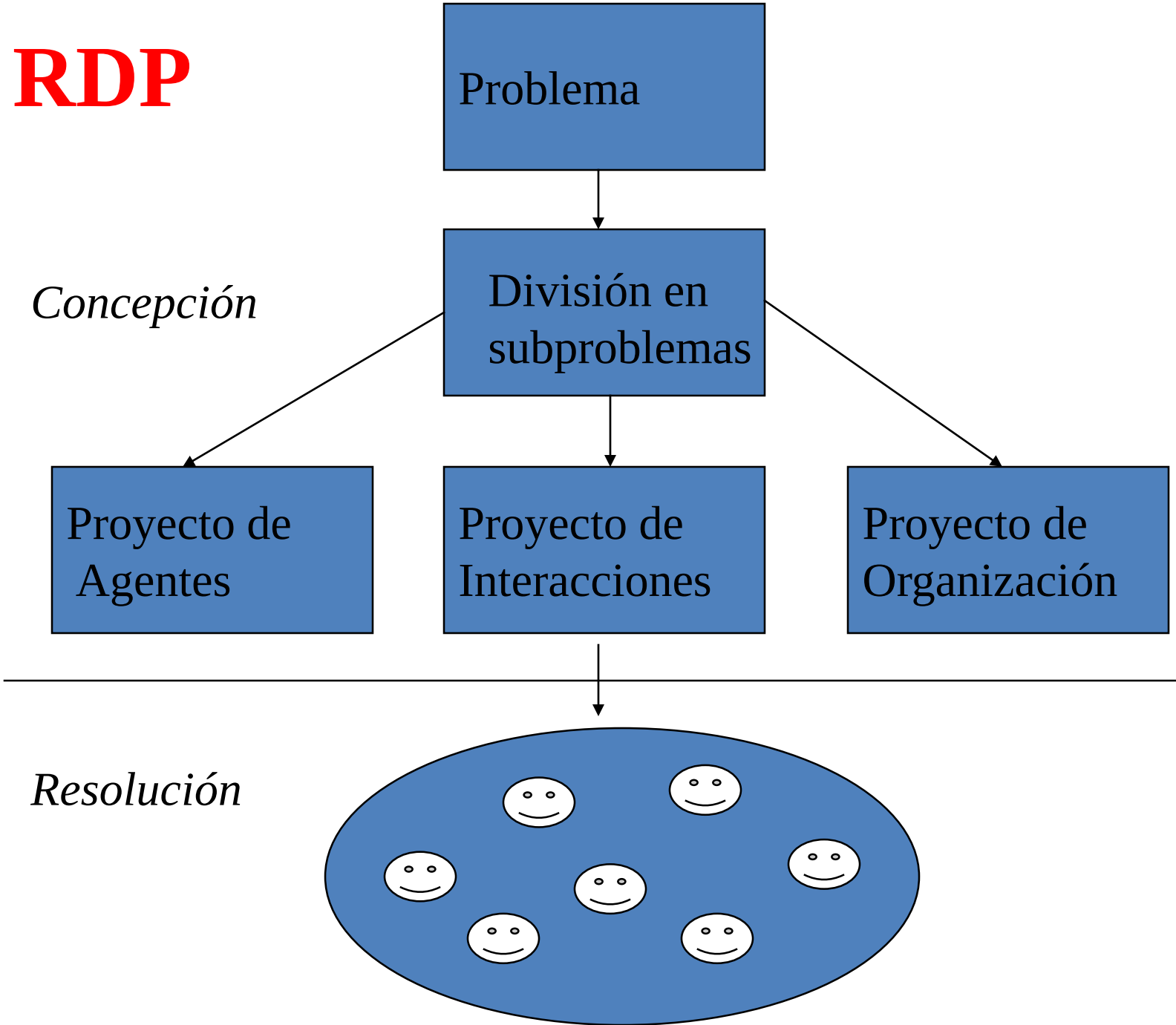
- Actualmente existe una tendencia a considerar todo como SMA

- La diferencia entre ellos podría estar en:
  - Distribución del conocimiento;
  - Existencia o no de un objetivo global;
  - Si objetivo global existe, cual es el grado de compromiso que cada agente tiene con los otros, cuando se confronta con sus objetivos individuales?

SMA

SMAs	RPD
Se enfocan primordialmente en la coordinación o negociación entre los agentes	Se enfocan en la resolución de un pb. dado, su descomposición, y síntesis de la soluciones
Son autónomos con la posibilidad de objetivos individuales	Necesitan sincronizarse
Permite la emergencia	Es determinista

RDP


Problema
Concepción
División en subproblemas
Proyecto de Agentes
Proyecto de Interacciones
Proyecto de Organización
Resolución

SMA

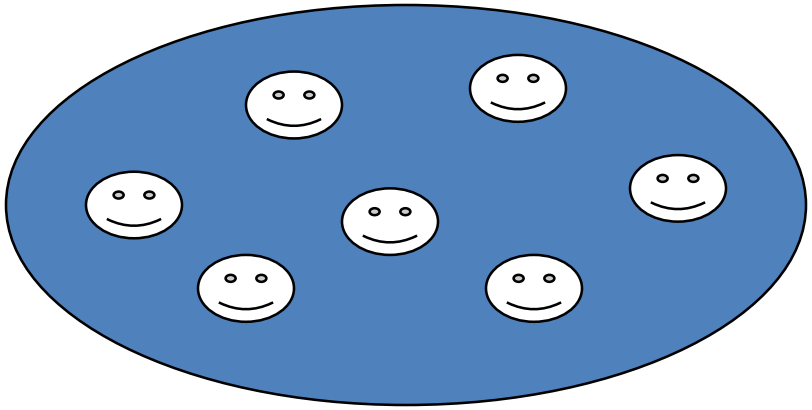
Concepción

Proyecto de Agentes

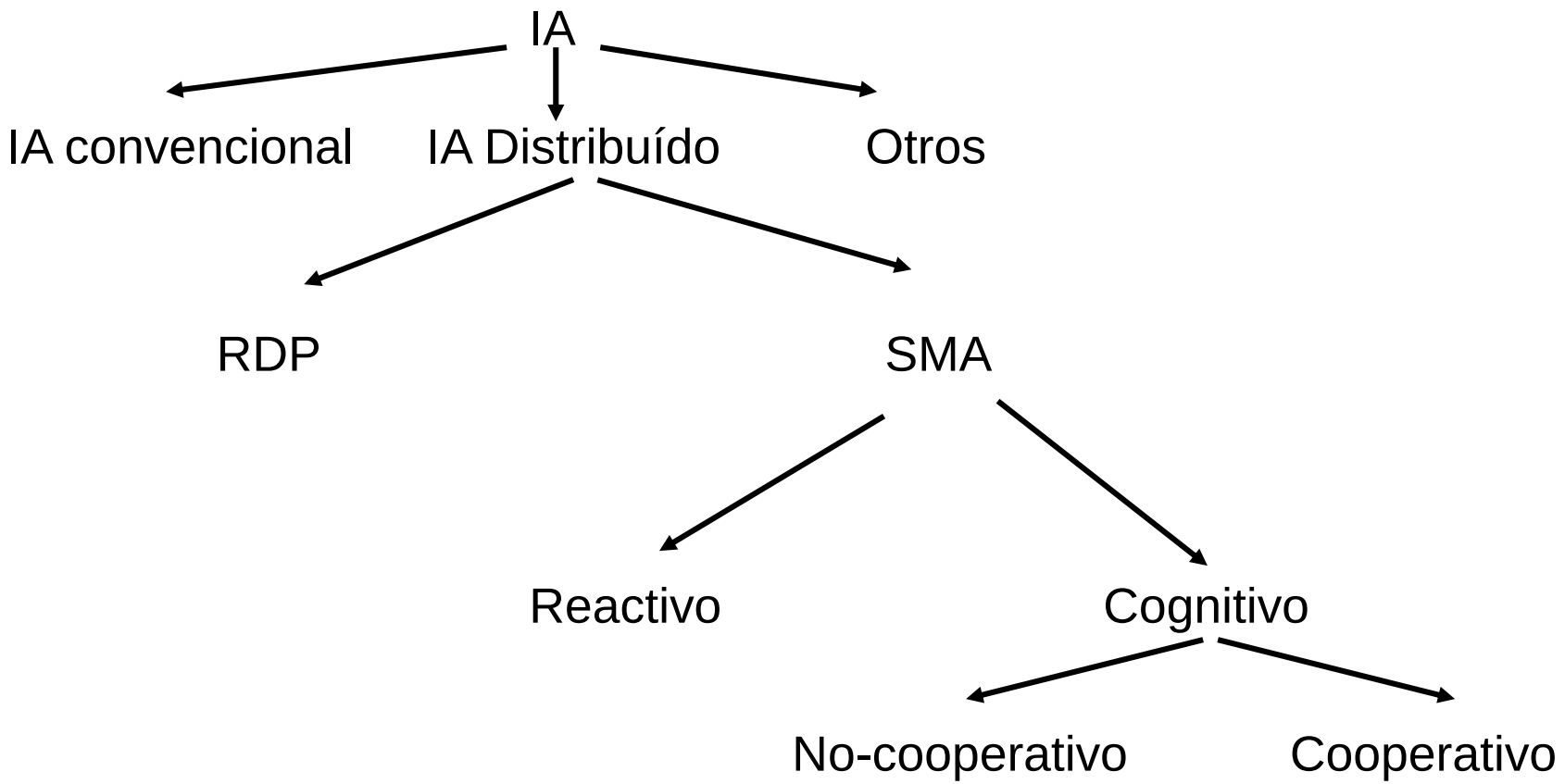
Proyecto de Interacciones

Proyecto de Organizaciones

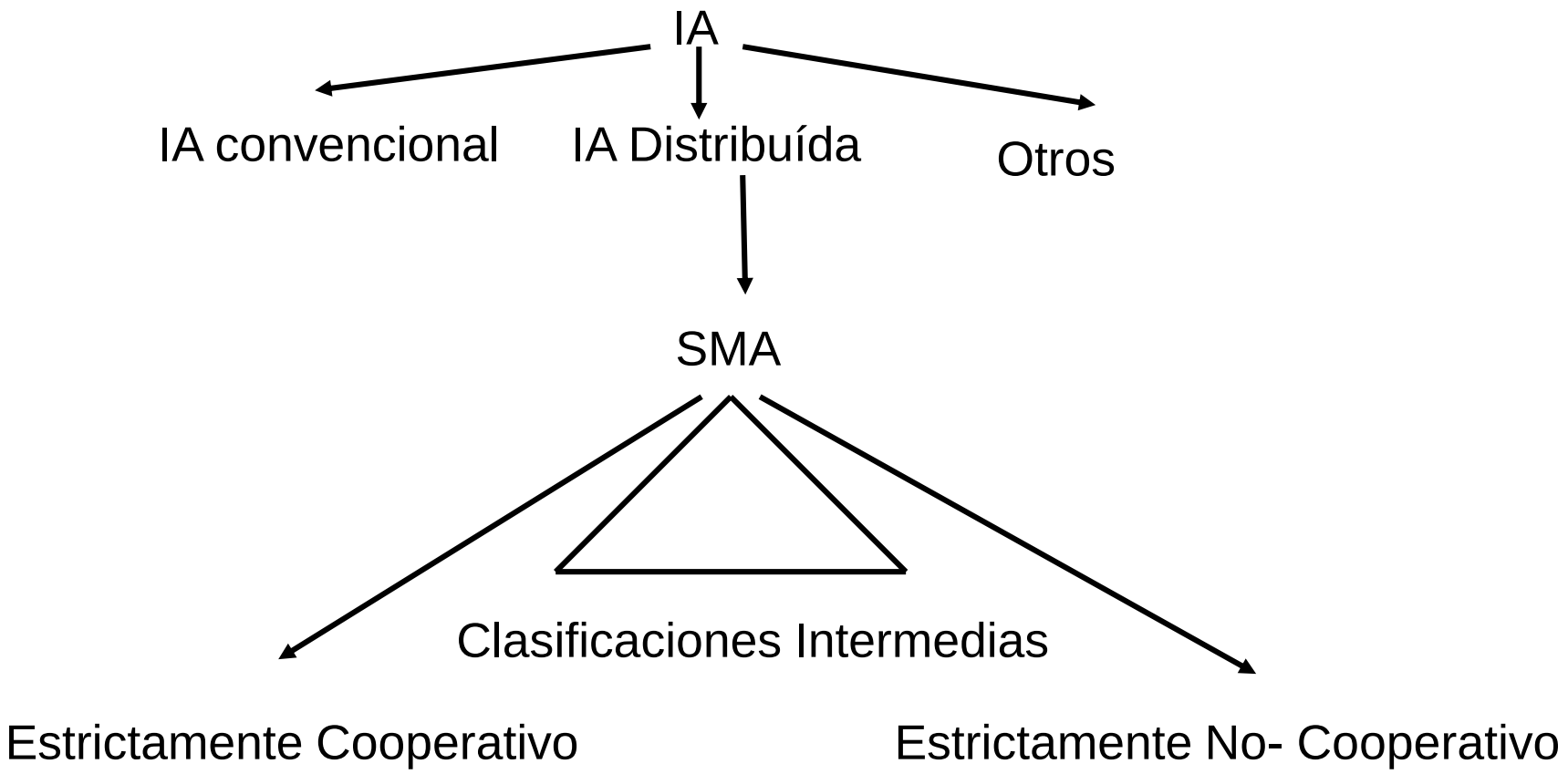
Resolución



Taxonomía Anterior



Taxonomía 'Moderna'



Problemas en los SMA

- ¿Cómo asignar las entidades de un problema entre los agentes del sistema?
- ¿Cómo habilitar a los agentes para comunicarse e interactuar? Qué lenguaje de comunicación utilizar?
- ¿Cómo garantizar que los agentes actúen de forma coherente, sin provocar efectos indeseados? **(Dilema del prisionero)**
- ¿Cómo habilitar a los agentes para representar e razonar sobre los actos de otros agentes para conseguir coordinarse con ellos?
- ¿Cómo gerenciar las limitaciones de recursos del entorno?

Sistemas Multiagente:
Temas de investigación

- **Heterogeneidad:**
  - Agent communication languages (ACLs)
  - Semántica: ontologies, service directories, matchmaking
- **Coordinación:**
  - Alcanzar consenso resolviendo conflictos de intertés:
    - Coalition formation, Negotiation, Trust, …
  - Regular agreement mechanisms :
    - Virtual Organisations, Norms, Incentive Engineering, …
  - Hacer que la i**nformación** esté **disponible:**
    - MA Planning, MA Optimisation, POMDPs, …
- **AOSE**: SOC, OOP, Grid, AOP
  - Plataformas,middleware, herramientas
  - Metodologías AOP
- **Interacción** Humano-Computadora
  - Avatares, agentes emocionales, …

Características de Ambientes Multiagentes

- Una comunidad de **agentes autónomos** tratando de resolver problemas, por lo cual **comparten conocimiento** acerca de los problemas y sus soluciones.
- **No tienen un diseño centralizado**.
- Contienen **agentes distribuidos**, que pueden estar interesados en si mismos o en cooperar
- Cada agente tiene capacidad para **solucionar parcialmente el problema**
- **No hay un sistema global que los controla**
- Los **datos no están centralizados**
- La **computación es asíncrona**
- Pueden **incorporarse dinámicamente nuevos tipos de entidades** en el sistema y **cambiar las existentes**
- **Evolución del comportamiento independiente** para cada uno de los componentes del sistema
- **Hay incertidumbre**
- **Organización de entidades que interactúan** para resolver conjuntamente problemas globales

Características de ambientes multiagentes

- Tienen una infraestructura específica de comunicación y protocolos de interacción.
- Autonomía de diseño:

  Plataforma – Protocolo de interacción – Lenguaje
- Arquitectura interna:
  - Infraestructura de comunicación: Memoria compartida (blackboard), Conexión basada en mensajes (punto a punto, multicast o broadcast), sincronización o no
  - Servicio de directorio: páginas blancas, páginas amarillas
  - Protocolos: FIPA, KQML, CORBA, etc.
  - Servicios de mediación: Basado en ontologías, Transacciones
  - Servicios de seguridad: autenticación, anonimato
  - Soporte de operaciones: Almacenamiento, Redundancia, Restauración, Contabilidad, Timestamp

Niveles de organización en un SMA

- ***El nivel microsocial***, que se interesa esencialmente por las interacciones y conexiones que existen entre los agentes.

- ***El nivel de grupo***, que se interesa en las estructuras que se producen en la composición de una organización compleja. En este nivel se estudian:
  - Los roles y actividades de los agentes,
  - La emergencia de las estructuras organizacionales entre los agentes, y
  - El problema general de agregación de agentes para la constitución de organizaciones.

- ***El nivel de la sociedad global (o poblaciones),*** que se interesa en la dinámica de un gran número de agentes, así como en la estructura general del sistema y su evolución.

Análisis Organizacional de los SMA

PRINCIPALES FUNCIONES QUE LOS COMPONENTES DE UNA ORGANIZACIÓN DEBEN CUMPLIR

FUNCION\AMBITO	SOCIALES	RELACIONAL	AMBIENTAL	PERSONAL
REPRESENTACION	REPRESENTACION DE LA SOCIEDAD DONDE ESTA INMERSO: ROLES, FUNCIONES,…	REPRESESNTACION DE LOS OTROS : OBJETIVOS , COMPETENCIA., ETC.	REPRESENTRACION DEL MUNDO,	AUTO-REPRESENTACION
ORGANIZARSE	PLANIFICACION. ACCIONES SOCIALES	COORDINACION COMUNICACIONES, PLANIF.ICACION INTERACCIONES	PLANIFICACION ACCIONES Y CONTROL DE EJECUCION EN EL AMBIENTE	GESTION TAREAS INTERNAS
ELEMENTOS PARA TOMA DE DECISION	OBJETIVOS Y RESTRICCIONES COLECTIVAS DE LA SOCIEDAD	DEMANDAS Y RESTRICCIONES DE LOS OTROS	FUENTES DE DESEOS, PLACER, ETC. QUE EXISTEN EN EL AMBIENTE	NECESIDADES Y RESTRICCIONES INTERNAS.
INTERACCION	PRIMITIVAS DE INTERACCION EN LA SOCIEDAD	MECANISMOS DE COMUNUNICACION CON LOS OTROS	MECANISMOS DE PERCEPCION Y ACCION	AUTO-COMUNICARSE, AUTO-HABLARSE
PRODUCCION U OPERATIVAS	GESTION. Y DIRECCION EN LA SOCIEDAD	COORDINACION Y COOPERACION CON LOS OTROS.	CONTROL, DIAGNOSTICO, ETC SOBRE AMBIENTE	AUTO-MODIFICACION (P.E. APRENDIZ.AJE) , AUTO-GENERARSE, …
CONSERVACION	CONSERVACION DE LA SOCIEDAD (	CONSERVACION DE LAS RELACIONES Y DE LOS OTROS.	CONSERVACION RECURSOS DEL AMB., DEFENSA	CONSERVACION INDIVIDUAL, REPARACION

Análisis Organizacional de los SMA

ORGANIZACIÓN REACTIVA

	SOCIALES	RELACIONAL	AMBIENTAL	PERSONAL
REPRESENT.				
ORGANIZAC			P	
DECISORIOS			P	o
INTERACC.			P	
PRODUCTIV			P	
CONSERVAT.			p	

ORGANIZACIÓN ANIMAL SOCIAL

	SOCIALES	RELACIONAL	AMBIENTAL	PERSONAL
REPRESENT.		P	P	
ORGANIZAC		P	P	
DECISORIOS	P	P	P	P
INTERACC.		P	P	P
PRODUCTIV				
CONSERVAT.	P	P	P	P

ORGANIZACIÓN HUMANA

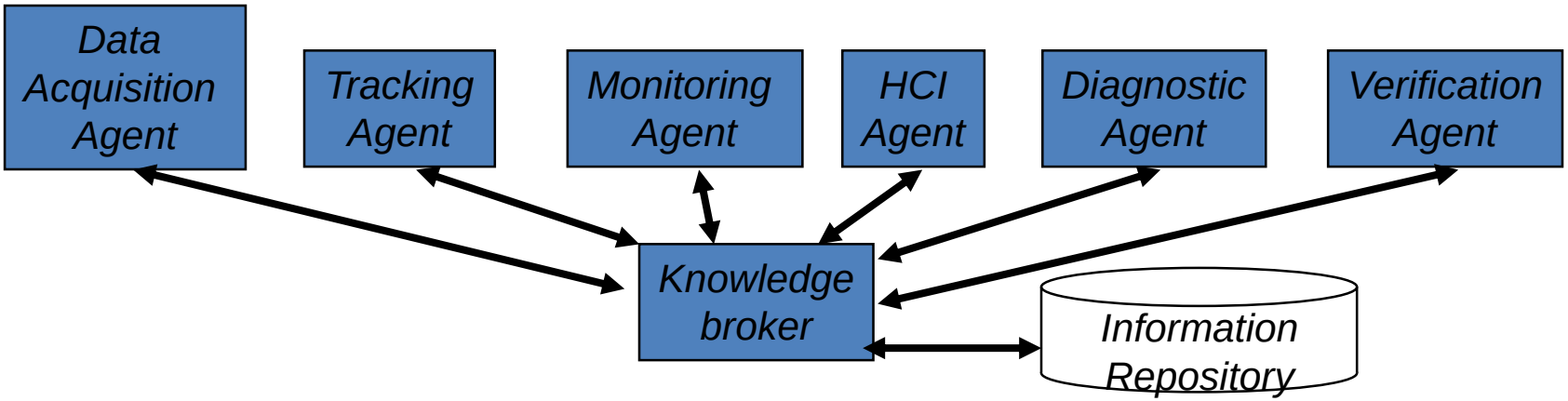
	SOCIALES	RELACIONAL	AMBIENTAL	PERSONAL
REPRESENT.	P	P	P	P
ORGANIZAC	P	P	P	P
DECISORIOS	P	P	P	P
INTERACC.	P	P	P	P
PRODUCTIV	O	O	O	O
VEGETATIVAS	P	P	P	P

P: PRESENCIA
C:OPCIONAL

Ejemplo SMA

Sistema de distribución de Agua: RDP

- Objetivo global;
- Tareas pre-definidas
- Usualmente no existe gran comunicación entre los agentes;
- Existe un control global



Ejemplo SMA

Hormiguero - SMA Reactivo

- Problema resuelto por individuos que interactúan entre si;
- Existe un gran número de ellos;
- Son normalmente idénticos y poseen conocimiento limitado;
- Cada uno no tiene conciencia del problema general;
- “Cooperan’ entre si de manera no explicita;
- Una solución ‘surge’ a través de las interacciones entre ellos.

Ejemplo SMA

Equipo de Futbol – SMA Cognitivo Cooperativo

- Cada jugador posee un conocimiento individual y limitado;
- Cada jugador no puede resolver el problema solo;
- Cada jugador puede tener características diferentes a los demás;
- Cada jugador actúa de forma autónoma y asíncrona;
- Existe un objetivo global que es de conocimiento de todos los individuos;
- Ese objetivo global está en**cima** de los objetivos individuales de cada agente;
- No existe un control global;
- Al juntar las capacidades individuales se resuelve el problema.

Ejemplo de sistema

Negociación trabajadores – SMA Cognitivo No-Cooperativo

- Objetivos de cada parte son usualmente contrapuestos;
- Información de cada parte incompleta;
- Existe un objetivo global, pero que no es mas importante que los objetivos individuales;
- Cada parte procura convencer su oponente (proceso de negociación);
- No hay certeza sobre que al otra parte va hacer;
- No existe un control centralizado del proceso;

SMAs Reactivos X SMAs Cognitivos

- Conocimiento implícito
- Sin historial
- Conductista o positivista
- Reactivo
- La comunicación indirecta
- Inspirado en colectivos de animales
- Una gran cantidad de agentes

- Conocimiento explícito
- Históricos
- Subjetivista (BDI)
- Deliberativo
- Comunicación directa
- Inspirado por las sociedades humanas
- pocos agentes
- s agentes

SMAs Reactivos

- La inteligencia puede ser generado sin conocimiento explícito o razonamiento abstracto!
  - Es una propiedad emergente de ciertos sistemas complejos.
  - La verdadera inteligencia se encuentra en el mundo, no en los sistemas (expertos, probadores, etc.)!
  - Comportamiento inteligente emerge de las interacciones con el medio ambiente.

