



UNIVERSIDAD  
DE LOS ANDES  
MÉRIDA VENEZUELA



**UTPL**  
UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

# 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