SMAs Cognitivos

- **Agentes organizados**
  - Múltiples puntos de vista, leyes y normas sociales
- **Agentes Distribuidores**
  - Resolución de conflictos mediante la negociación
- **Agentes intencionales**
  - Intenciones (u opciones), compromisos, planes parciales
- **Agentes cooperativos**
  - Representación mutua, asignación de tareas

ANALISIS ESTRUCTURAL

PERMITE *ANALIZAR* LAS RELACIONES ABSTRACTAS ENTRE AGENTES, *ORDENAR* AL CONJUNTO DE INTERACCIONES ENTRE AGENTES, *ESTABLECER* LA EVOLUCION DE LAS RELACIONES

ESTUDIA:

- RELACION ENTRE AGENTES Y TAREAS
- RELACIONES ENTRE AGENTES
- MODOS DE ACOPLAMIENTO ORGANIZACIONAL
- ESTRUCTURAS DE SUBORDINACION Y TOMA DE DECISION

ANALISIS ESTRUCTURAL

A. RELACION ENTRE AGENTES Y TAREAS:

AGENTE *CARACTERIZADO* POR:

ARQUITECTURA+COMPORTAMIENTO+CAPACIDADES

ENFOQ UE FUNCIONAL (VERTIC. U HORIZ.) O POR OBJETO

B. RELACIONES ENTRE AGENTES

RELACIONES	**ESTATICA**	**DINAMICA**
AMISTAD		
COMUNICACIONAL		
SUBORDINACION	MAEST/ESCL	DEMANDA DE SERV.
OPERATIVA	DEPEND DE TAREAS	COMPROM. HACER
INFORMACIONAL	DEPEND SABERES	
CONFLICTUAL		
COMPETITIVA		

ANALISIS ESTRUCTURAL

MODOS DE *ACOPLAMIENTO*

- ***Acoplamiento fijo:*** cada componente posee un rol fijo (o eventualmente varios roles, pero todos fijos), al igual que las relaciones entre ellos son fijas. La reorganización es imposible, por lo que es imposible adaptarse a su medio ambiente.

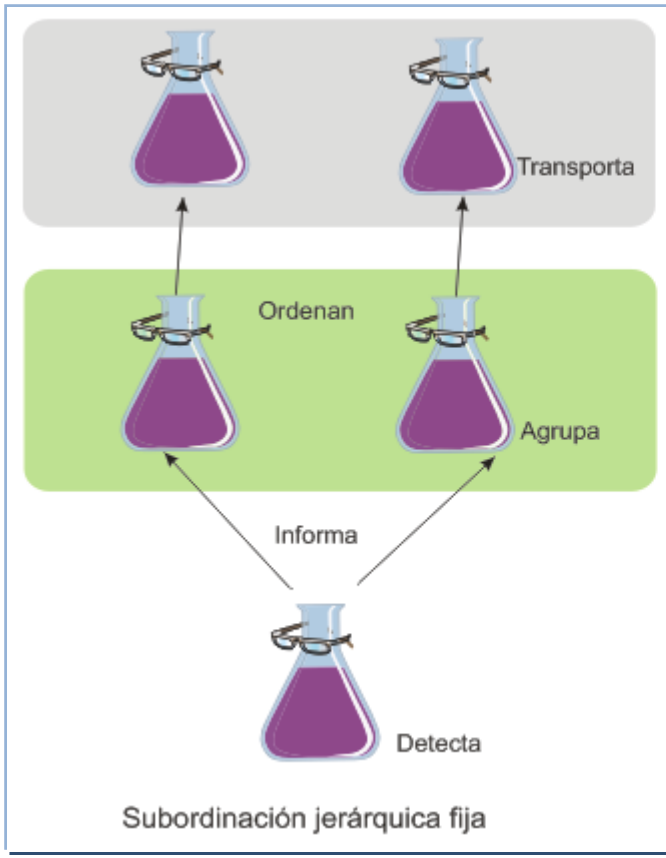
- ***Acoplamiento variable:*** corresponden a estructuras organizativas fijas, cuya concretización (instanciación) puede variar. Las relaciones entre los agentes pueden cambiar, pero a través de mecanismos predefinidos bien precisos.

- ***Acoplamiento evolutivo:*** caracterizados por estructuras organizativas variables, pudiendo su concretización variar o no. Las relaciones abstractas entre los agentes pueden evolucionar (emerger), en función de los resultados de la organización (SMA).

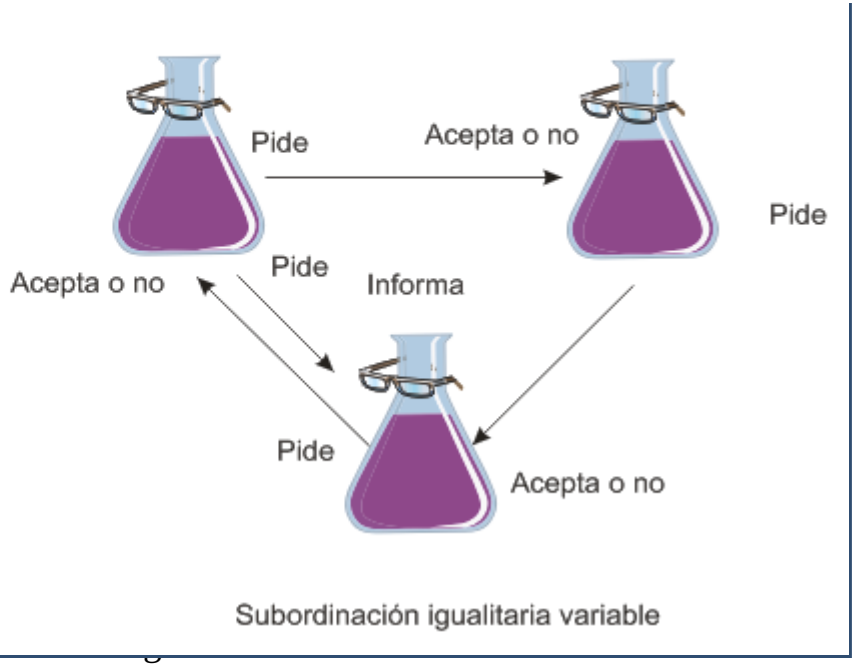
ANALISIS ESTRUCTURAL

ESTRUCTURAS DE *SUBORDINACION*

- **JERARQUICAS:** VARIABLES (POR SERVICIO) Y FIJAS (TIPO MILITARES)
- **IGUALITARIAS:** VARIABLES (MODELO DE MERCADO)Y FIJAS (RDP)



Subordinación jerárquica fija



Subordinación igualitaria variable

ANALISIS ESTRUCTURAL

TIPOS DE *CONSTITUCION* DE LAS ESTRUCTURAS ORGANIZACIONALES

PREDEFINIDAS

EMERGENTES

CÓMO SE REPARTE EL TRABAJO EN UN SMA:

GRADO DE ESPECIALIZACION — TODO PUDIENTE / ESPECIALIZADO

GRADO DE REDUNDANCIA

TIPOS DE ORGANIZACIONES

ORGANIZACION HIPERESPECIALIZADA NO REDUNDANTE

ORGANIZACION ESPECIALIZADA REDUNDATE

ORGANIZACION GENERAL REDUNDANTE

ORGANIZACION GENERAL NO REDUNDANTE

Estructuras sociales

- *La estructura de poder y de interdependencia*
- *La estructura de conocidos*, que emerge de la unión de todos los conocidos de cada agente;
- *La estructura de comunicación* (la red global de los posibles canales de comunicación directos o indirectos);
- *La estructura de compromiso*, que emerge de todas la relaciones de delegación-adopción y de formación de coaliciones;
- *La estructura de reglas y normas* sobre acciones e interacciones.

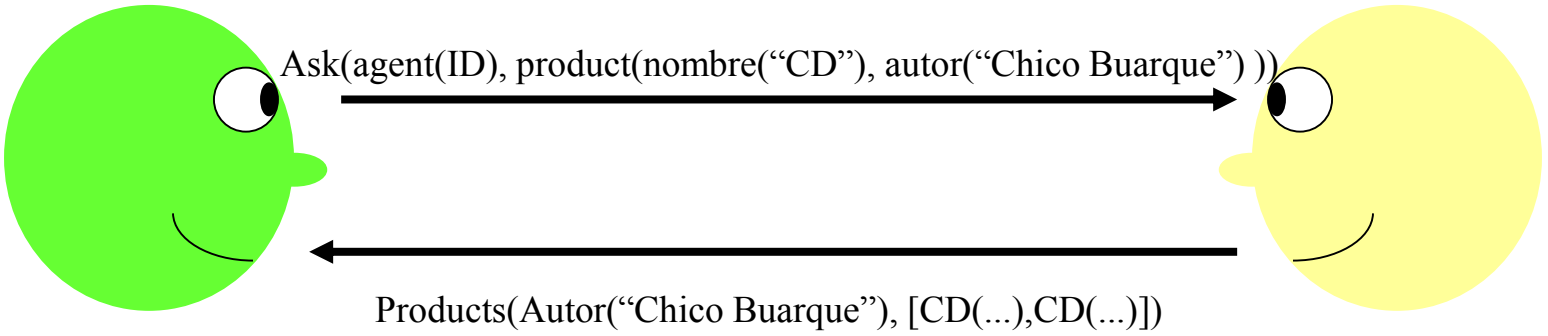
SMA

Análisis Estructural

Analizar las Relaciones abstractas entre Agentes,
Ordenar las Interacciones entre Agentes, *Establecer la evolución de las Relaciones*

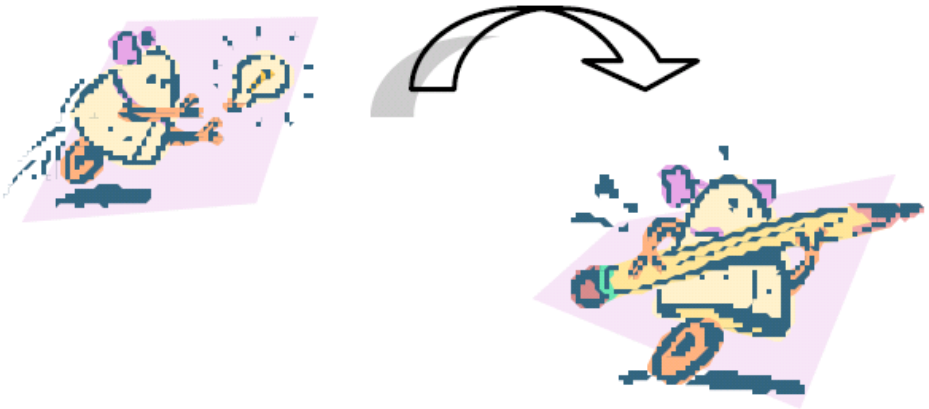
Desarrollo de SMA

- Busca definir:
  - Quiénes son los agentes
  - Protocolo de comunicación entre los agentes (redes de contrato)
  - Reglas de funcionamiento de cada agente (Relaciones entre: creencias, percepción y acciones)



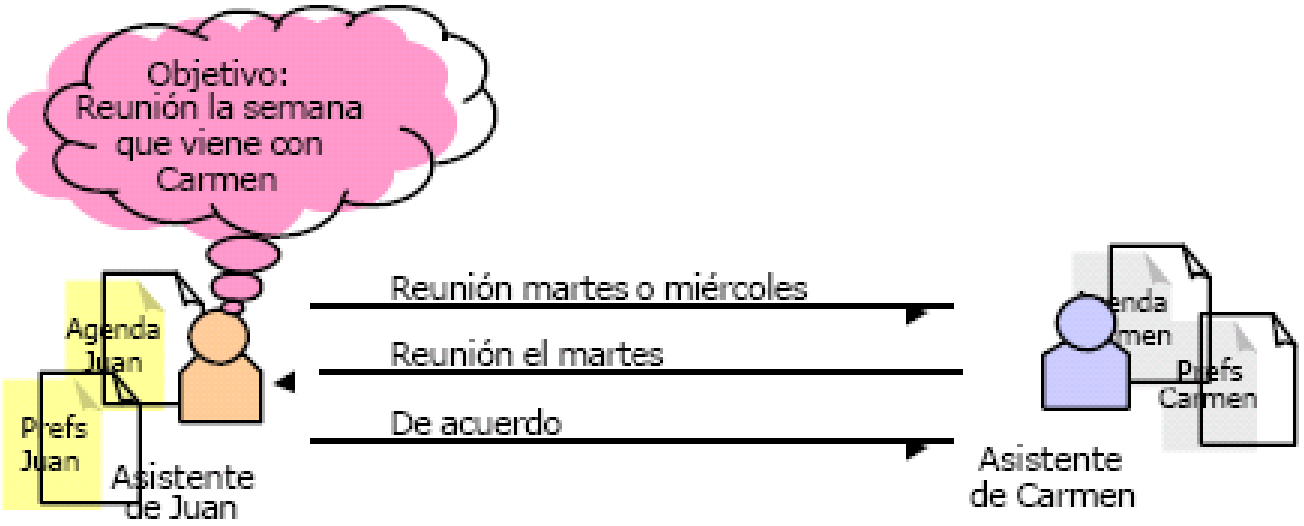
SMA

- El agente persigue satisfacer sus objetivos
  - Toma decisiones
  - Puede descomponer objetivos en subobjetivos
  - Ejecuta tareas



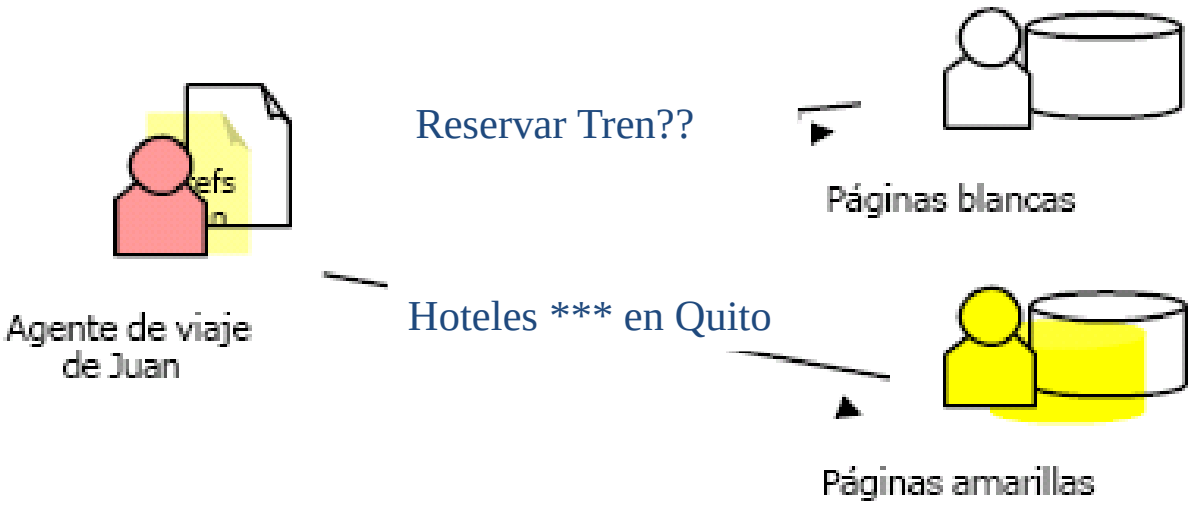
SMA

- Para cumplir objetivos necesita la colaboración con otros agentes
  - Negociación
  - Delegación
  - Coordinación



SMA

- Los agentes necesitan servicios de localización de agentes
  - Páginas blancas
  - Páginas amarillas



Utilidad de los sistemas multiagentes

- **En el diseño de sistemas distribuidos los agentes proporcionan:**
  - 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

Aplicaciones

Servicios de información en Internet
Recuperación y extracción de información
Comercio electrónico
Mercado de servicios electrónico
Sistemas de Control (tiempo real)
Equipos móviles y PCs en el hogar
Redes públicas de telecomunicaciones
Provisión de servicios bajo demanda
Descentralización del control y gestión de redes
Grid Computing
Gestión de procesos (workflow)
Simulación de sistemas dinámicos
Juegos (bots)
Robótica
Etc.

Personalización de servicios

Flexibilidad de la distribución

Delegación de tareas

Cuándo usar...

- **Problemas complejos**
  - divide y vencerás, ej. conjunto de 8 números, clasificación, etc.
- **Problemas distribuidos intrínsecamente**
  - ej. sistemas control de tráfico, etc.
- **Problemas que requieren un tiempo de respuesta rápido**
  - procesamiento paralelo, ej. buscar en la internet, etc.
- **Problema con dominios de tareas o conocimiento**
  - un agente para cada tipo de conocimiento / tarea, ej. planta nuclear

UNIVERSIDAD
DE LOS ANDES
MERIDA VENEZUELA

Ingeniería del Software Orientada a Agentes

Jose Aguilar

Ingeniería del Software Orientada a Agentes

- Los agentes representan un **nuevo nivel de abstracción** que puede ser utilizado por los desarrolladores de software para **entender, modelar y desarrollar** de un modo más natural una clase importante de sistemas distribuidos.

- Las técnicas de **desarrollo software habituales no son adecuadas para esta tarea**, ya que no son capaces de capturar los aspectos únicos de los SMA:
  - **Comportamiento flexible, autónomo, de resolución de problemas**
  - **Riqueza en sus interacciones**
  - **Complejidad de la estructura organizacional**

Metodologías de desarrollo de SMA

Como todo desarrollo de software se requieren unos estándares de desarrollo.

La metodología debe permitir ir actualizando los agentes en función de la caracterización que sea necesaria en el SMA.

Tener en cuenta las necesidades de especificación de los SMA

- Complejidad de la estructura organizacional
- Planificación de tareas
- Riqueza en sus interacciones: Intercambio de información con lenguajes de comunicación orientados a agentes
- Movilidad del código
- Motivación de los componentes del sistema
- Comportamiento flexible, autónomo, de resolución de problemas

La construcción de SMA integra tecnologías de distintas áreas de conocimiento

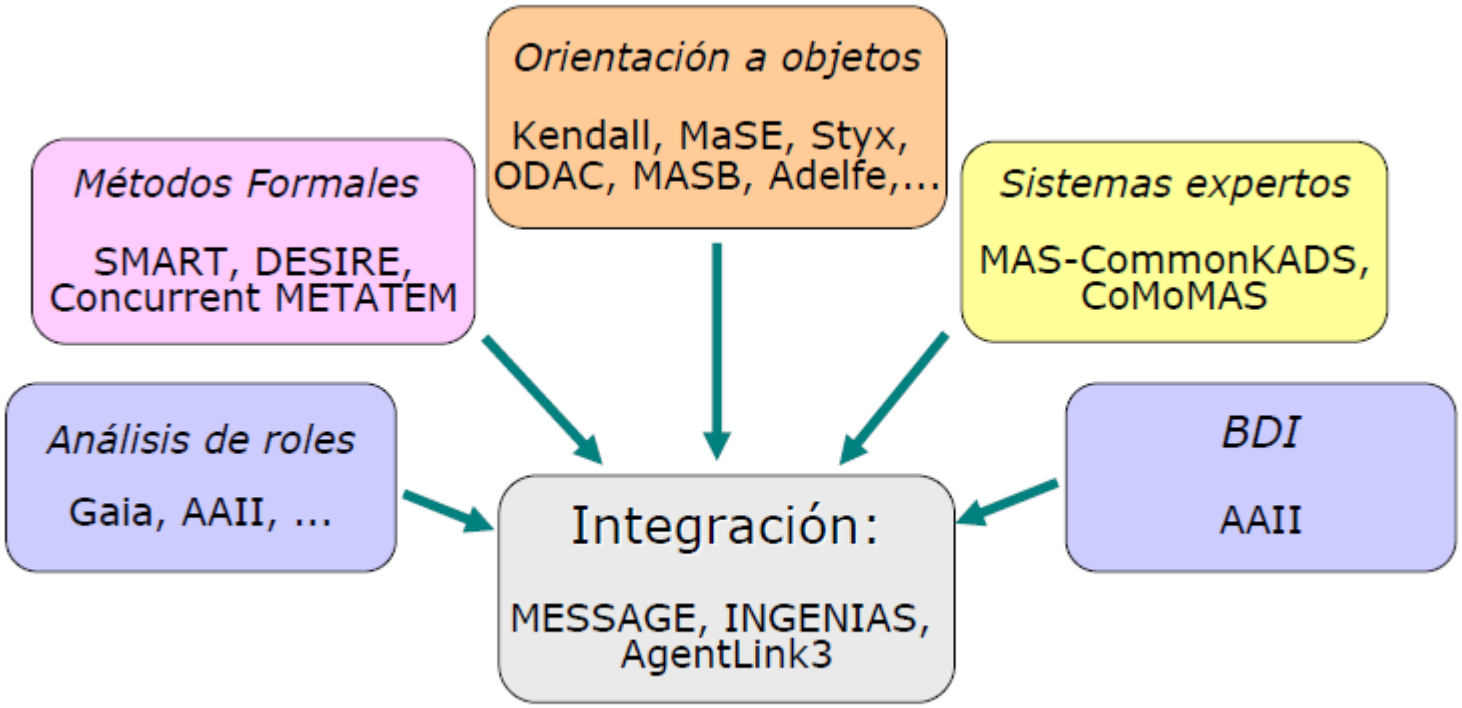
- **Técnicas de *ingeniería del software :*** *estructurar el proceso* de desarrollo
- **Técnicas de *inteligencia artificial :*** dotar a los programas de capacidad para tratar situaciones imprevistas y tomar decisiones
- **Programación concurrente y distribuida :** *tratar* la coordinación de tareas ejecutadas en diferentes
- máquinas bajo diferentes políticas de planificación.

Ingeniería del Software Orientada a Agentes

Resolver

- Las entidades que estarán representados en el sistema (del dominio);
- Definición de las percepciones y acciones que cada agente puede realizar;
- Definición de creencias y objetivos.
- Definición de las relaciones de comunicación entre agentes (de orden);

Metodologías de desarrollo de SMA



Ingeniería del Software Orientada a Agentes

- Agent-Oriented Software Engineering (AOSE)
  - Gaia
  - MaSE
  - Agent UML
  - Prometheus
  - MASINA
- Ingeniería del Conocimiento Orientada a Agentes
  - MASCommonKADS
  - DESIRE
  - Cassiopeia
- Métodos formales orientados a agentes
  - Métodos formales en AOSE
  - Especificación en Z

MASINA

Metodología que permite especificar Sistemas Multi-agentes, la cual es una extensión de MAS-CommonKADS.

Fases

Conceptualización

- Casos de uso (descripción de acciones necesarias para producir un resultado útil)
- Actores (roles desempeñados por alguna persona, una pieza de software, u otro sistema)

Análisis y Diseño

- Modelos para describir los agentes del sistema, sus tareas, su organización y los medios de comunicación.
- Diseño técnico del sistema (modelo de implementación).

Codificación y prueba

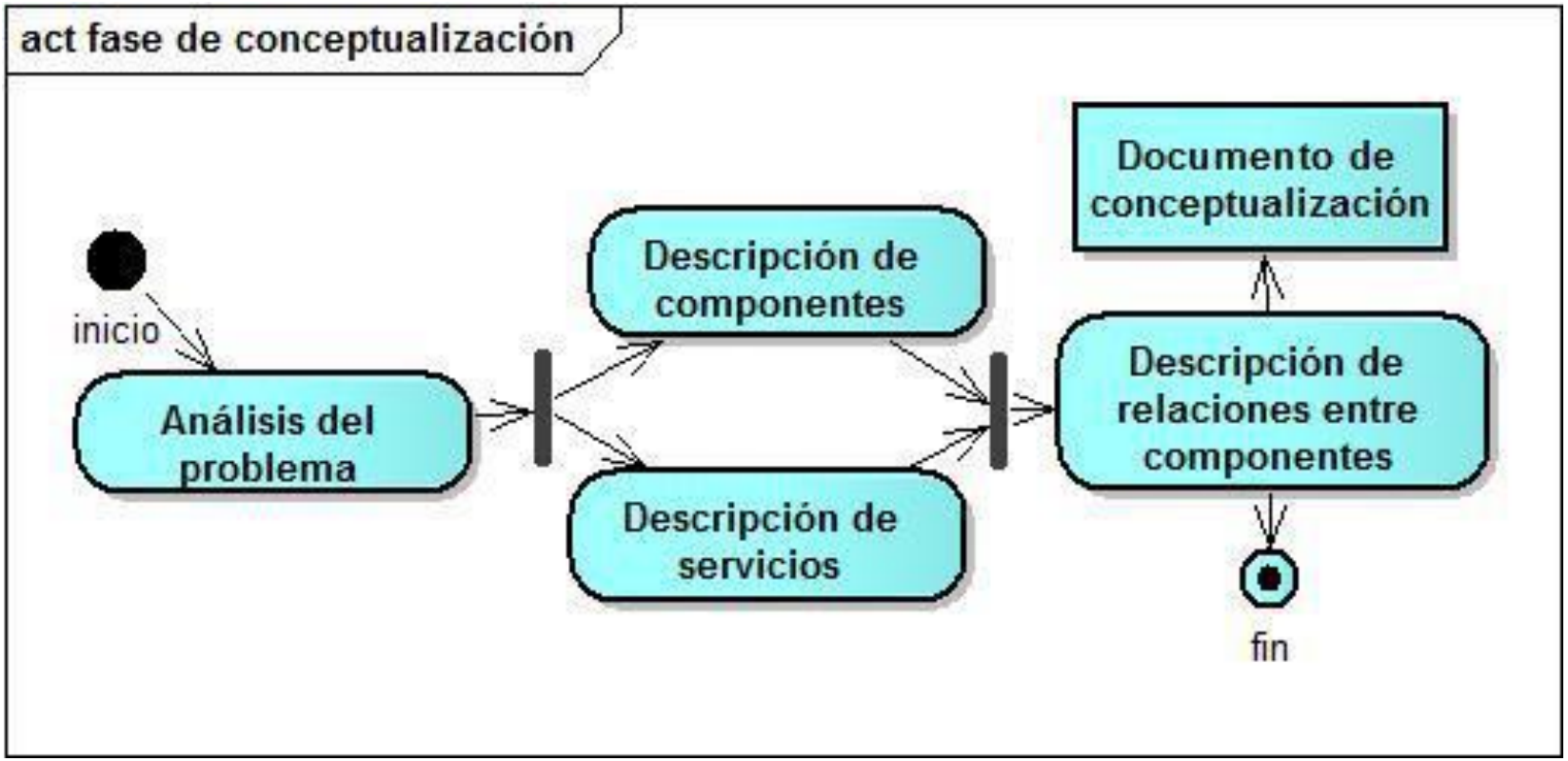
Integración

Operación y mantenimiento

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

MASINA

Fase de Conceptualización



UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

MASINA

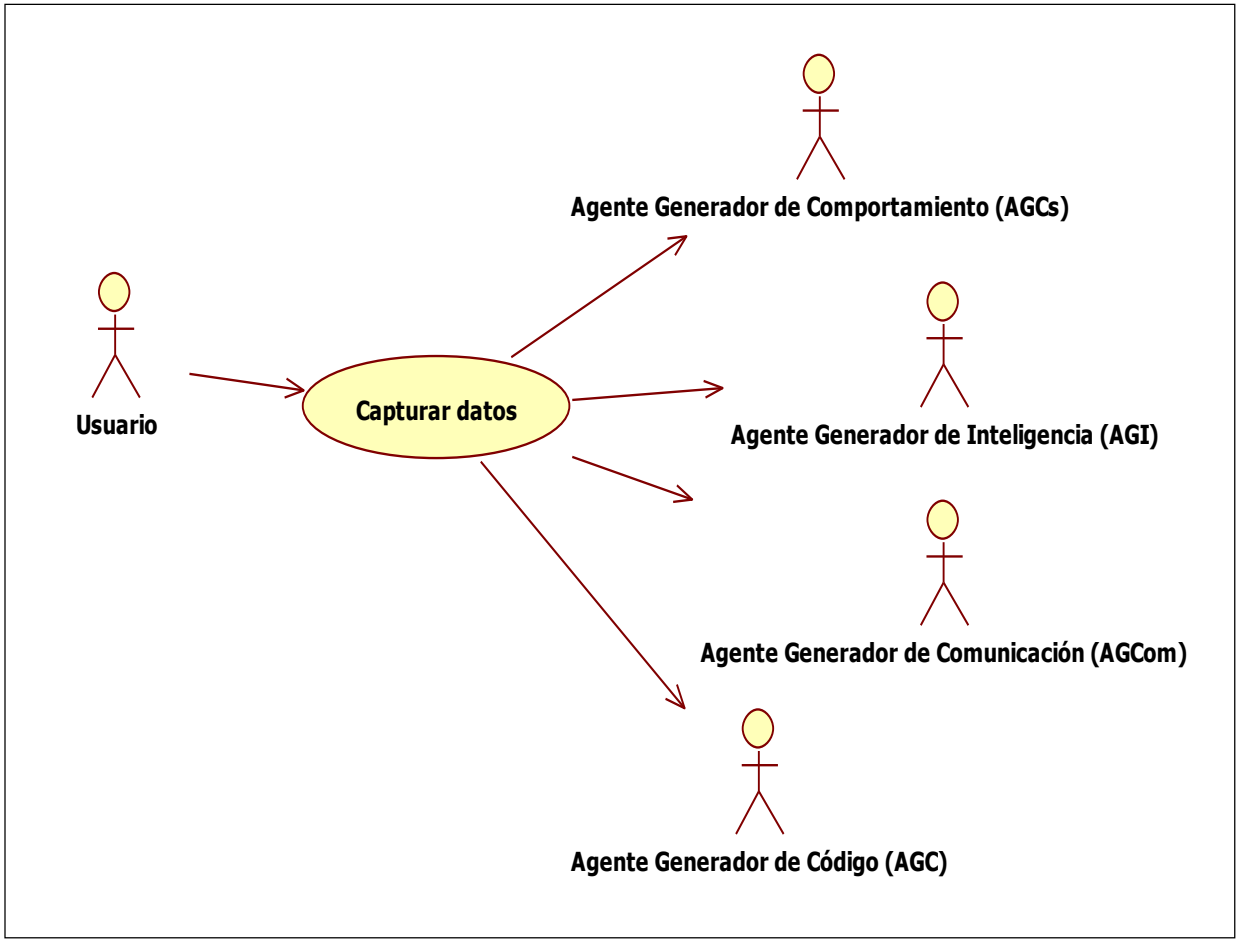
- El producto de esta fase es un **documento de conceptualización**:
  - La descripción de los componentes (agentes) del sistema,
  - La especificación de los servicios y de las actividades para prestar los servicios ofrecidos por cada componente del sistema
  - La descripción general de las relaciones entre los componentes del sistema.
- Para eso se usan: **casos de uso y diagramas de actividades**

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

MASINA

Fase de Conceptualización

Casos de uso

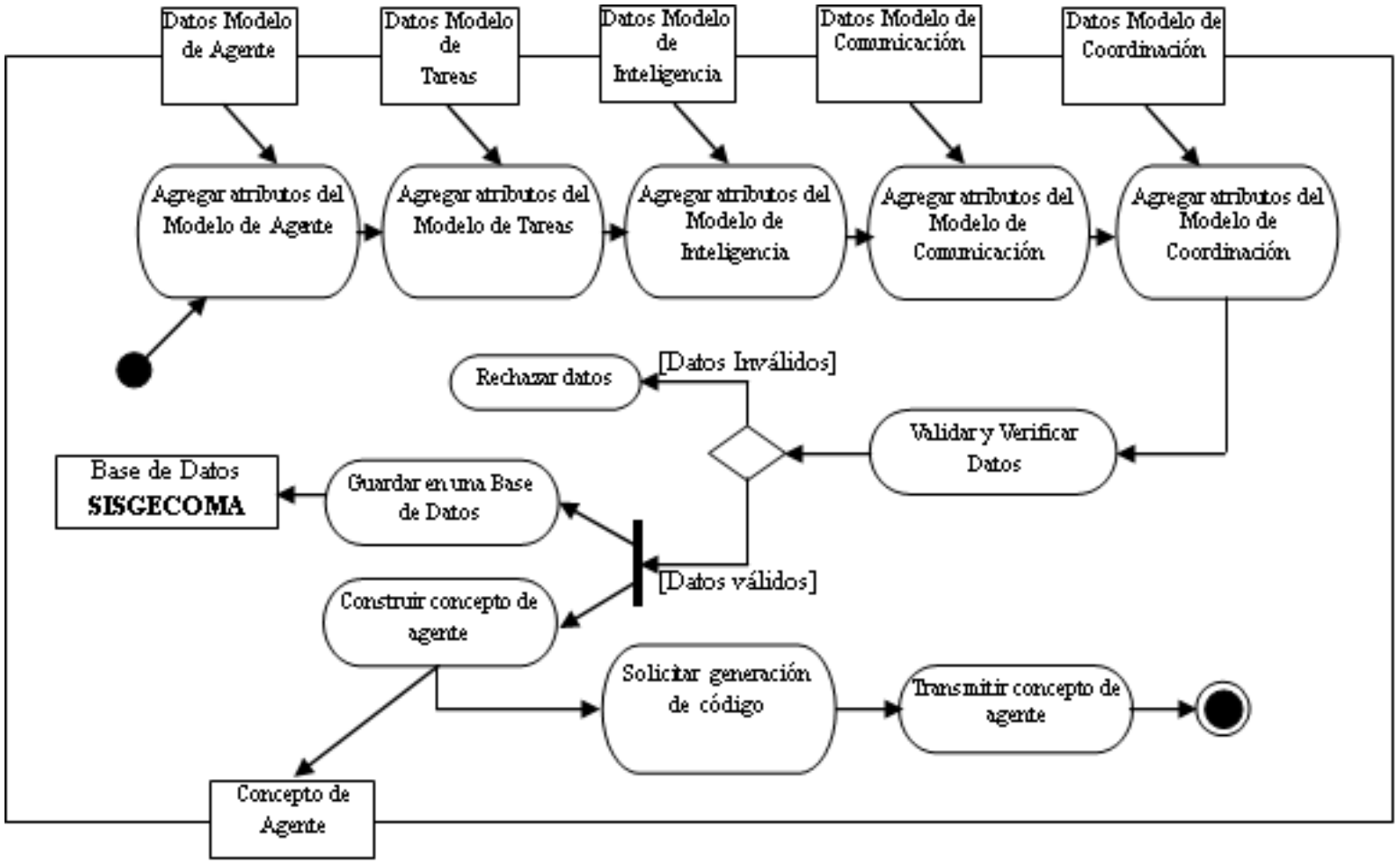


UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

MASINA

Fase de Conceptualización

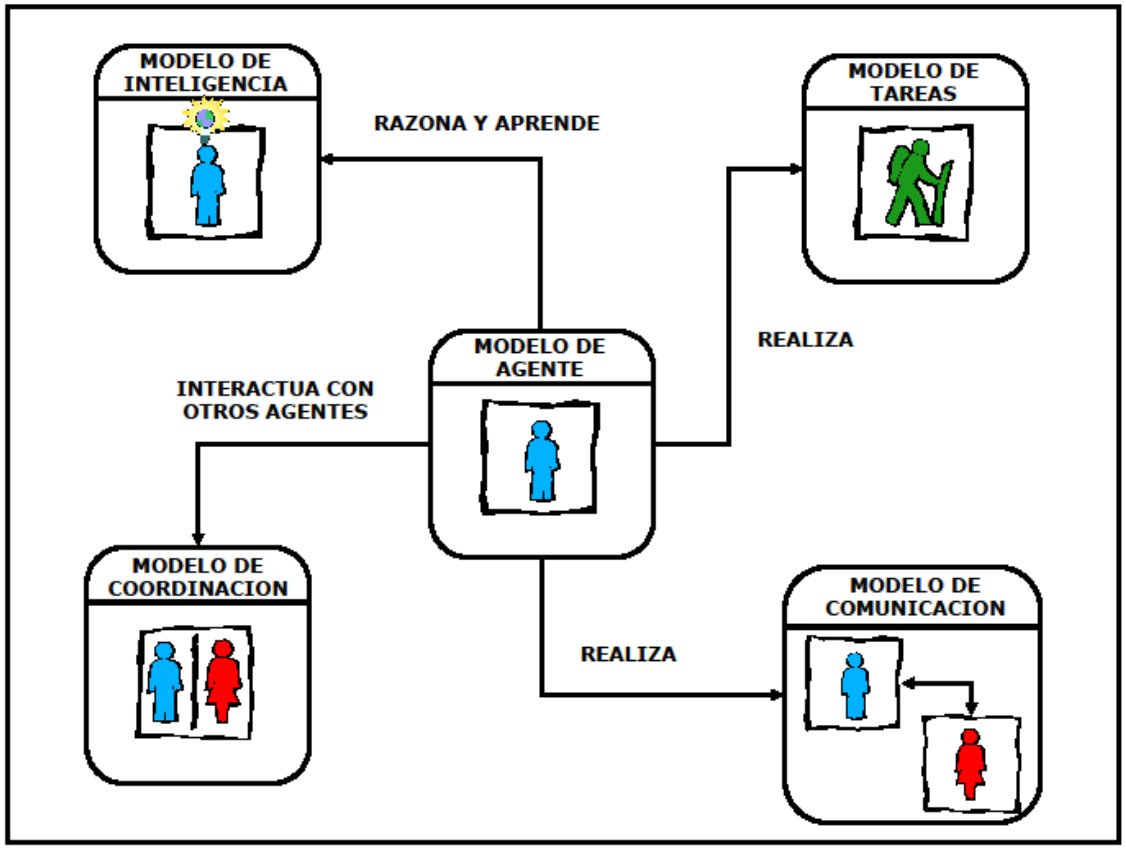
Diagrama de actividades



UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

MASINA

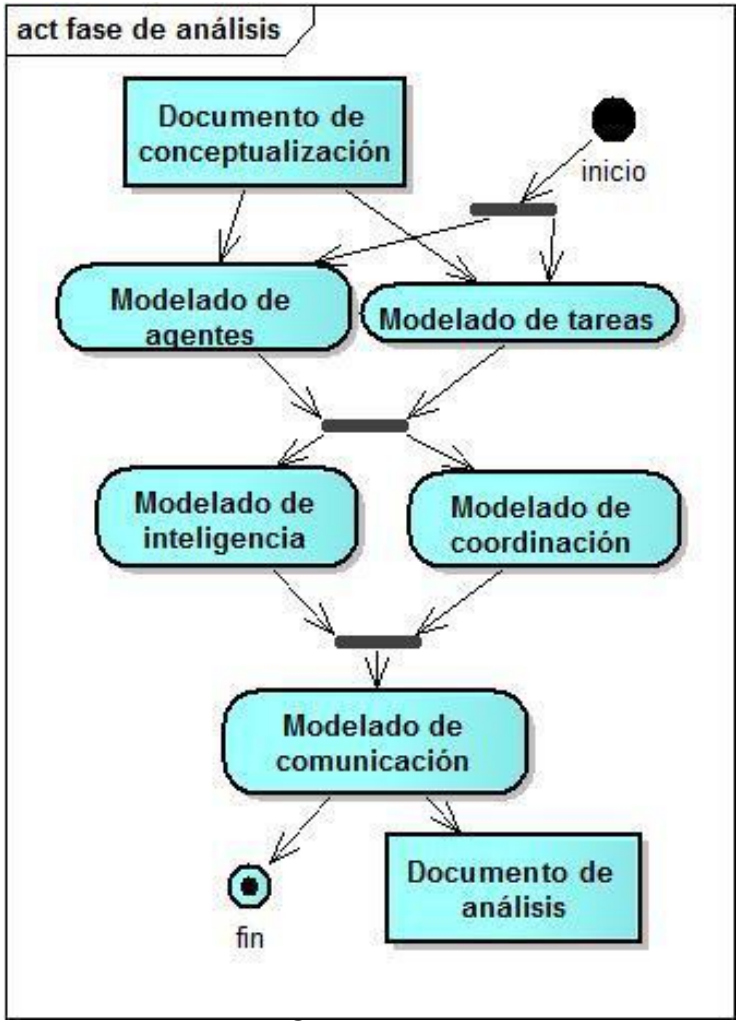
Fase de Análisis



UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

MASINA

Fase de Análisis



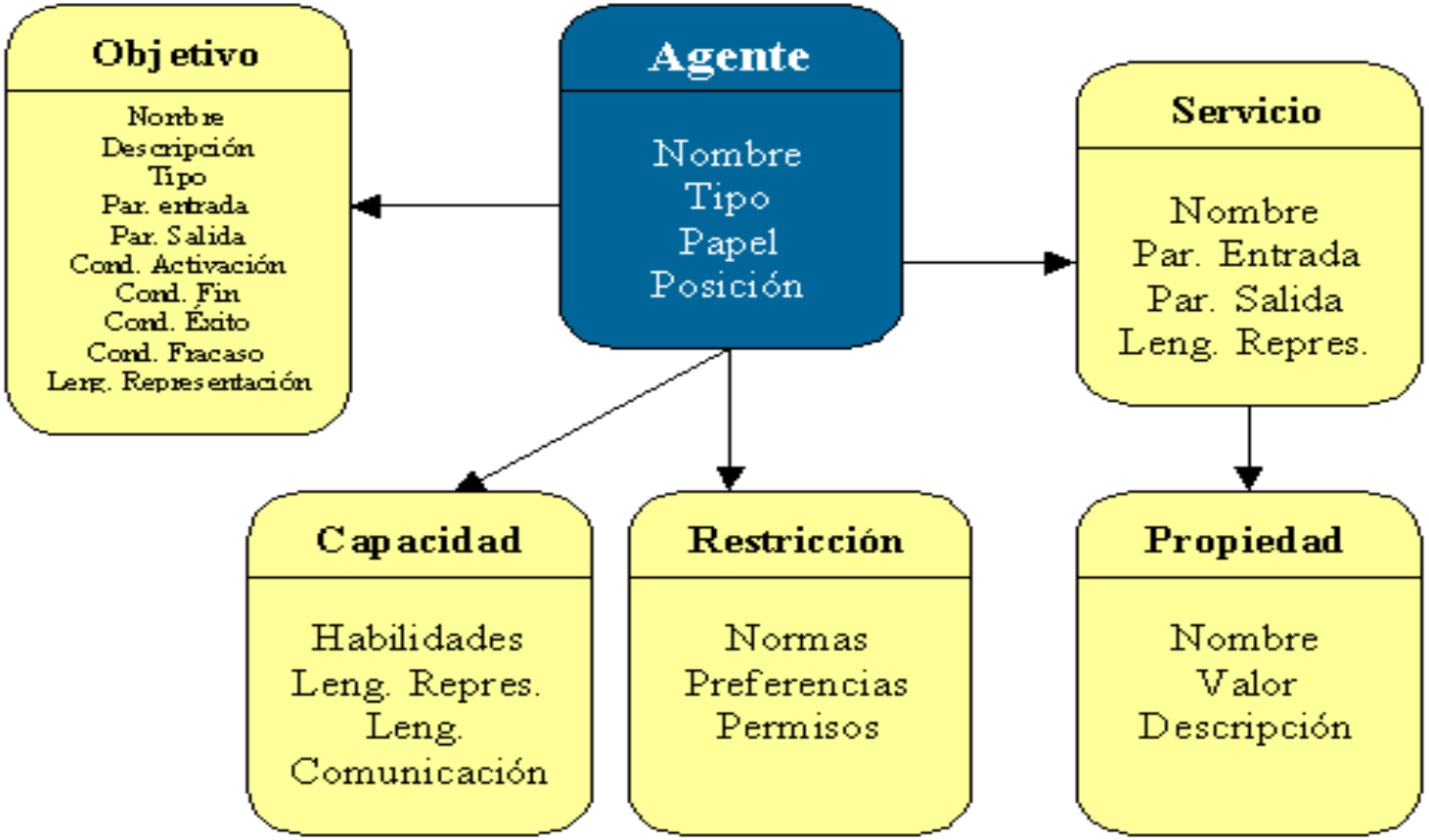
UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

MASINA

- El producto de esta fase es un **documento de análisis** contentivo de:
  - Modelo de agentes,
  - Modelo de tareas,
  - Tabla relación Agentes-Tareas
  - **Modelo de inteligencia**
  - **Modelo de Coordinación/conversación**
  - **Diagrama de Interacción**
  - **Modelo de comunicación.**

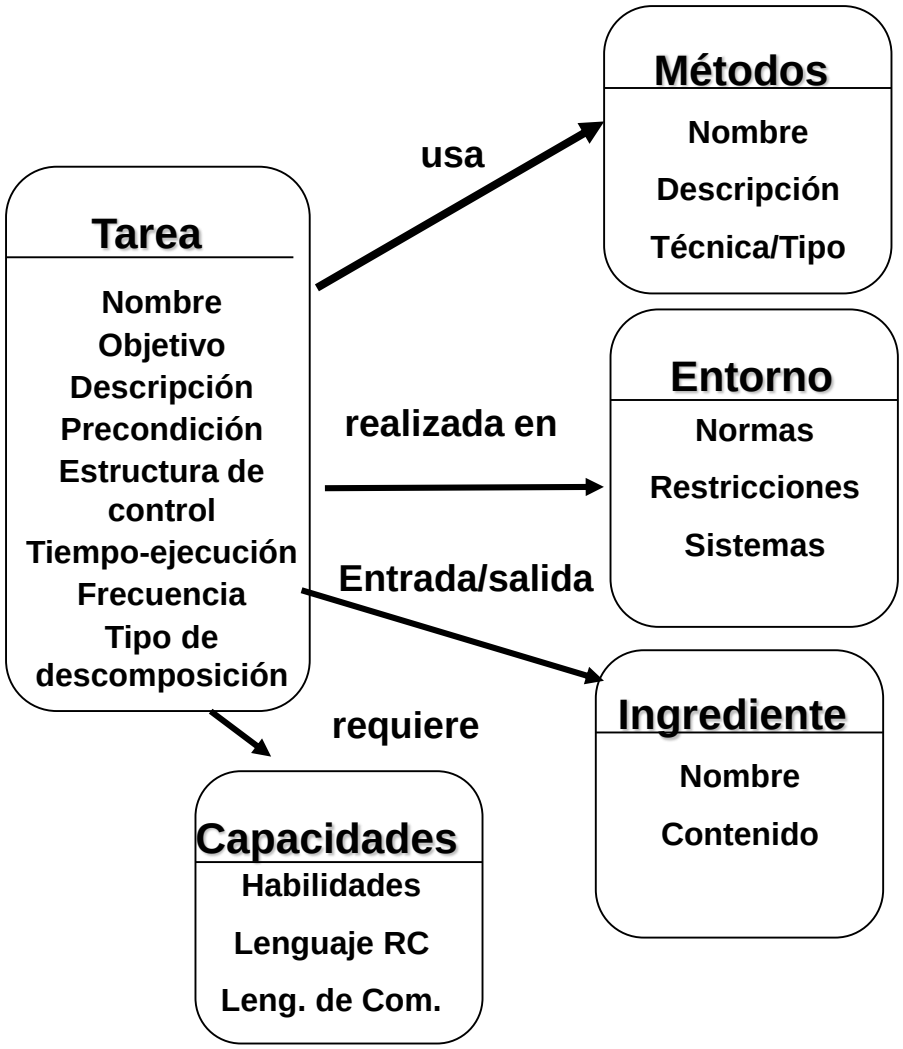
UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Modelo de Agente



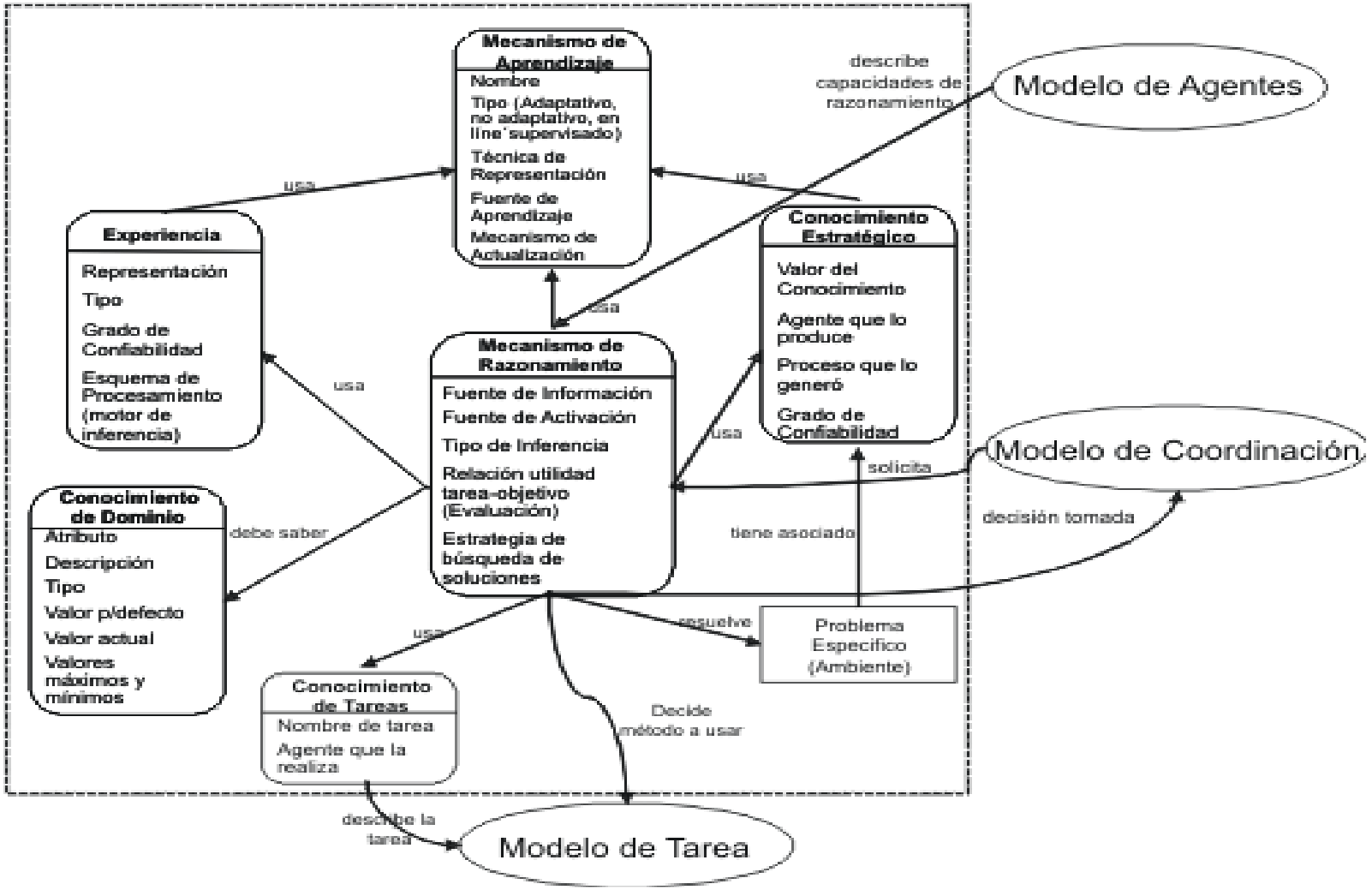
UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Modelo de Tareas

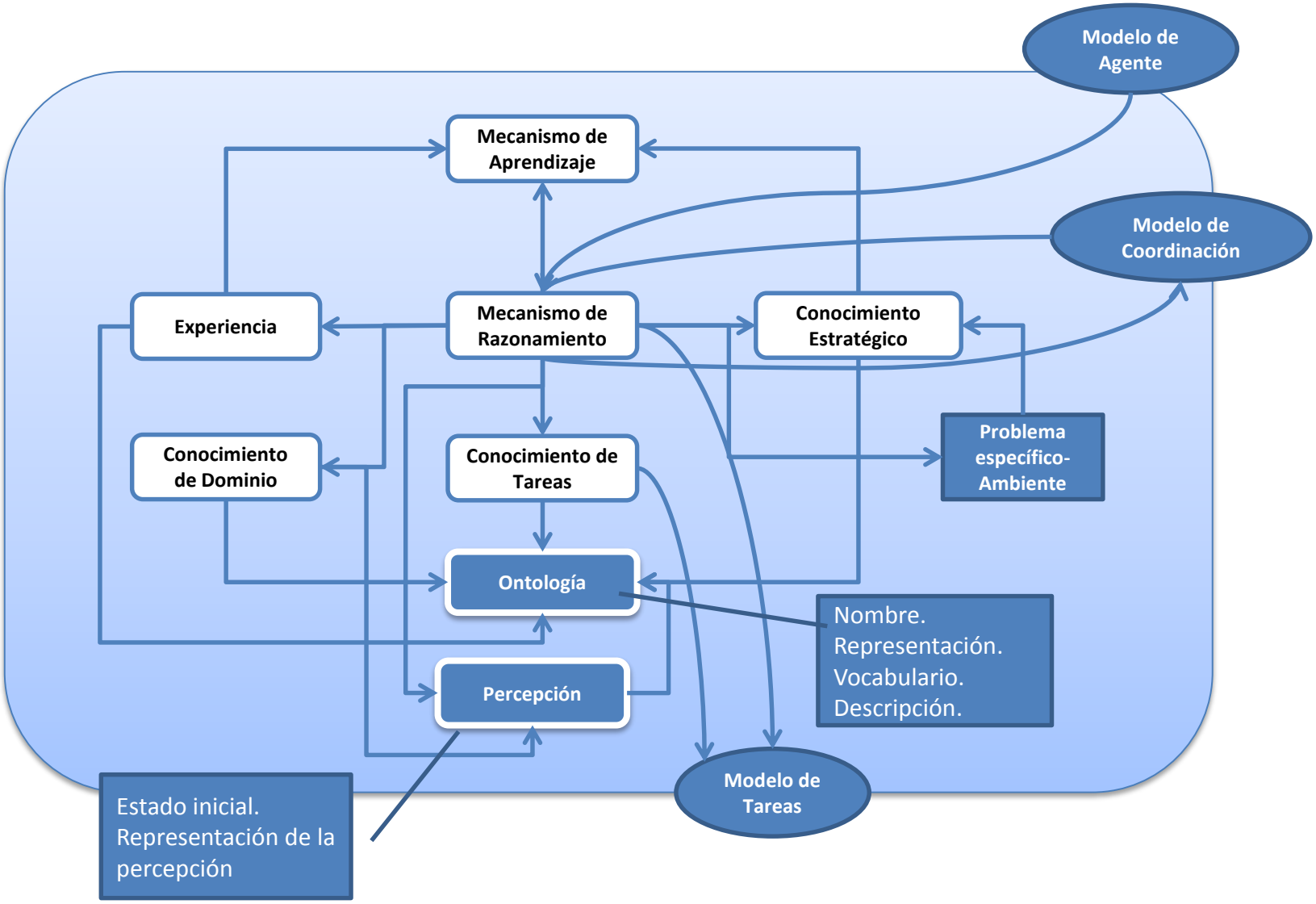


UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

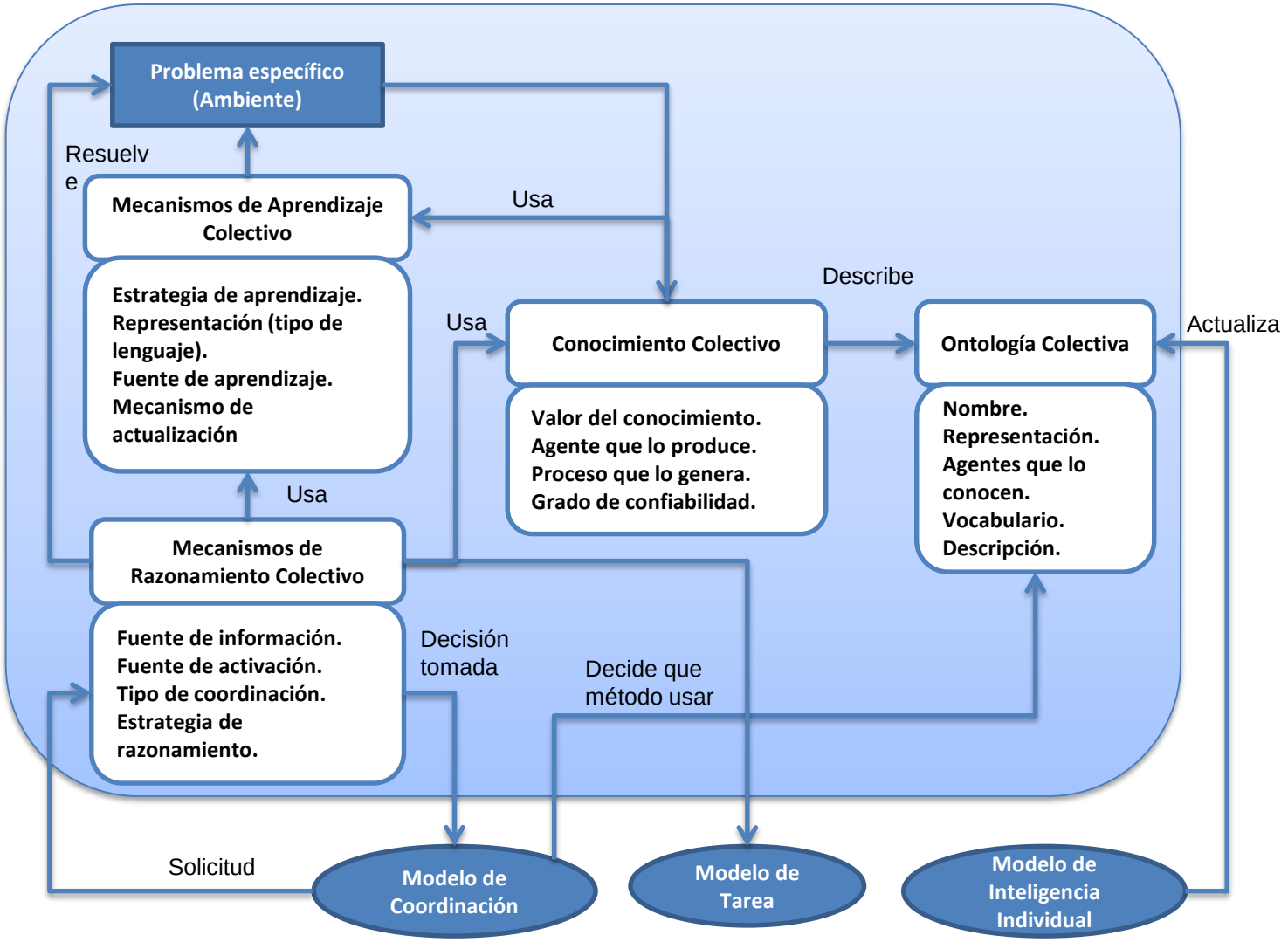
Modelo de Inteligencia



Modelo de Inteligencia individual de MASINA

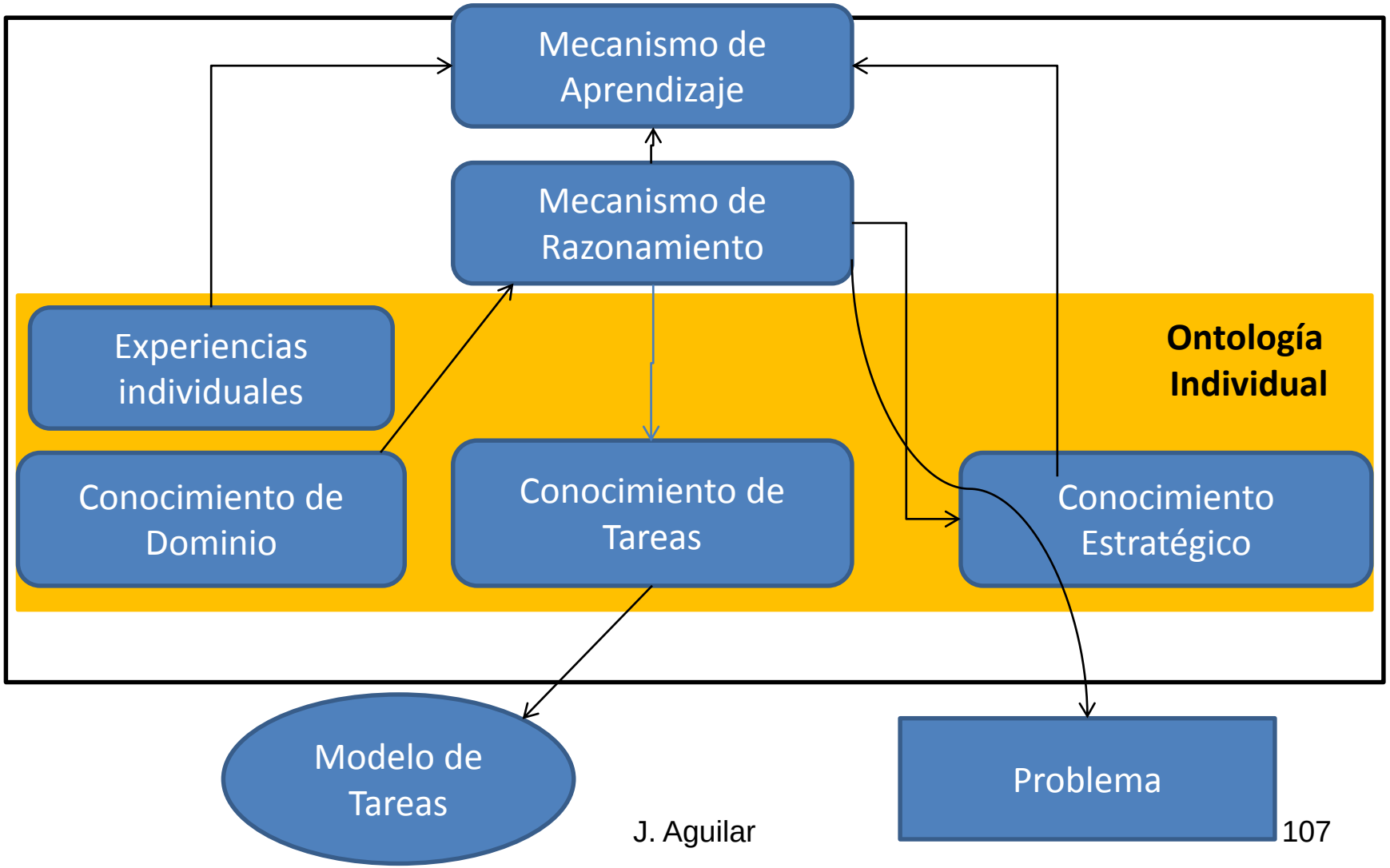


Modelo de Inteligencia Colectiva de MASINA



UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Modelo de Inteligencia



UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Modelo de Inteligencia

Mecanismo de Aprendizaje	
Nombre	Nombre del mecanismo de aprendizaje
Tipo	Supervisado, no supervisado, reforzamiento
Técnica de representación	Redes neuronales, Arboles de decisión, reglas, clasificación, redes bayesianas, minería de datos, etc.
Fuente de aprendizaje	Origen de la información que se usa para aprender. la misma normalmente se divide en una parte para entrenar y otra para probar
Mecanismo de actualización	hebbiano, corrección de error, etc.

Mecanismo de Razonamiento	
Fuente de información	Origen de la información utilizada para el proceso de razonamiento.
Fuente de alimentación	Tareas que requieren el proceso de razonamiento.
Técnica de inferencia	Lógica difusa, reglas, lógica de predicados, etc.
Lenguaje de representación de conocimiento	OWL, RDF, etc.
Relación tarea-inferencia (Resultado esperado)	Relación que existe entre la tarea que activo el proceso de razonamiento y el resultado obtenido.
Estrategias de razonamiento	Deductivo, inductivo, abductivo

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Modelo de Inteligencia

Experiencias (ontología histórica)	
Descripción	La experiencia en si del agente
Caracterización	Conceptos, relaciones, propiedades, etc
Fuente	Basada en casos, empírico, imitación, sus actividades, etc.
Valores por omisión (para cada concepto)	Valor inicial
Valores max y min	Valores máximos y mínimos que puede tomar la variable, rangos o bandas.
Momento (tiempo)	Manera como cambia en el tiempo.

Conocimiento de Dominio (ontología de dominio)	
Descripción	El conocimiento en si de un ámbito dado
Caracterización	Conceptos, relaciones, propiedades, etc.
Fuente	Taxonomía de un.área de conocimiento
Valores por omisión (para cada concepto)	Valor inicial
Valores max y min	Valores máximos y mínimos que puede tomar la variable, rangos o bandas.

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Modelo de Inteligencia

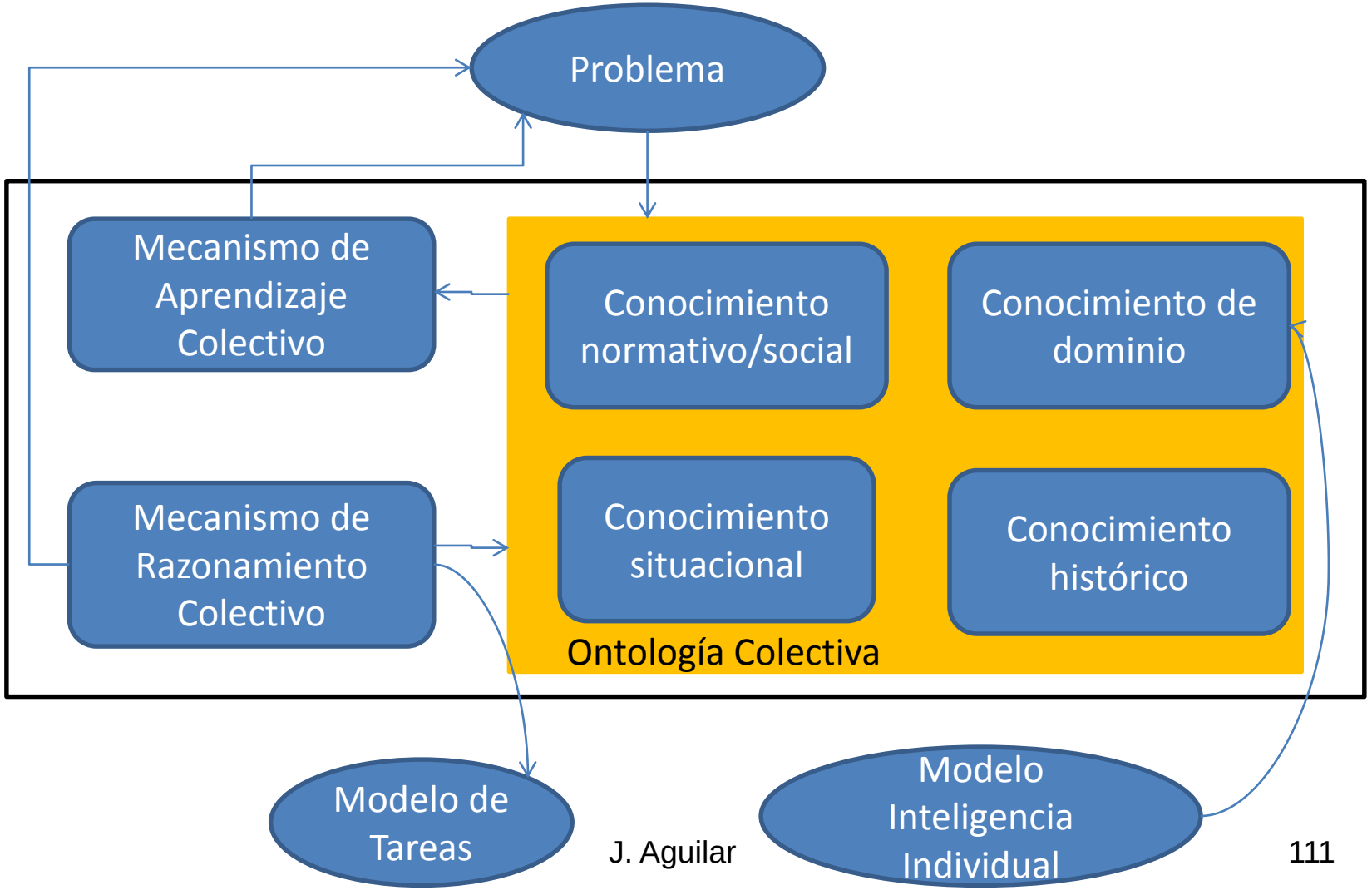
Ontología de contextualización o situacional: está compuesto por dos tipos de conocimiento

Conocimiento estratégico	
Valor del conocimiento	Pasos a seguir para resolver un problema, característica especificas de un ambiente, etc.
Agente o ambiente que lo produce	Agente que produce el conocimiento.
Proceso que lo genero (cuando sea el caso)	Proceso que genero el conocimiento.
Grado de confiabilidad	Nivel de certitud del conocimiento adquirido
Caracterización	Conceptos, relaciones, propiedades, etc.
Valores por omisión (para cada concepto)	Valor inicial
Valores max y min	Valores máximos y mínimos que puede tomar la variable, rangos o bandas.

Conocimiento de tareas	
Nombre de la tarea	Nombre de la tarea a realizar.
Agente que la realiza	Agente que origina la tarea a realizar.

Marco Ontológico del individuo	
Nombre	Nombre de la ontología
Descripción	Descripción de “lo que conoce el agente”
Ontologías que la integran	Listado de conocimiento (ontologías) que lo componen

Modelo de Inteligencia Colectiva



Modelo de Inteligencia Colectiva

Mecanismo de Aprendizaje Colectivo	
Nombre	Nombre del mecanismo de aprendizaje
Tipo	Supervisado, no supervisado, reforzamiento
Técnica de representación	Redes neuronales, Arboles de decisión, reglas, clasificación, redes bayesianas, minería de datos, etc.
Fuente de aprendizaje	Origen de la información que se usa para aprender. la misma normalmente se divide en una parte para entrenar y otra para probar
Mecanismo de actualización	hebbiano, corrección de error, etc.

Mecanismo de Razonamiento Colectivo	
Fuente de información	Origen de la información utilizada para el proceso de razonamiento.
Fuente de alimentación	Tareas que requieren el proceso de razonamiento.
Técnica de inferencia	Lógica difusa, reglas, lógica de predicados, etc.
Lenguaje de representación de conocimiento	OWL, RDF, etc.
Relación tarea-inferencia (Resultado esperado)	Relación que existe entre la tarea que activo el proceso de razonamiento y el resultado obtenido.
Estrategias de razonamiento	Deductivo, inductivo, abductivo

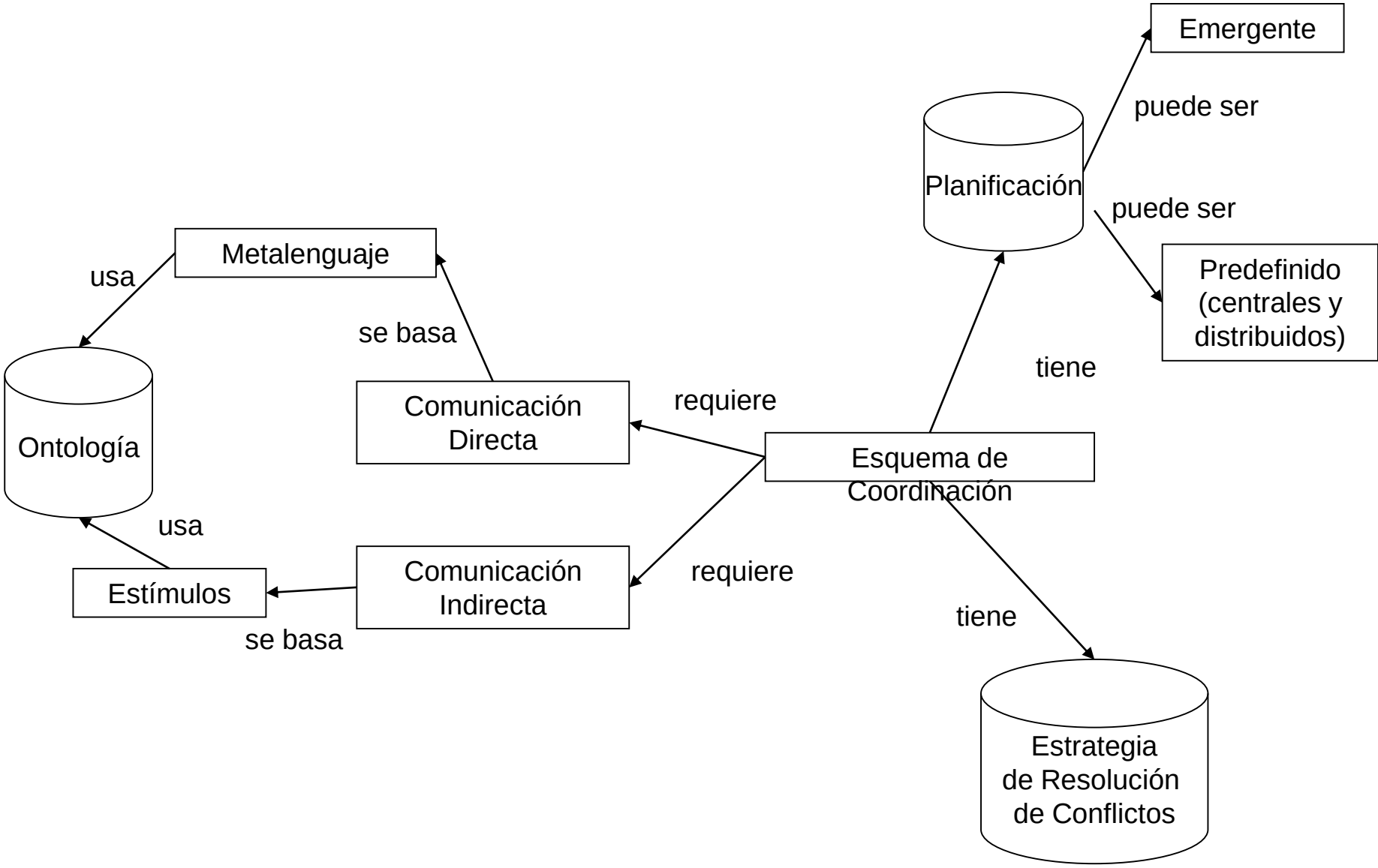
Modelo de Inteligencia Colectiva

Ontología situacional colectiva	
Descripción	Conocimiento colectivo
Caracterización	Conceptos, relaciones, propiedades, etc
Fuente	Quien la genera, quien la actualiza, en el SMA
Grado de confiabilidad	Nivel de certitud del conocimiento adquirido
Caracterización	Conceptos, relaciones, propiedades, etc.
Valores por omisión (para cada concepto)	Valor inicial
Valores max y min	Valores máximos y mínimos que puede tomar la variable, rangos o bandas.

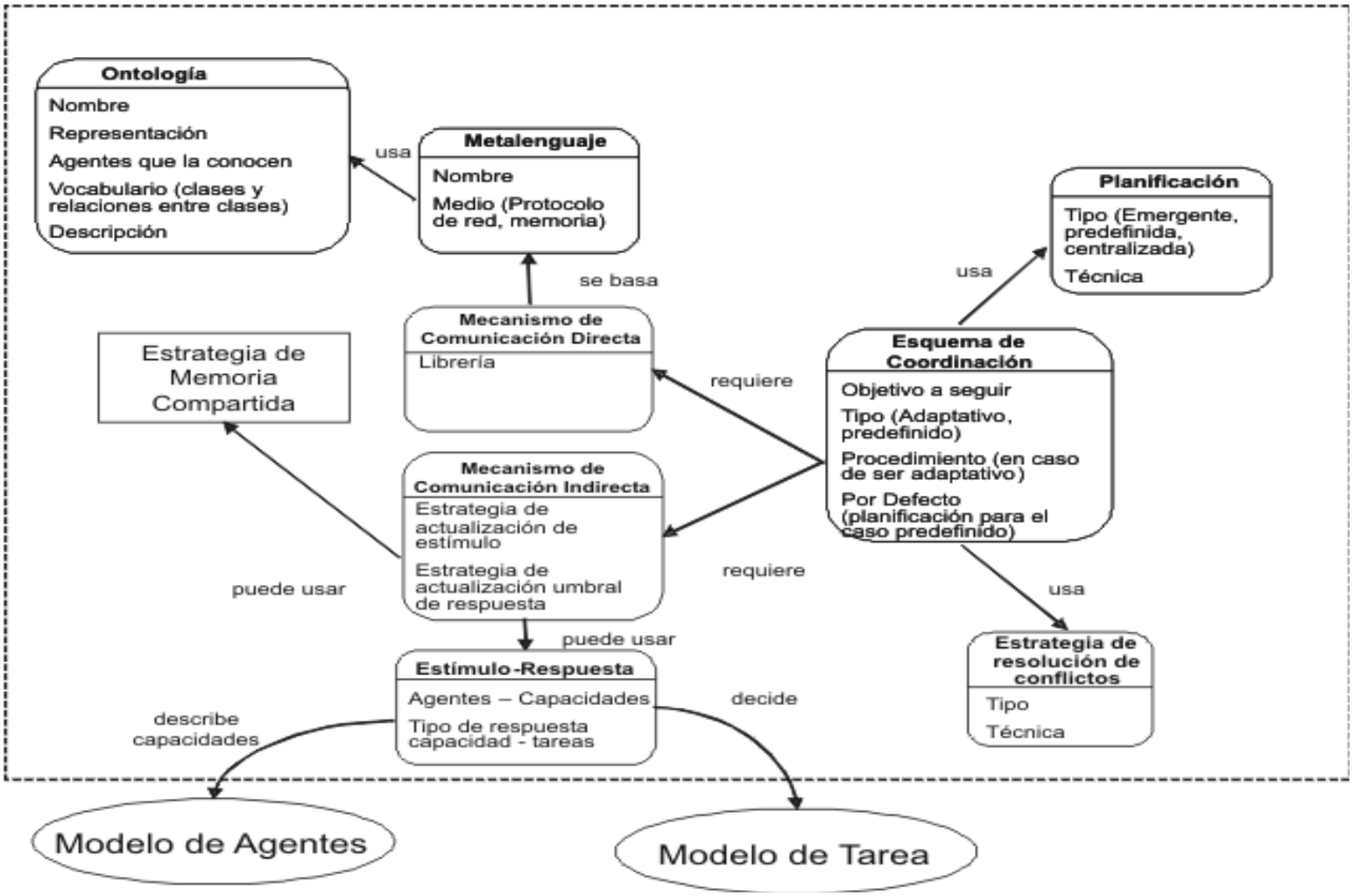
Marco Ontológico colectivo	
Nombre	Nombre de la ontología
Objetivos colectivos para lo que se usa	Qué tareas colectivas lo usan para alcanzar objetivos grupales
Descripción	Descripción de “lo que conoce el agente”
Ontologias que la integran	Listado de ontologías situacionales que lo componen
Agentes que la conocen	Miembros del SMA que usan esa ontología en sus tareas colectivas

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Problema de Coordinación



Modelo de Coordinación



Modelo de Coordinación y Comunicación

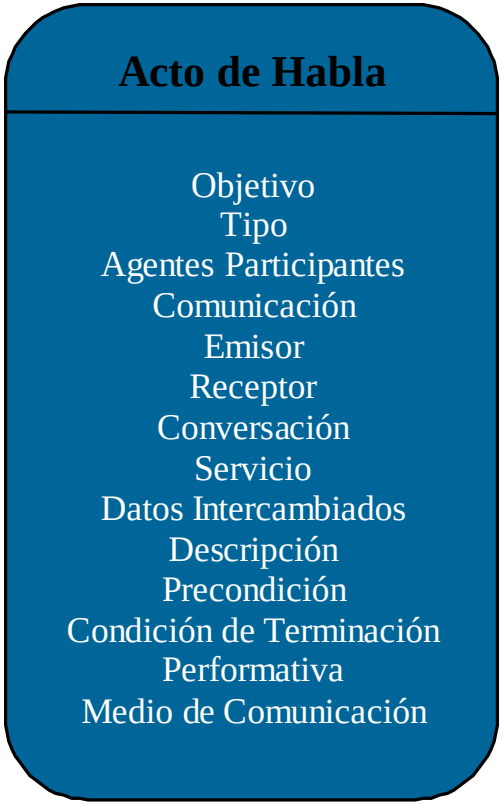
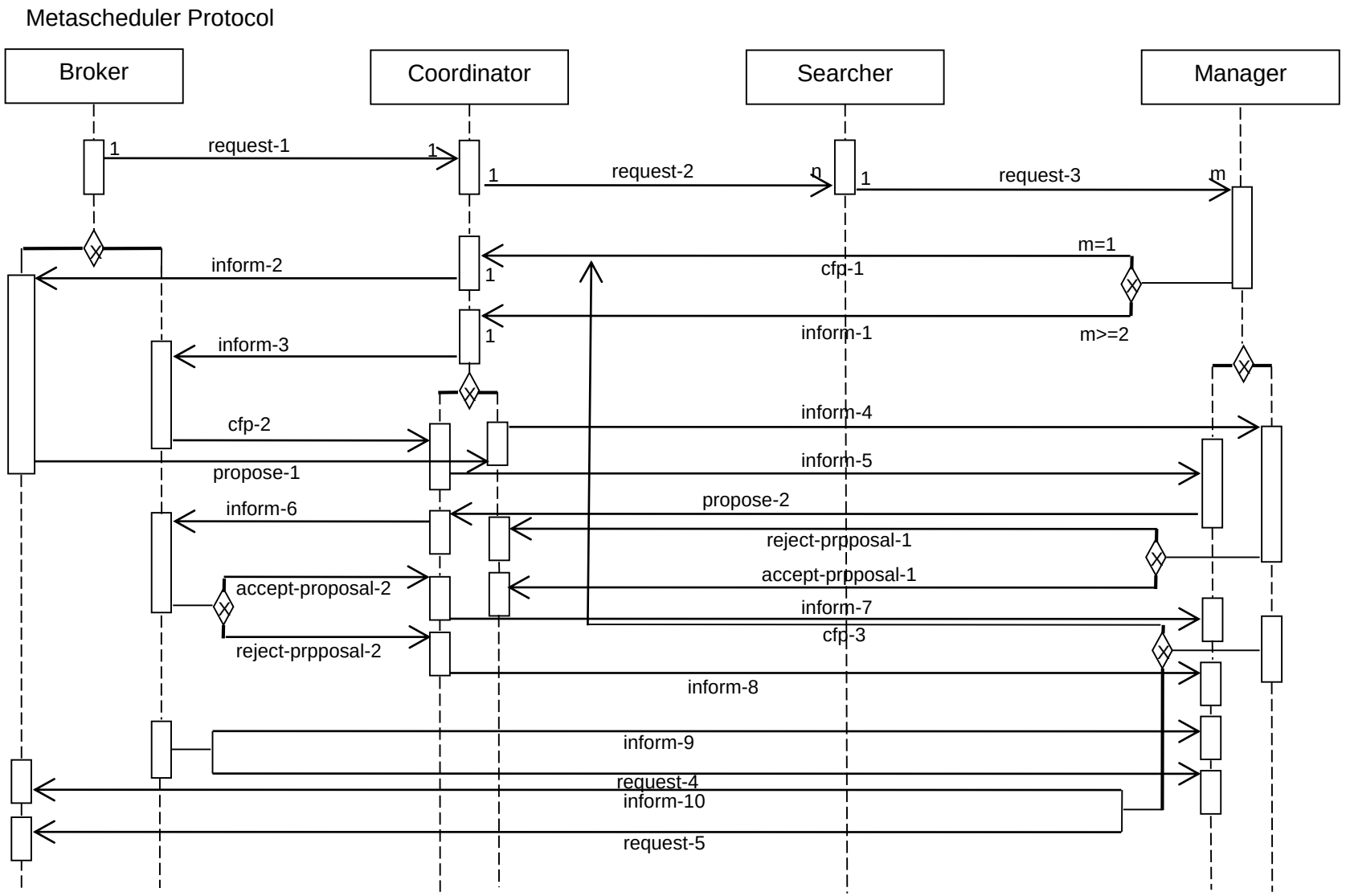


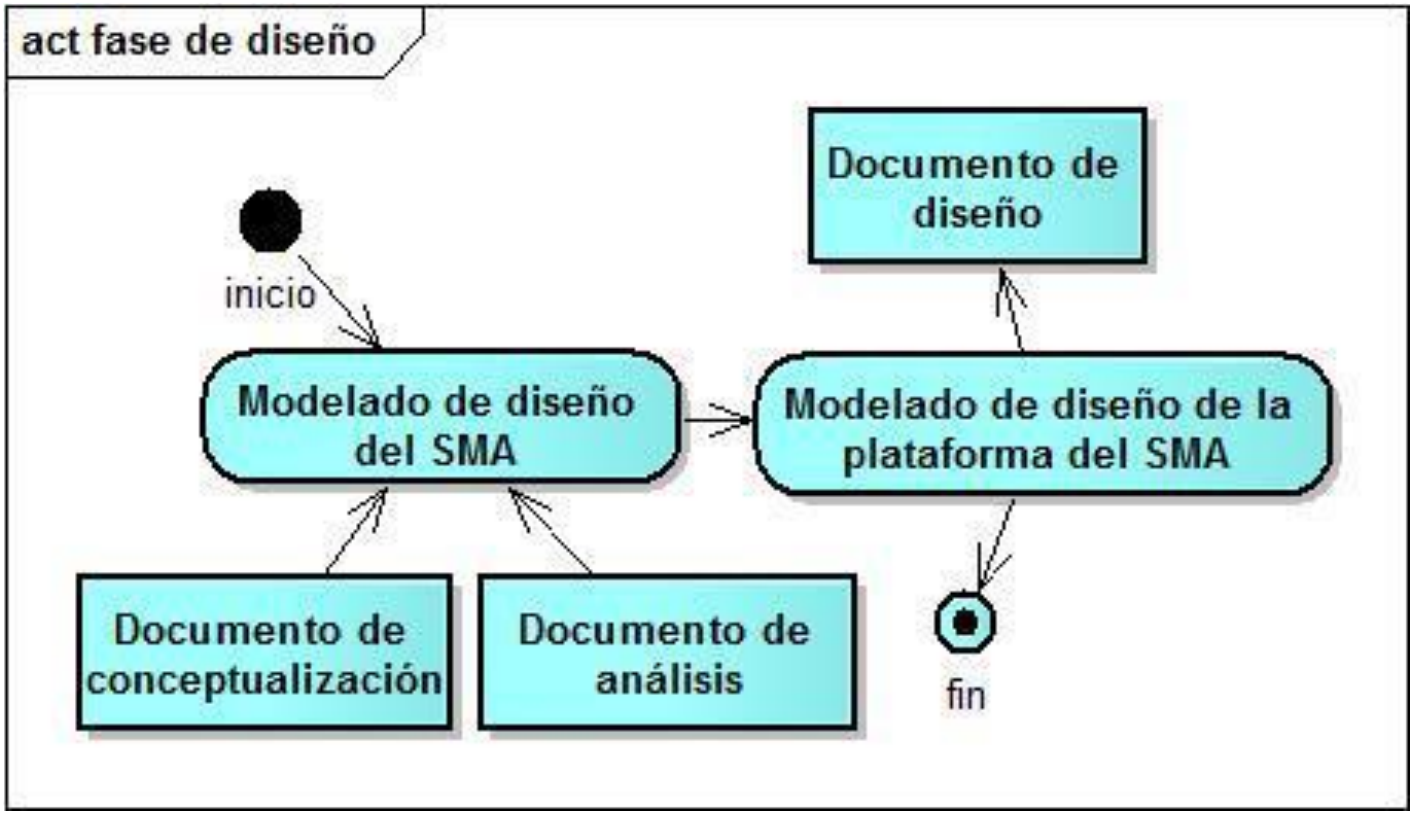
Diagrama Interacción: Protocolo de interacción FIPA



UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

MASINA

Fase de diseño



UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

MASINA

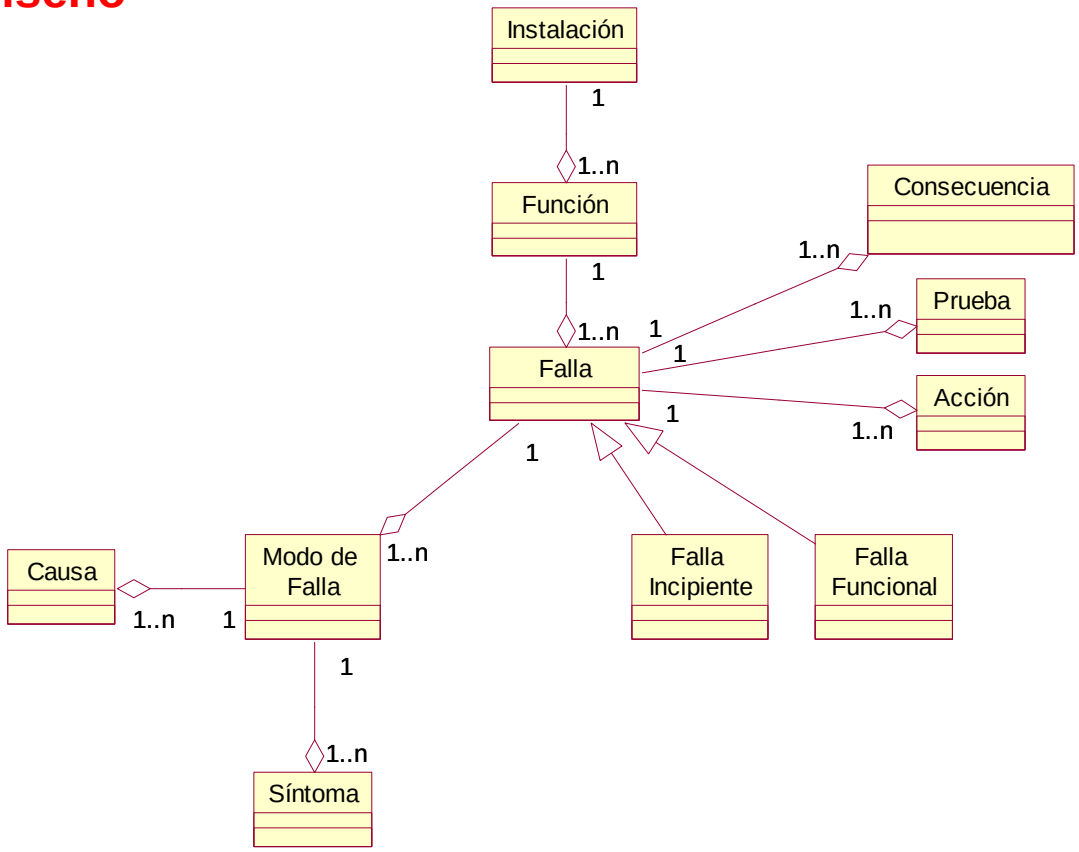
- El producto de esta fase es un **documento que especifica**:
  - El universo de clases del sistema.
  - Para cada una de las clases descritas en el universo de clases su especificación formal.
  - Para cada uno de los métodos que componen las clases su especificación formal.

Nosotros particularmente usamos TDSO

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

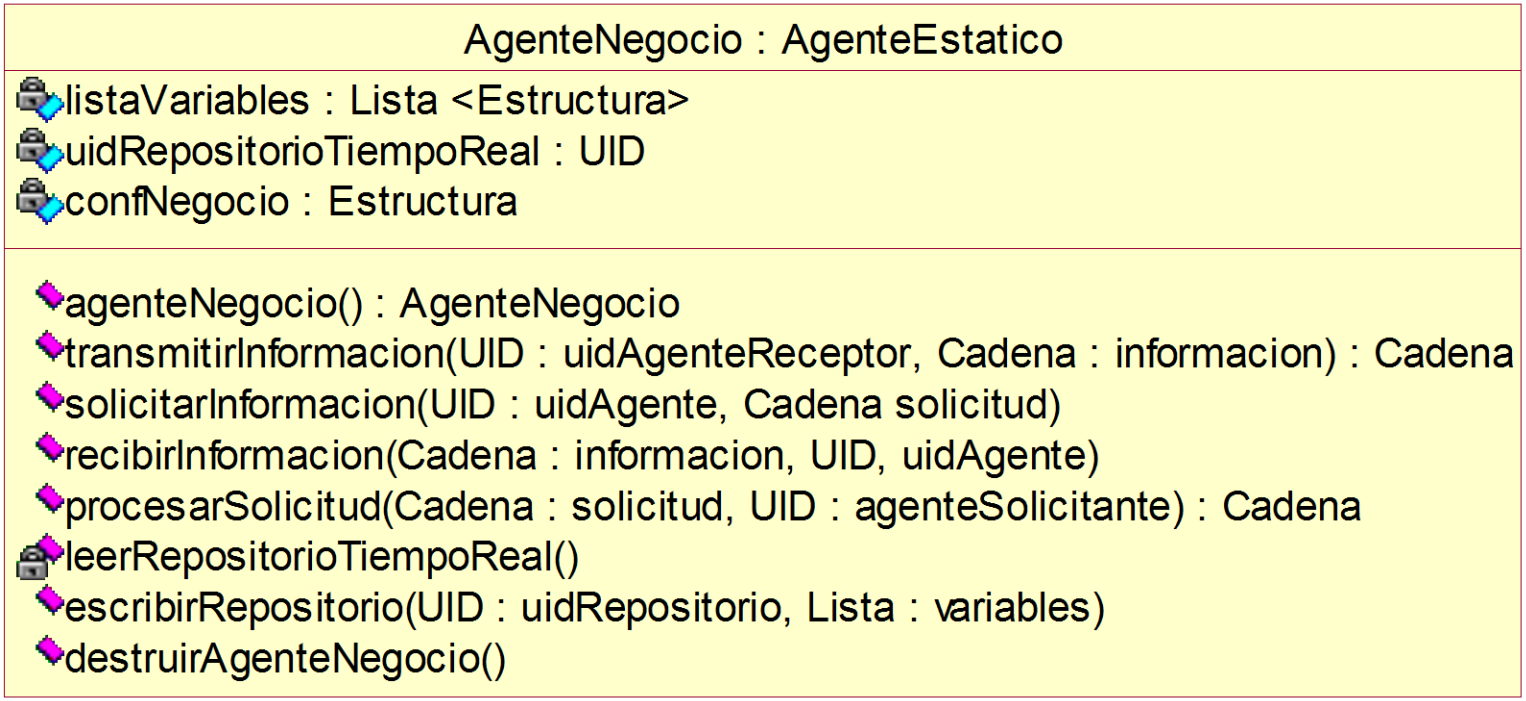
MASINA

Modelo de Diseño



UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Diagrama de clases del Agente de Negocio



Definición del universo de clases y tipo de datos abstractos (TDAs)

20/11/07		Versión 1.0
Universo de clases y TDAs AgenteNegocio {Colección de clases y TDAs requerida para implantar el Agente de Negocio}		
1 2 3 4 5 6	Agente Negocio ( ) Cadena UID Logico TablaTiempoReal Estructura	*AgenteNegocio ( )*: clase que permite la creación de un Agente de Negocio. *Cadena*: TDA cadena de caracteres de longitud variable. *Entero*: valor entero. *UID*: tipo entero que representa un identificador único en el sistema multiagente. *Logico*: tipo lógico, conformado por los valores cierto y falso. *TablaTiempoReal:* TDA que contiene los datos del proceso real. *Estructura:* tipo de dato que contiene campos asociados a información configurada.

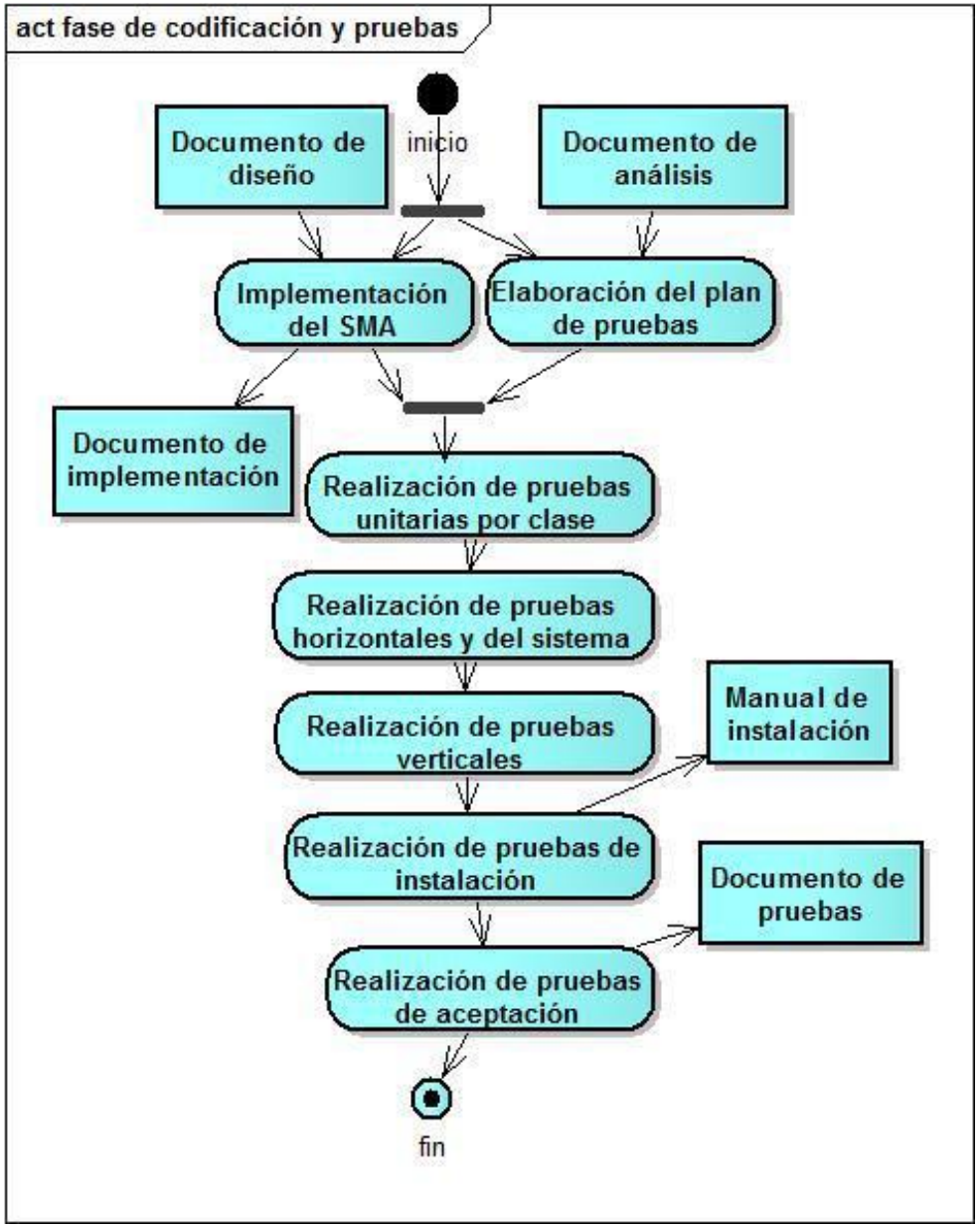
Definición del universo de clases y tipo de datos abstractos (TDAs)

20/11/07		Versión 1.0
1,1 (Constructor, Público) **agenteNegocio( ): AgenteNegocio** {Crea un Agente del tipo AgenteNegocio} {**pre:** existencia de memoria y Negocio.dat distinto de Null}		{**pos:** se crean componentes del agente o error}
1 2 3 4	abrirArchivoConf("Negocio.dat") Leer(Negocio.dat, ConfNegocio) uidRepositorioTiempoReal = asignaUid( ) listaVariables: Lista<TablaTiempoReal>	*asignaUid()*: método que asocia el AN con un único repositorio datos de tiempo real *listaVariables:* Lista que contiene los datos del proceso real
1 2	agenteNegocio x ⇒ se creó el AN agenteNegocio x ⇒ error	Se instancia el agente x, si éste se puede crear hay éxito, por el contrario hay error.

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

MASINA

Fase de Codificación Y pruebas



UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

MASINA

- El producto principal de esta fase consiste en un sistema de ingeniería orientado a agentes

Resumen de otras metodologías

- ***GAIA*.** se centra en la idea de que la construcción de sistemas basados en agente es un proceso de diseño organizacional.
- ***DESIRE***. La principal contribución es que constituye un entorno lo suficientemente expresivo para permitir a los diseñadores de sistemas multiagente centrarse en el diseño conceptual y la especificación de su sistema.
- ***MASSIVE*** (Multi-Agent SystemS Iterative View Engineering) está constituido por un conjunto de vistas diferentes del sistema a construir donde el desarrollo que se sigue consiste en una visión iterativa del mismo.
- ***AUML*.** intenta adaptar herramientas de desarrollo ya existentes y que están teniendo éxito para aplicaciones industriales reales, como es el caso de UML, tratando de orientarlas hacia el campo de los agentes.

Resumen de otras metodologías

- ***Tropos*.** metodología de desarrollo de software basado en agentes mediante extensiones de UML y empleando un entorno de modelado denominado *i** . El concepto principal es el de actor, así como sus objetivos y posibles dependencias con otros actores.
- ***MaSE*** (Multiagent System Engineering). Es una metodología que parte de la especificación del mismo hasta su implementación. Lenguaje de especificación basado en UML+OCL
- ***MESSAGE*** (Methodology for Engineering Systems of Software Agents): incorpora técnicas de ingeniería del software cubriendo el análisis y diseño de sistemas multiagente. La metodología provee un lenguaje, un método y unas guías de cómo aplicar la metodología, centrándose en las fases de análisis y diseño

Middleware (medio de gestión de servicios) o herramientas para el despliegue de los SMA

Middleware para SMA

Un cliente de un MGS para agentes aspira:

- **Mecanismos genéricos**: a nivel de protocolos de cooperación, de comunicación, de negociación, de coordinación, entre otros;
- **Capas de bajo nivel para los SMA**: primitivas de comunicación entre agentes (KQML, ACL, entre otros), mecanismos que permitan a los agentes entrar (registrarse) y salir (desregistrarse) del SMA, duplicarse (crear otras instancias), migrar (moverse a otro nodo), etc.
- **Motores que implementen el ciclo básico de comportamiento de los agentes**, incluyendo lenguajes de ontologías, sistemas de razonamiento, etc.
- **Acceso a los recursos disponibles**: mecanismos de comunicación de bajo nivel (sockets, RPC), mecanismos de procesamiento distribuido (hebras),
- etc.
- **Usar el concepto de agentes** como una abstracción.
- **Centrarse en los detalles del comportamiento de alto nivel del agente**.
- **Servicios de nombramiento, de transporte de mensajes y de paginas amarillas**.

Middleware para SMA

Algunos basados en FIPA son:

- JADE
- JIAC
- JACK
- Zeus
- Fipa-OS
- MGS

JADE

- JADE (*Java Agent DEvelopment Framework*) es un software para desarrollar aplicaciones basadas en agentes que sigue las especificaciones de FIPA,
- El objetivo de JADE es simplificar el desarrollo de agentes, garantizando seguir los estándares establecidos
- por la FIPA, a través de un conjunto de servicios ofrecidos por la plataforma.
  - Plataforma para la gestión de los agentes y
  - Un IDE (*Integrated Development Environment*) para el desarrollo de agentes.
  - Paquete basado en Java para desarrollar agentes.
  - Varias extensiones para ejecutarse en dispositivos con aplicaciones especificas (PDA, teléfonos inteligentes, etc.).

JADE

- La plataforma de agentes se puede distribuir en varios *hosts*. Es suficiente una Maquina Virtual Java (JVM) en cada host.

- La plataforma JADE soporta la coordinación de múltiples agentes FIPA y proporciona una implementación estándar del lenguaje de comunicación FIPA-ACL,

- Para la implantación de SMA, JADE proporciona:
  - Un entorno de ejecución en el que los agentes se ejecutan.
  - Bibliotecas de clases para la creación de agentes mediante la herencia y la redefinición de comportamientos.
  - Un conjunto de herramientas gráficas para el monitoreo y la administración de la plataforma.

JADE

JADE tiene un contenedor principal que tiene dos agentes especiales:

- *DF (Directory Facilitator):*
- *AMS (Agent Management System): register() takedown()*, y
- *deregister()* del AMS.

JADE define la Clase *Agent*, la cual es una superclase que permite a los usuarios crear agentes

Esta clase suministra métodos que permiten ejecutar las tareas básicas de los agentes como:

- Pasar mensajes utilizando objetos ACLMessage
- Dar soporte al ciclo de vida de un agente.
- Planificar y ejecutar m´ múltiples actividades al mismo tiempo.

JADE

JADE tiene un contenedor principal que tiene dos agentes especiales:

- *DF (Directory Facilitator):*
- *AMS (Agent Management System): register() takedown()*, y
- *deregister()* del AMS.

JADE define la Clase *Agent*, la cual es una superclase que permite a los usuarios crear agentes

Esta clase suministra métodos que permiten ejecutar las tareas básicas de los agentes como:

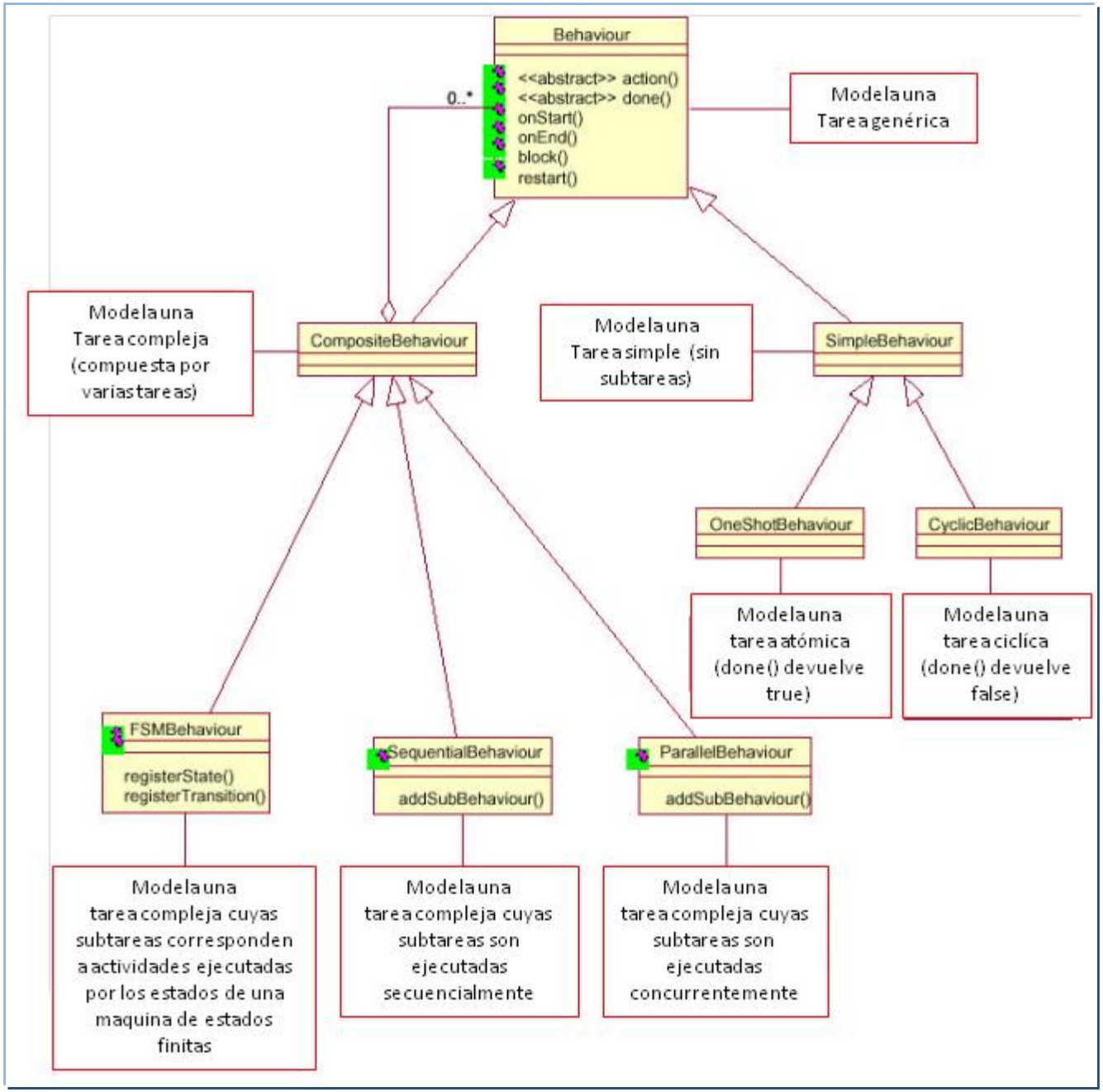
- Pasar mensajes utilizando objetos ACLMessage
- Dar soporte al ciclo de vida de un agente.
- Planificar y ejecutar m´ múltiples actividades al mismo tiempo.

JADE

- El “comportamiento” de un agente define las acciones a realizar bajo un determinado evento y hereda de la clase agentes los métodos: *addBehaviour, removeBehaviour*.
- Los diferentes comportamientos que el agente adopta se definen a partir de la clase abstracta *Behaviour*, para lo cual usa el método *action(), done(), reset(),*
- JADE incluye prototipos de comportamientos listos para utilizarse, tales como los protocolos de interacción FIPA, despertar bajo una cierta condición, etc.

- los agentes se implementan como un hilo por agente.
- Además de la solución multi-hilo, ofrecidos directamente por Java, JADE también admite la programación de comportamientos cooperativos

JADE


Behaviour
<<abstract>> action()
<<abstract>> done()
onStart()
onEnd()
block()
restart()
0..*
Modela una Tarea genérica
Modela una Tarea compleja (compuesta por varias tareas)
CompositeBehaviour
Modela una Tarea simple (sin subtareas)
SimpleBehaviour
OneShotBehaviour
CyclicBehaviour
Modela una tarea atómica (done() devuelve true)
Modela una tarea cíclica (done() devuelve false)
FSMBehaviour
registerState()
registerTransition()
SequentialBehaviour
addSubBehaviour()
ParallelBehaviour
addSubBehaviour()
Modela una tarea compleja cuyas subtareas corresponden a actividades ejecutadas por los estados de una maquina de estados finitas
Modela una tarea compleja cuyas subtareas son ejecutadas secuencialmente
Modela una tarea compleja cuyas subtareas son ejecutadas concurrentemente

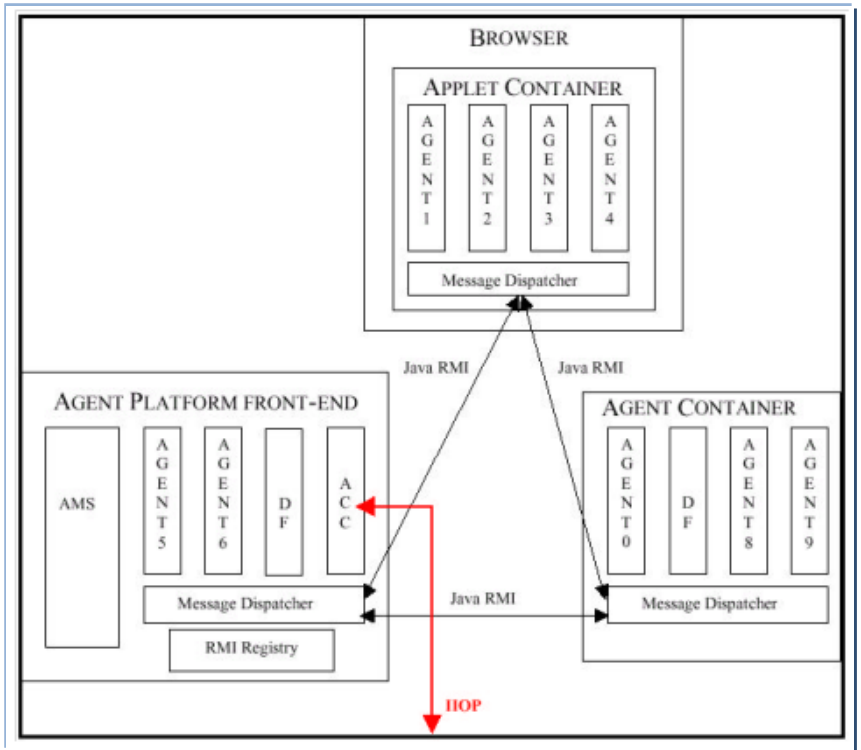
JADE

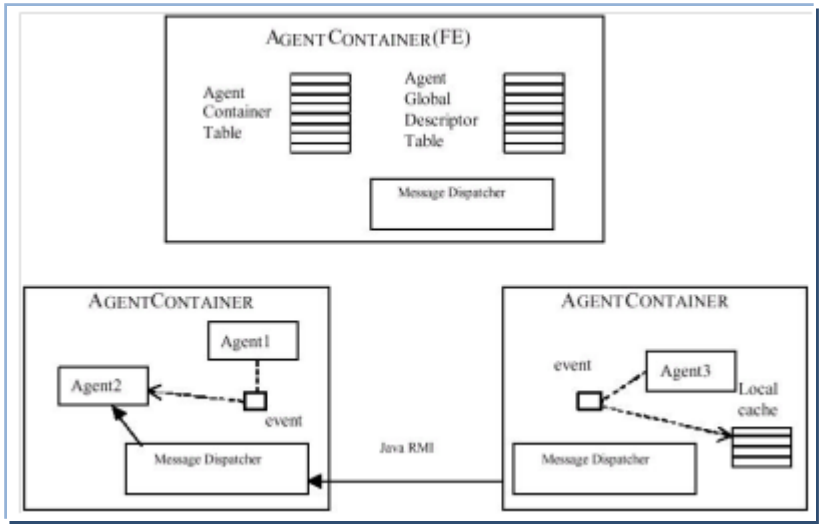
- En el caso de las ontologías, un sistema de gestión de ontología de agentes ha sido desarrollado, así como el soporte a lenguajes y ontologías que pueden ser implementadas y registradas con los agentes.
- JADE ofrece una llamada a *JessBehaviour* que permite la integración con JESS, un entorno para la programación de reglas con su motor de razonamiento

- Cada agente vive en un contenedor, que es una maquina virtual de Java, el cual provee un completo ambiente de ejecución para los agentes, permitiendo que varios de ellos puedan ejecutarse concurrentemente, controlando sus ciclos de vida (crearlos, suspenderlos, eliminarlos, etc.) y comunicaciones (envió y enrutamiento de mensajes).

JADE

- Existe un contenedor especial, ***llamado front-end (FE),*** que ejecuta los agentes de administración
- Existe otro contenedor especial para mostrar la interfaz Web. Este contenedor ejecuta un agente llamado ***Remote Management Agent (RMA)***

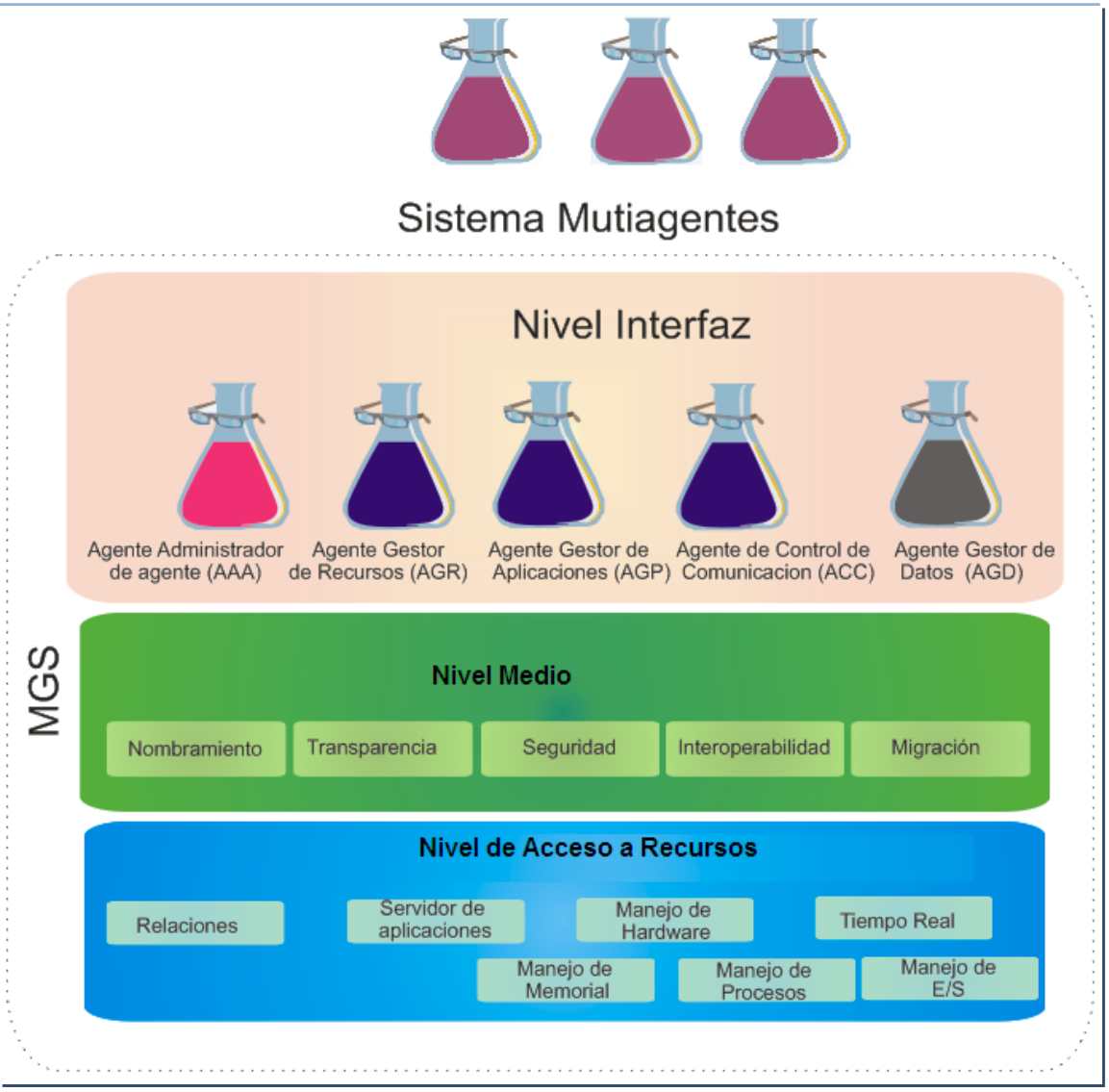




JADE

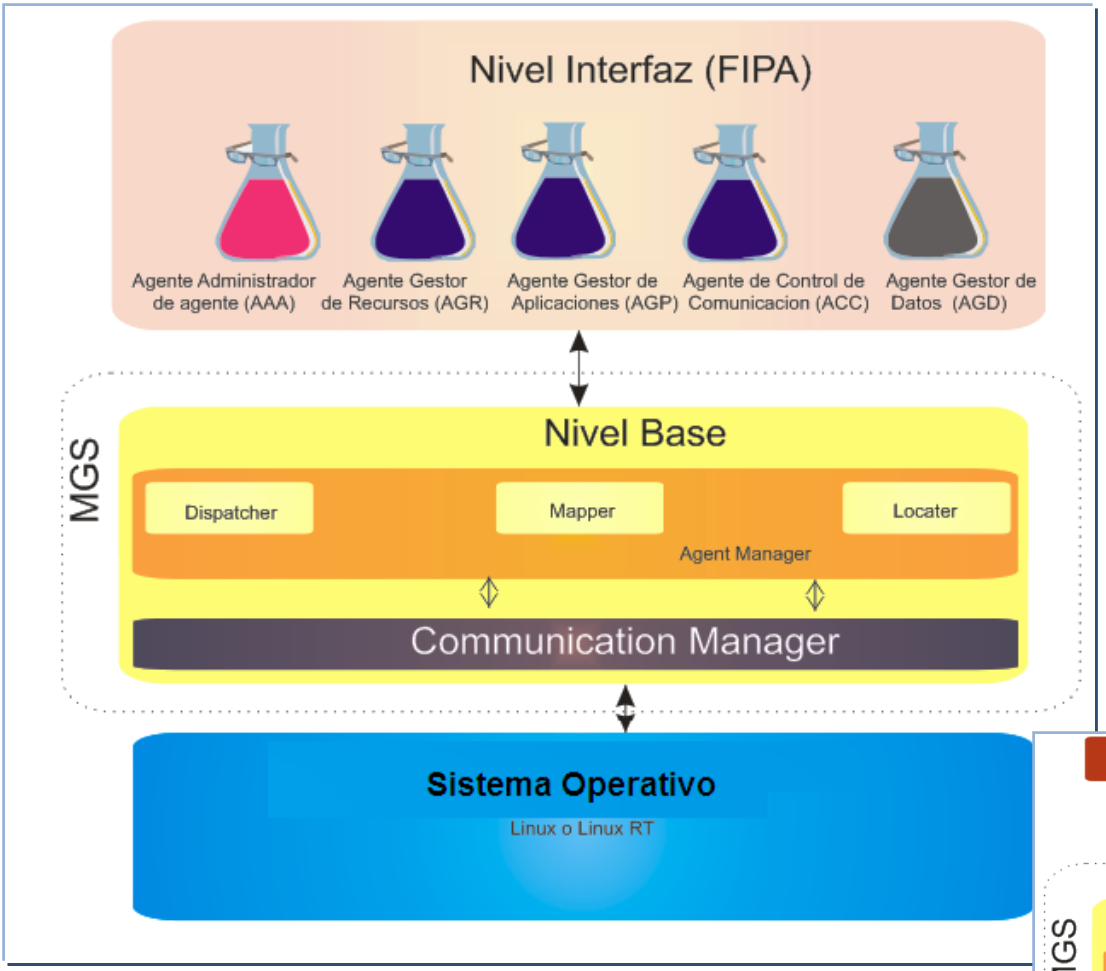
- **El agente *dummy*** es una herramienta para la inspección de los intercambios de mensajes entre los agentes. Este agente facilita la prueba y validación de la interfaz de los agentes antes de su integración en un SMA.
- **El agente *sniffer***, a través de su IGU, permite el seguimiento de los mensajes intercambiados en una plataforma de agentes JADE. Cuando el usuario decide rastrear un agente o un grupo de agentes, cada mensaje dirigido a, o procedente de, el agente o grupo de agentes, se sigue y se muestra en la ventana del *sniffer*, utilizando una notación similar a los diagramas de secuencia de UML.
- **El agente *introspector*** permite monitorizar y controlar el ciclo de vida de un agente en ejecución, y sus mensajes intercambiados, en particular, su cola de mensajes enviados y recibidos.

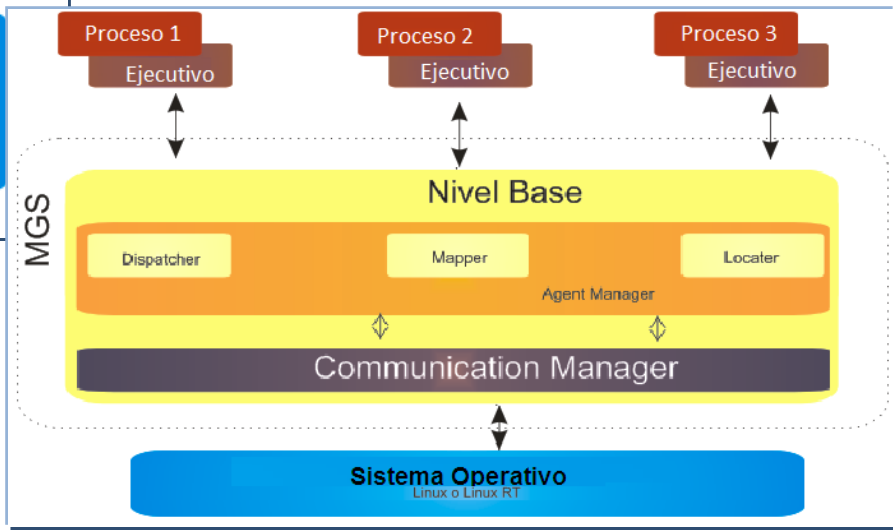
MGS

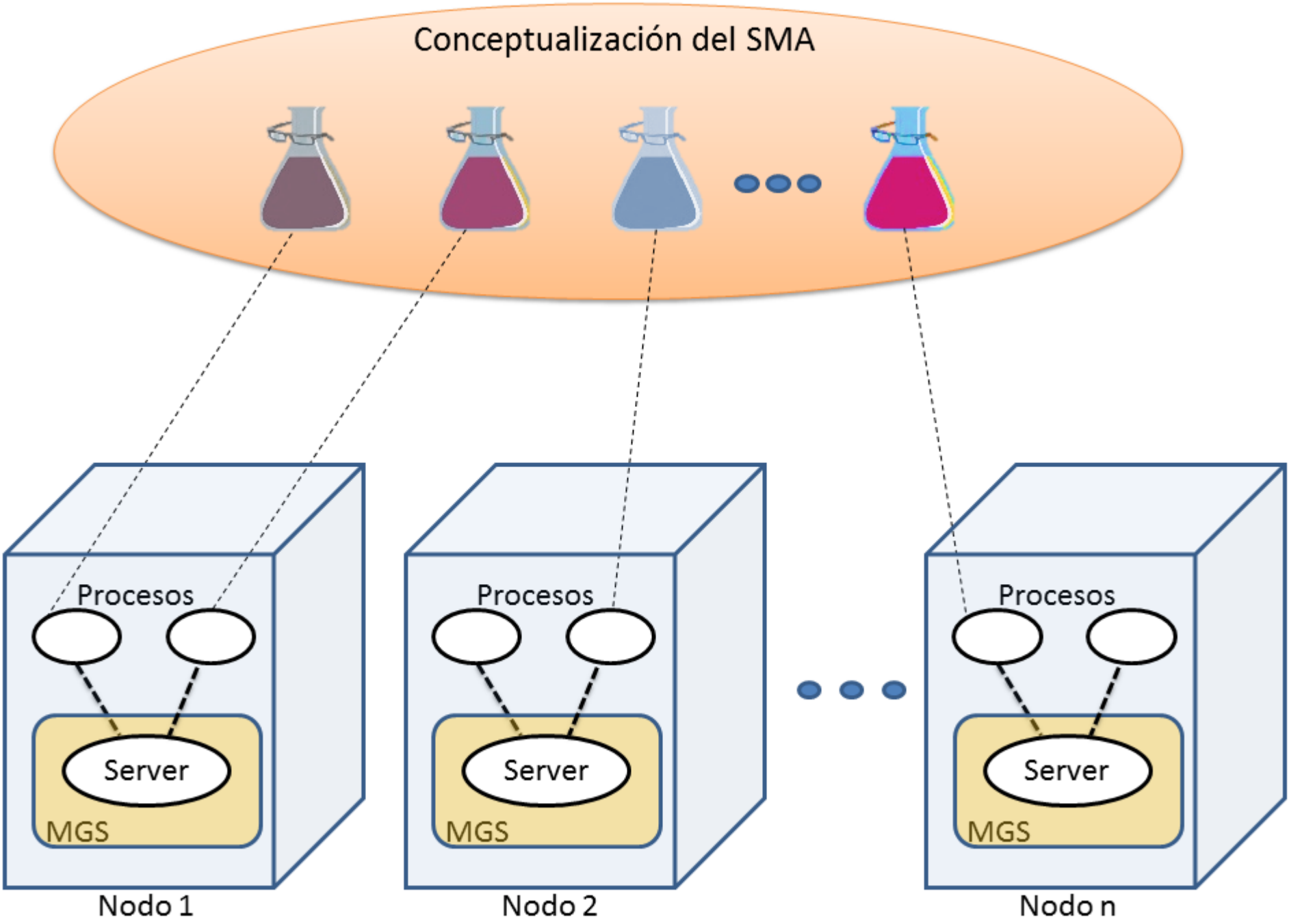


- Proveer mecanismos que permitan a una (comunidad de agentes:
- Conocer los servicios disponibles agrupados por niveles, por tipos de servicio, o alguna otra clasificación
- Conocer los agentes que prestan un determinado servicio y donde están
- Conocer las formas para acceder a esos servicios. entre otros.
- La movilidad,
- Comunicarse los agentes
- Su activación y desactivación, la posibilidad de invocarlos, etc.

Middleware para SMA






Conceptualización del SMA
Procesos
Server
MGS
Nodo 1
Procesos
Server
MGS
Nodo 2
Procesos
Server
MGS
Nodo n

Comparación Plataformas Despliegue

- Qué lenguaje de comunicación de agentes (ACL) soportan: FIPA ACL y/o KQML
- Si soportan movilidad de código,
- Arquitectura base de la plataforma,
- De qué tipo son los agentes soportados,
- Cuáles son los lenguajes en los que se pueden desarrollar los agentes,
- Cuáles son las licencias de los paquetes de software,
- Si están disponibles en Internet,
- Cuán dificultosa es la instalación de dichos paquetes,
- Qué tan completa es la documentación
- Qué tipo de interface posee.
- Si tienen un EDI

Herramientas para el desarrollo o IDE para agentes

IDE para SMA

- un IDE es un sistema informático compuesto por un conjunto de herramientas de programación.

- Un IDE puede dedicarse, en exclusiva, a un solo lenguaje de programación o bien puede utilizarse para varios.

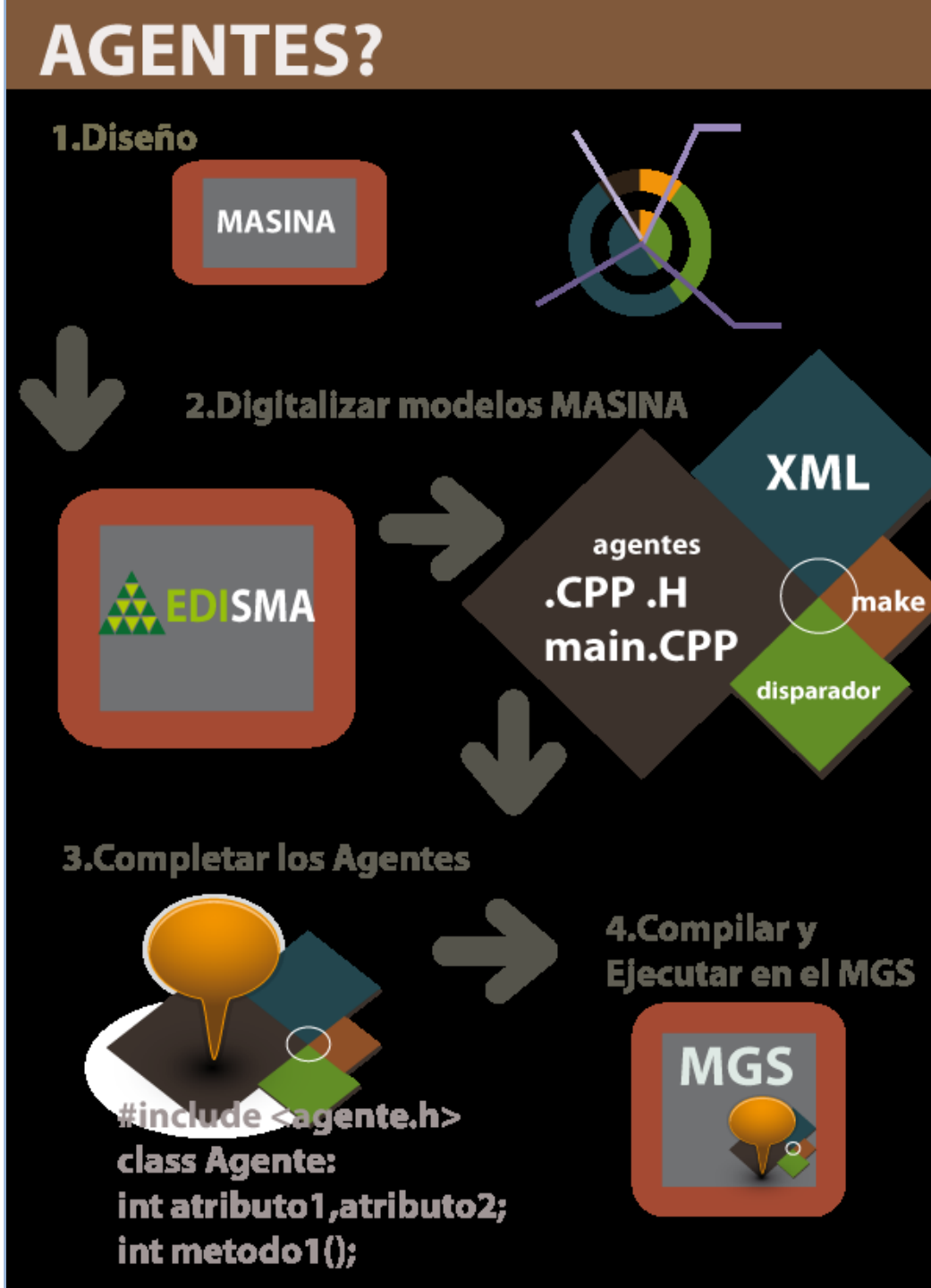
- Por lo general, un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de IGU.

EDISMA

Entorno de Desarrollo Integrado para la construcción de SMA.

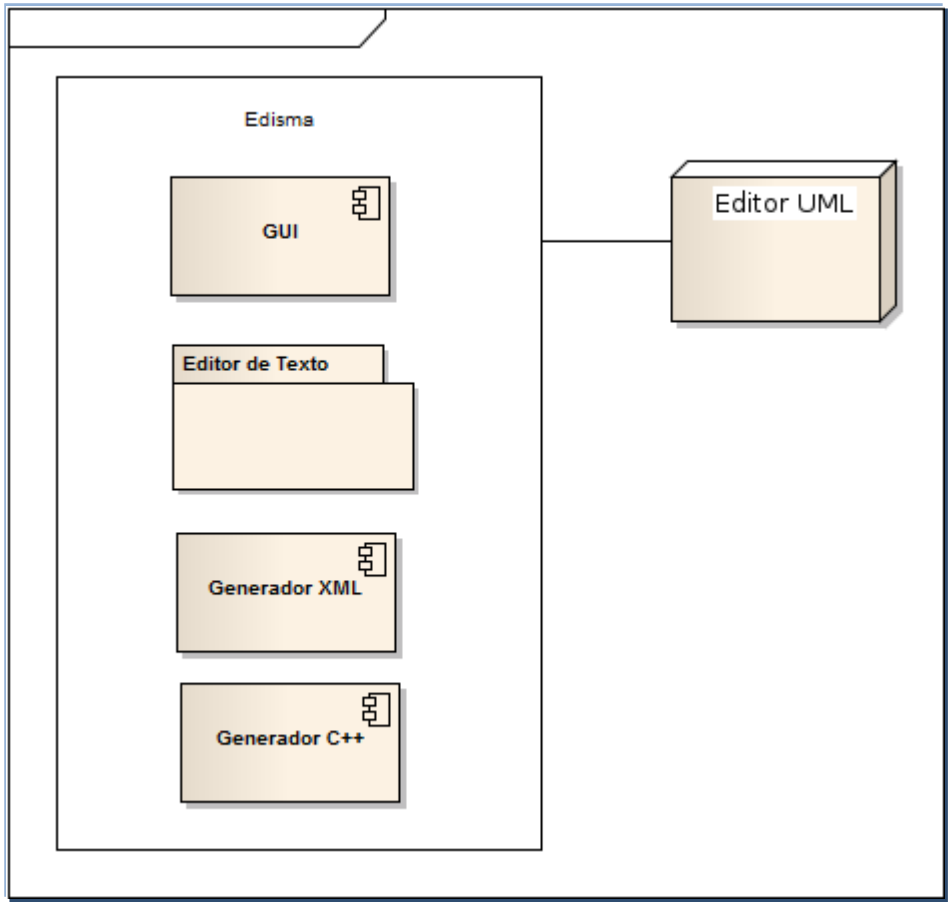
EDISMA pretende facilitar la creación de agentes modelados a través de MASINA.

Pero además, EDISMA despliega los agentes sobre MGS



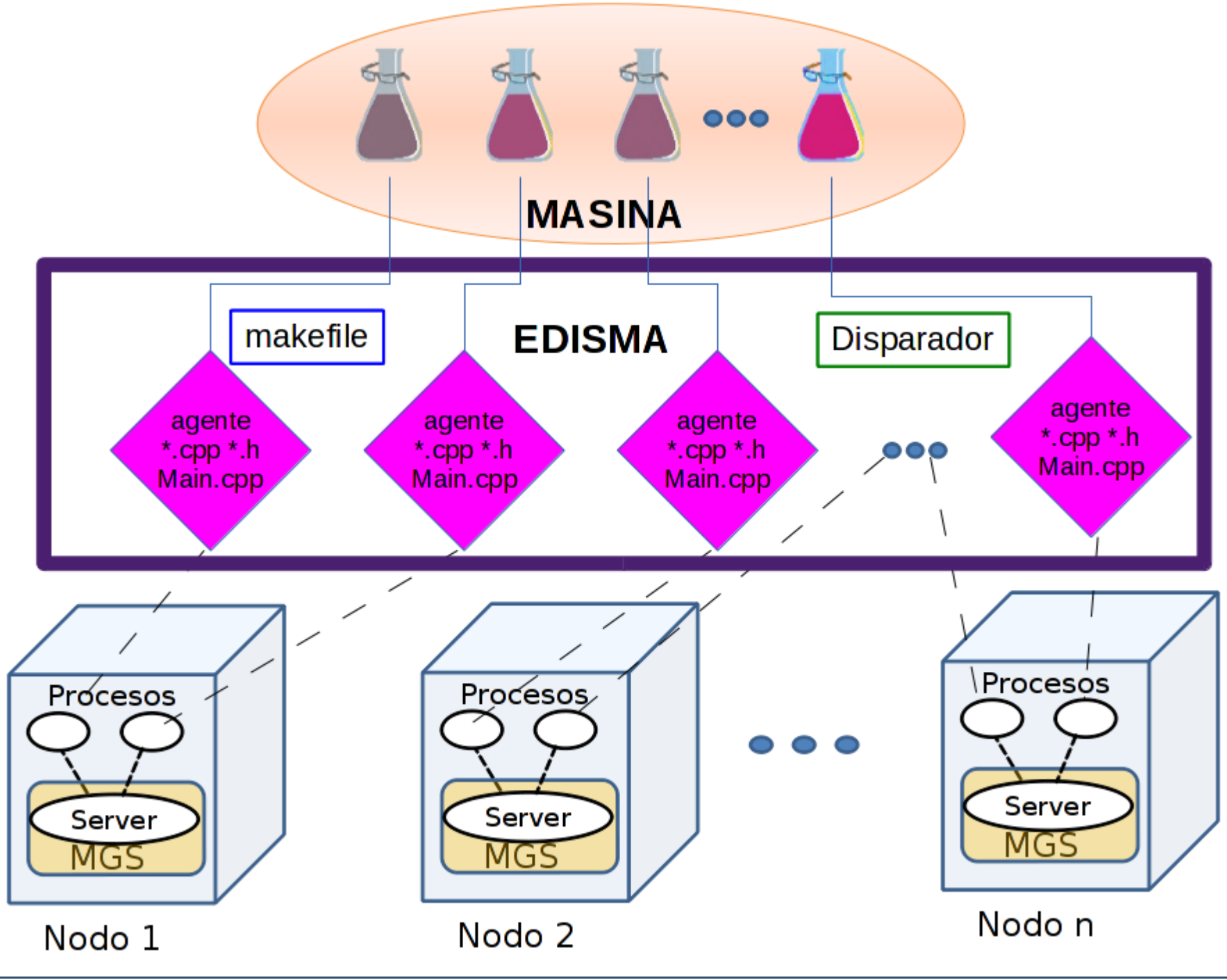
EDISMA.

- Proveer una IGU para modelar los agentes, seg´un MASINA.
- Implementar un componente para especificar los modelos. Almacena los modelos en un formato digital estándar, para ser usado tanto en la documentación del SMA, como en la generación de código fuente.
- Generar la estructura del SMA, integrando aspectos vinculados a la comunicación y coordinación entre los agentes,
- Proveer un mecanismo que permita modificar los códigos generados.

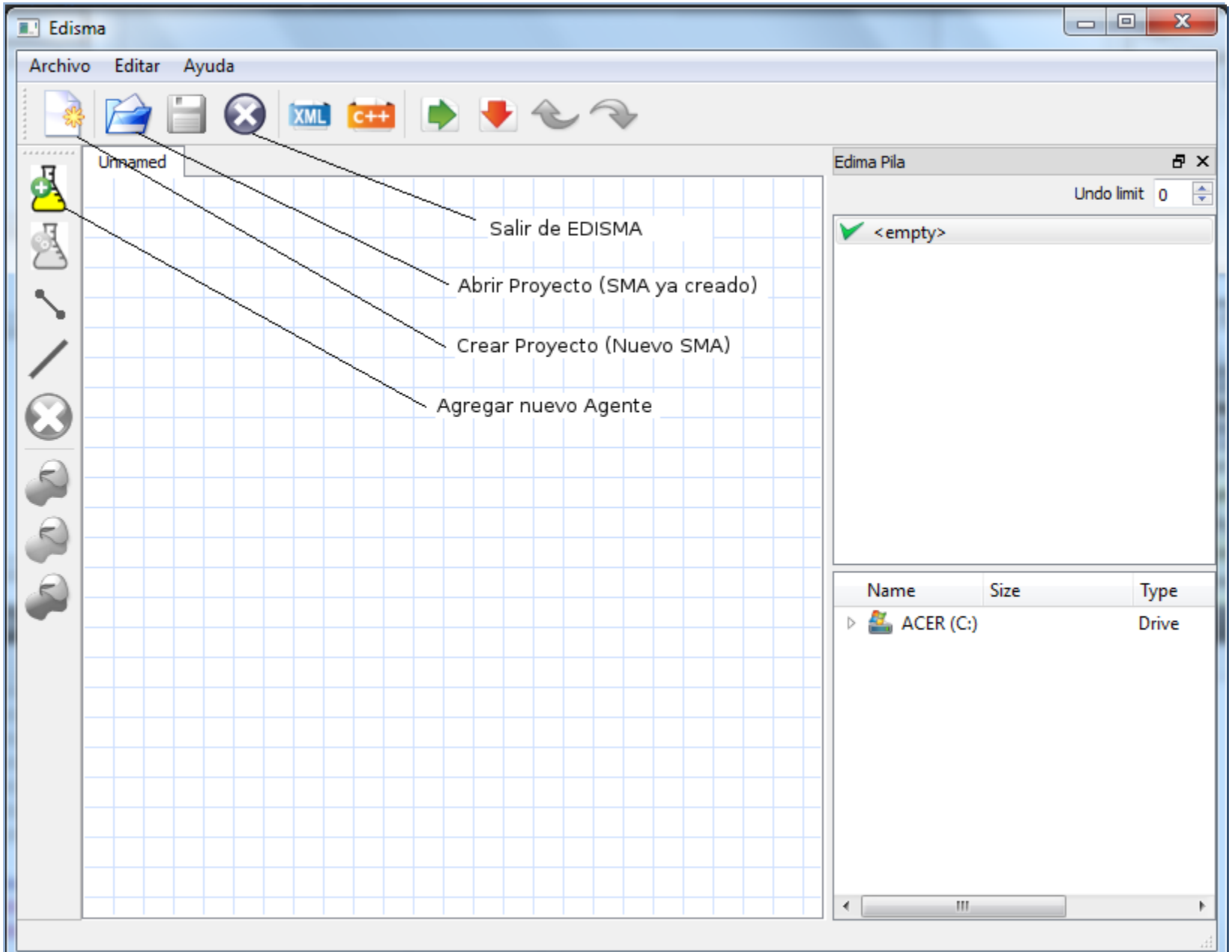


Servicios EDISMA

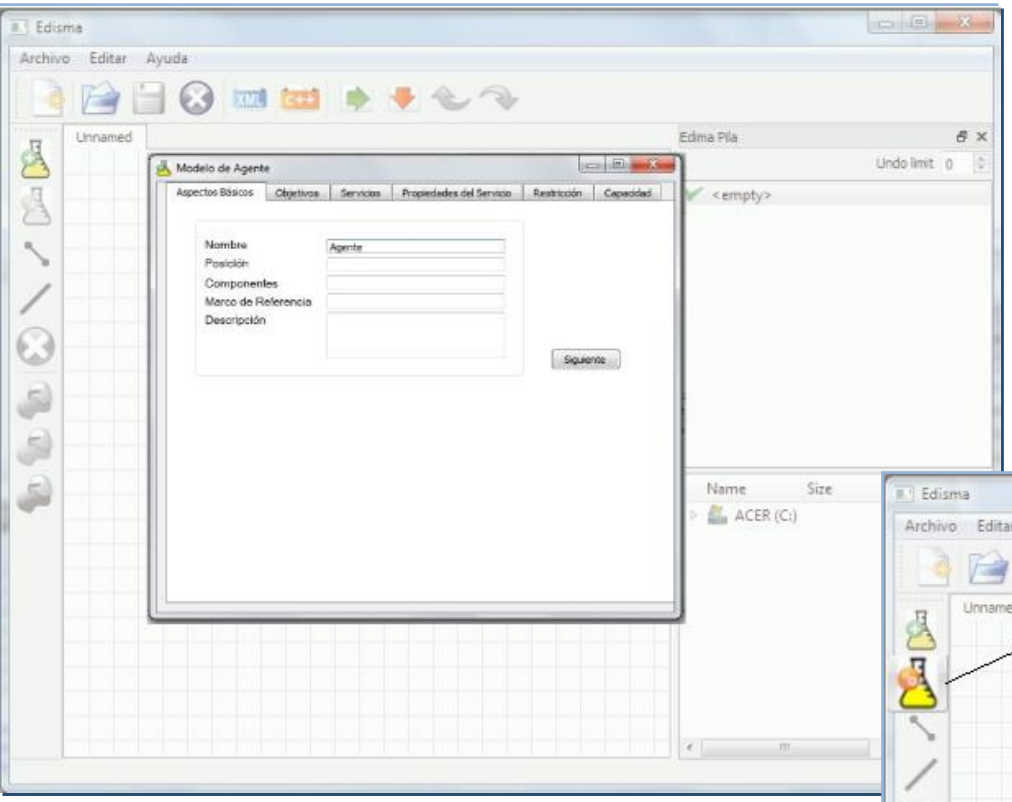
- Crear un SMA. En el contexto de la IGU, se asocia un SMA con un proyecto de software.
- Crear un agente.
- Crear un modelo de tarea a un agente.
- Crear un modelo de conversación entre agentes.
- Crear un modelo de comunicación entre agentes.
- Abrir un SMA elaborado creado previamente.
- Crear los archivos que permiten la implantación de los agentes en el MGS.
- Editar los archivos creados por EDISMA.


MASINA
EDISMA
makefile
Disparador
agente
*.cpp *.h
Main.cpp
agente
*.cpp *.h
Main.cpp
agente
*.cpp *.h
Main.cpp
agente
*.cpp *.h
Main.cpp
Procesos
Server
MGS
Procesos
Server
MGS
Procesos
Server
MGS
Nodo 1
Nodo 2
Nodo n

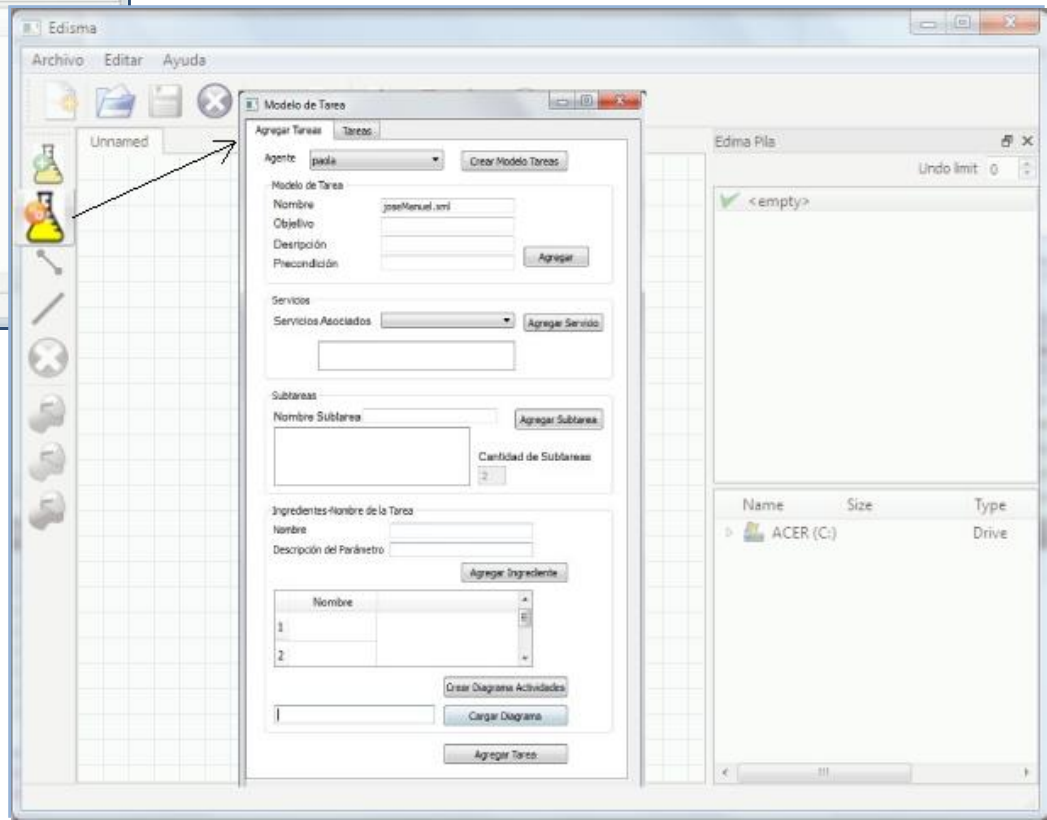
EDISMA


Edisma
Archivo
Editar
Ayuda
XML
C++
Unnamed
Salir de EDISMA
Abrir Proyecto (SMA ya creado)
Crear Proyecto (Nuevo SMA)
Agregar nuevo Agente
Edima Pila
Undo limit
0
<empty>
Name
Size
Type
ACER (C:)
Drive

EDISMA



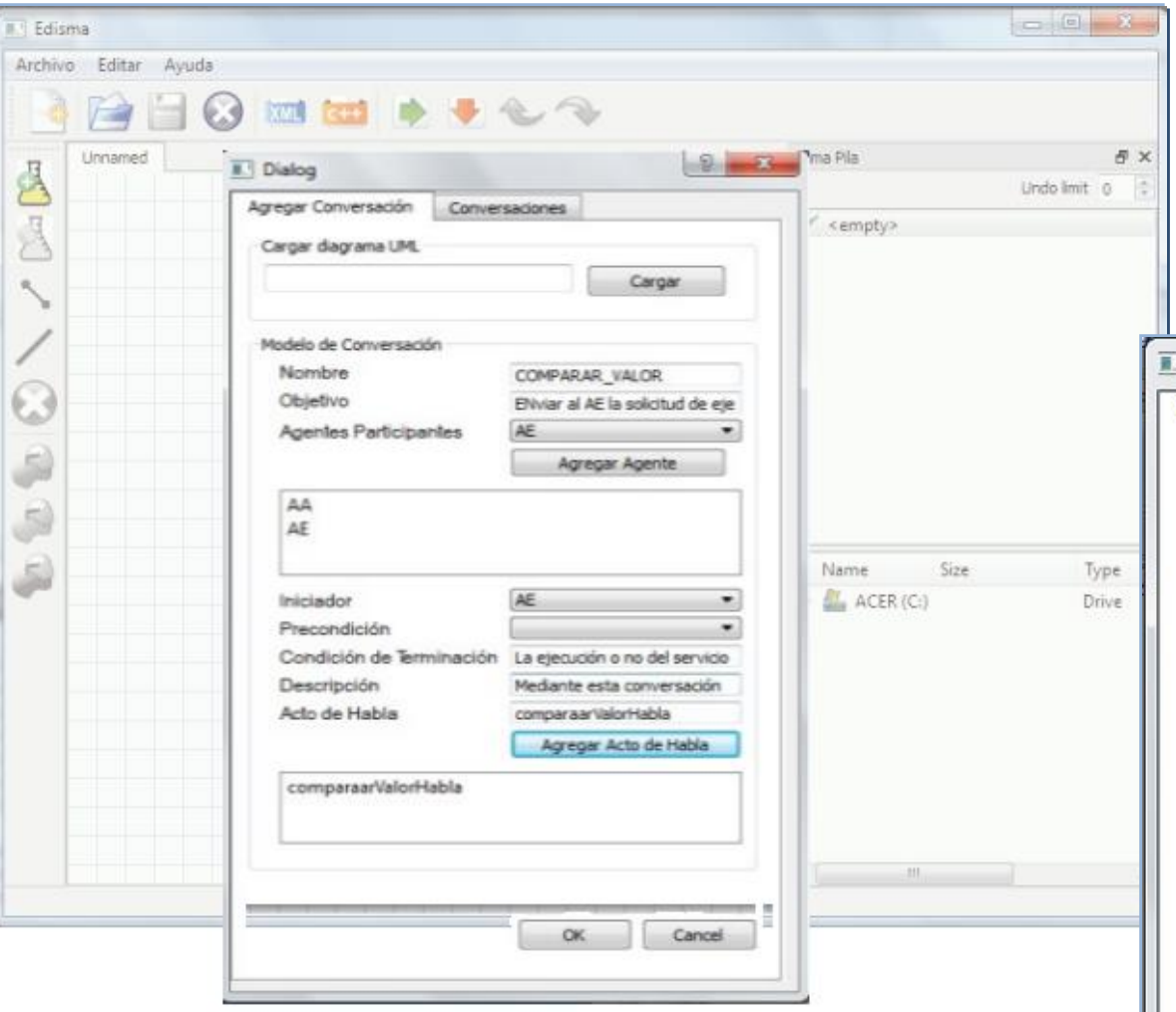
Modelo de agentes

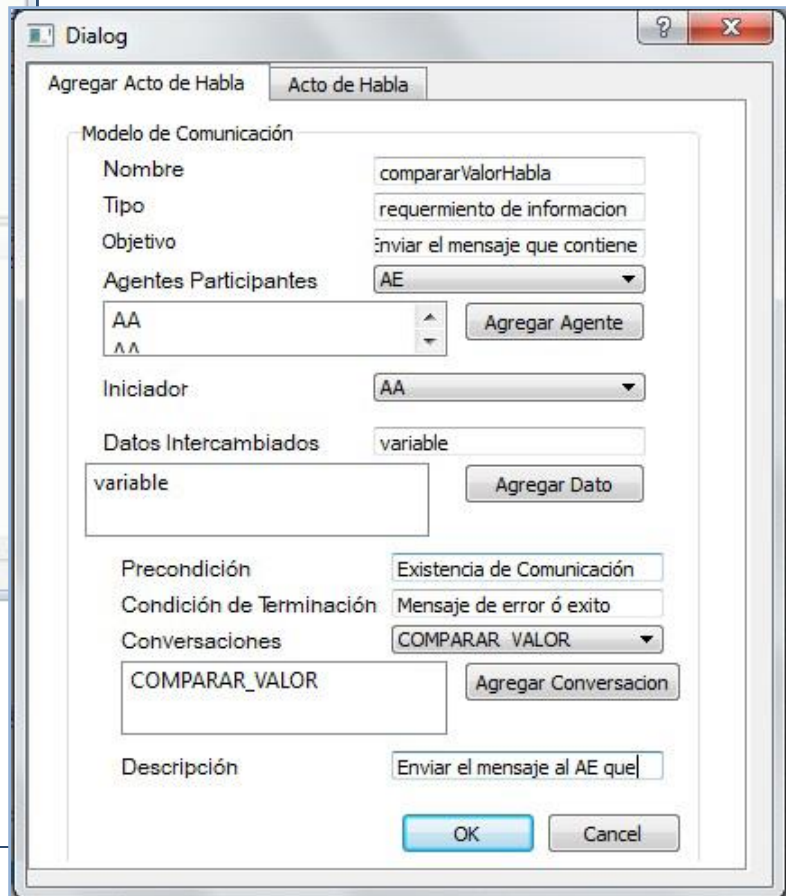


Modelo de tareas

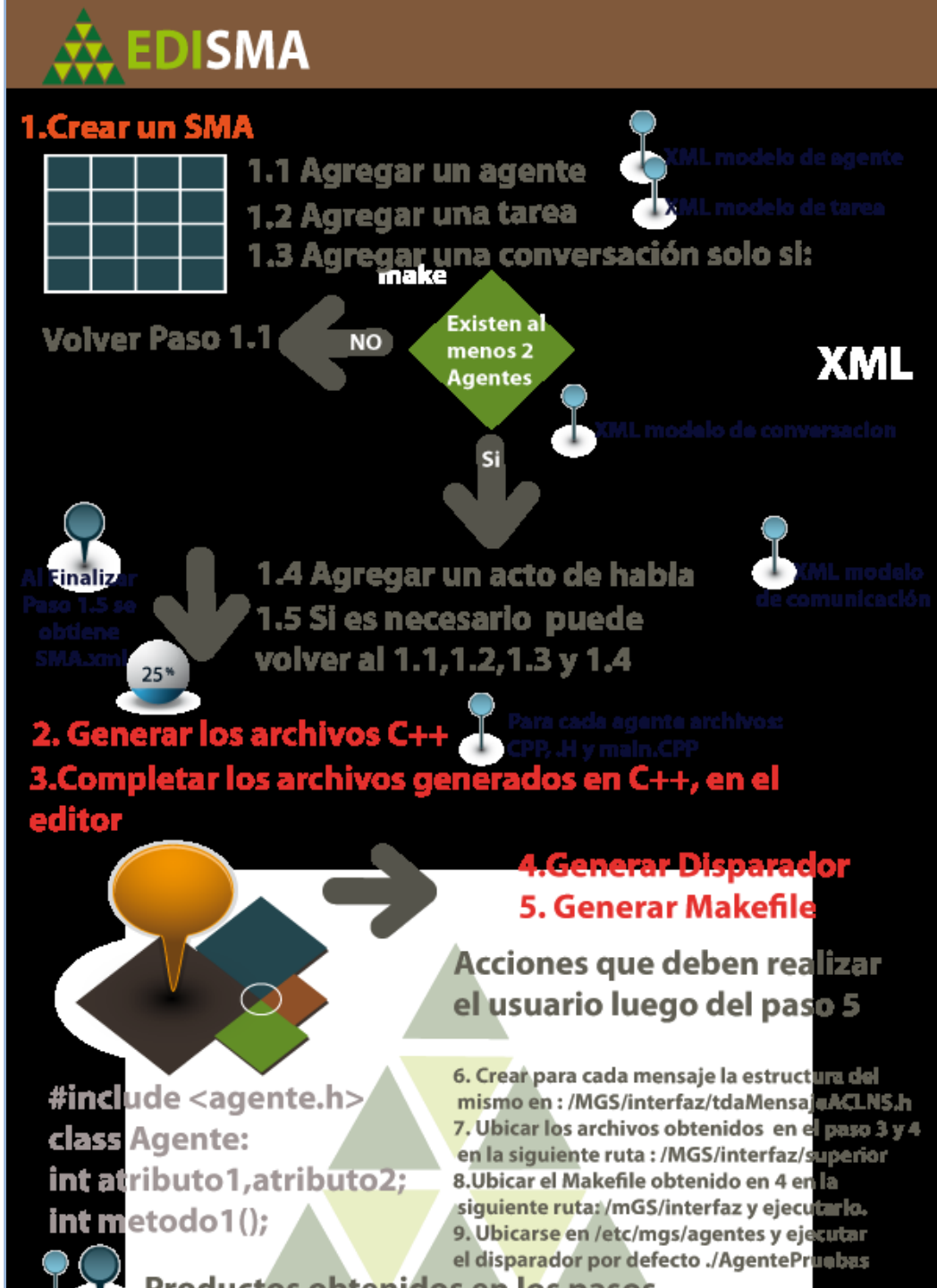
EDISMA

Modelo de conversación





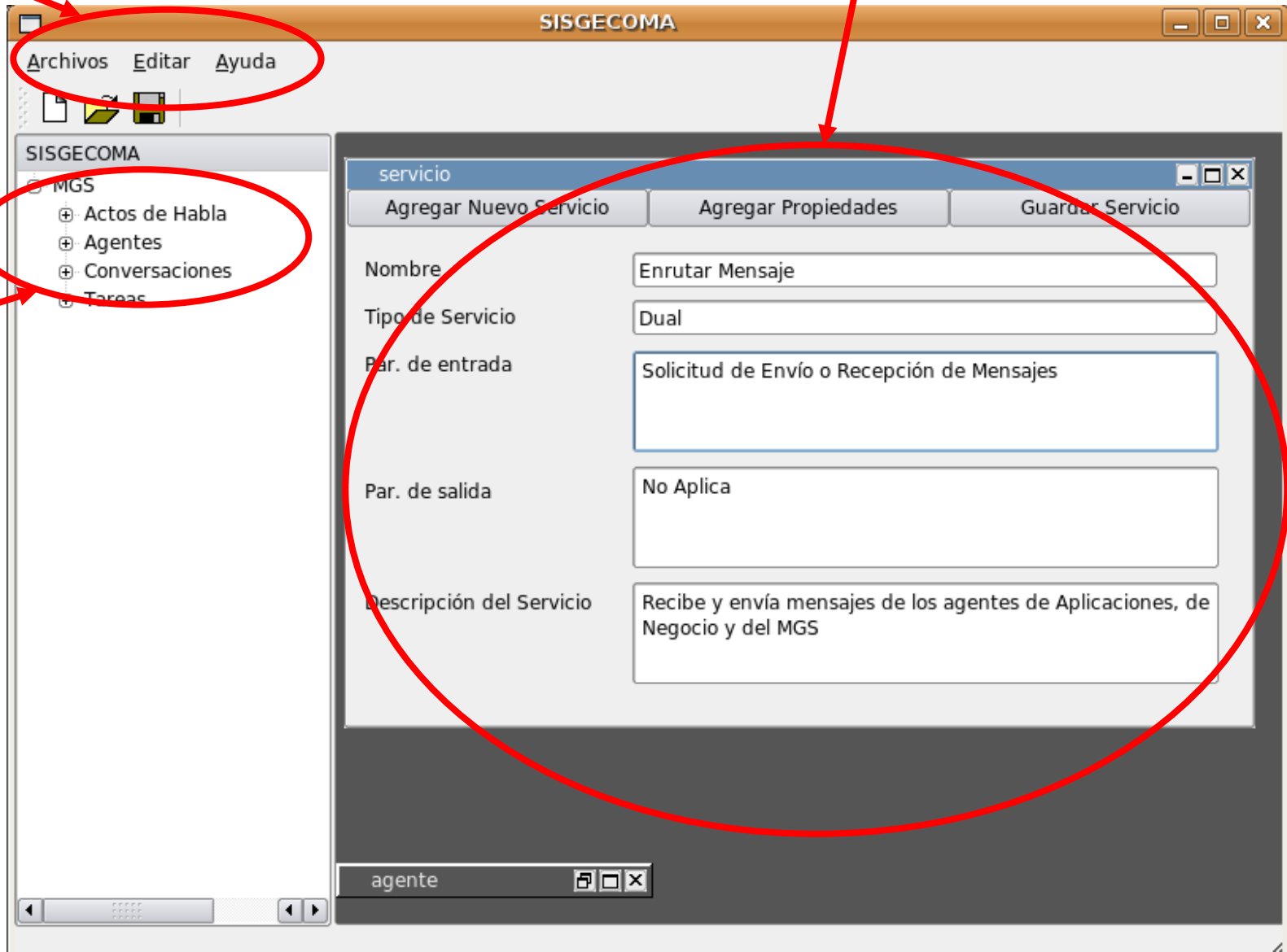
EDISMA


EDISMA
1.Crear un SMA
1.1 Agregar un agente
1.2 Agregar una tarea
1.3 Agregar una conversación solo si:
XML modelo de agente
XML modelo de tarea
make
Volver Paso 1.1
NO
Existen al menos 2 Agentes
XML
XML modelo de conversacion
Si
Al Finalizar Paso 1.5 se obtiene SMA.xml
25%
1.4 Agregar un acto de habla
1.5 Si es necesario puede volver al 1.1,1.2,1.3 y 1.4
XML modelo de comunicación
2. Generar los archivos C++
Para cada agente archivos: CPP, .H y main.CPP
3.Completar los archivos generados en C++, en el editor
4.Generar Disparador
5. Generar Makefile
Acciones que deben realizar el usuario luego del paso 5
#include <agente.h>
class Agente:
int atributo1,atributo2;
int metodo1();
6. Crear para cada mensaje la estructura del mismo en : /MGS/interfaz/tdaMensajeACLNS.h
7. Ubicar los archivos obtenidos en el paso 3 y 4 en la siguiente ruta : /MGS/interfaz/superior
8.Ubicar el Makefile obtenido en 4 en la siguiente ruta: /mGS/interfaz y ejecutarlo.
9. Ubicarse en /etc/mgs/agentes y ejecutar el disparador por defecto ./AgentePruebas

Sistema Generador de Código para la Metodología MASINA

Menú

Área de Trabajo



Árbol de Búsqueda

SISGECOMA

cemisid
centro de estudios en microelectrónica y sistemas distribuidos

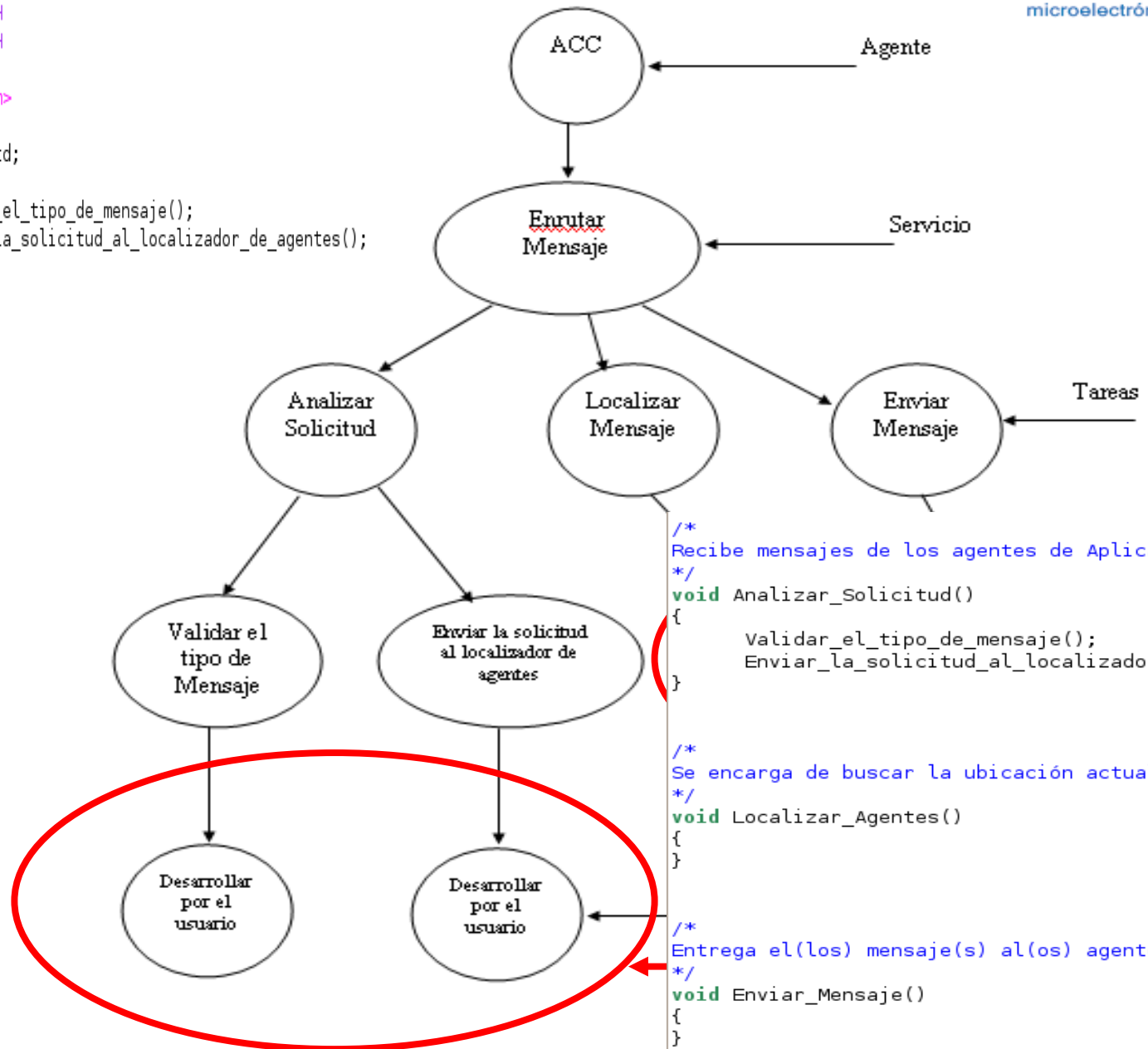
```
#ifndef SUBTAREA_H
#define SUBTAREA_H

#include <iostream>

using namespace std;

     void Validar_el_tipo_de_mensaje();
     void Enviar_la_solicitud_al_localizador_de_agentes();

#endif
```




```
/*
Recibe mensajes de los agentes de Aplicaciones, de Negoci
*/
void Analizar_Solicitud()
{
      Validar_el_tipo_de_mensaje();
      Enviar_la_solicitud_al_localizador_de_agentes();
}


/*
Se encarga de buscar la ubicación actual del(os) agente(s
*/
void Localizar_Agentes()
{
}


/*
Entrega el(los) mensaje(s) al(os) agente(s)
*/
void Enviar_Mensaje()
{
}
```

esclavo

SISGECOMA

cemisid
centro de estudios en microelectrónica y sistemas distribuidos

Servicios	Tareas
obedecer	T1 Reporte T2 Caminante

Servicios	Tareas
ordenar	T1 Manda

TAREA: Caminante	
Nombre	Caminante
Objetivo	Caminar
Descripción	Camina
Servicios asociados	obedecer
Precondición	
Sub-tareas	for(int i=1; i<11; i++) cout<<"estoy dando el paso"<<i<<endl;

jefe

```
void manda()
{
      cout<<"ponte a caminar";
}


void reporte()
{
      cout<<"hola, ya llegue"<<endl;
}


void caminante()
{
      for(int i =1; i<11;i++)
        cout<<"estoy dando el paso "<<i<<endl;
}
```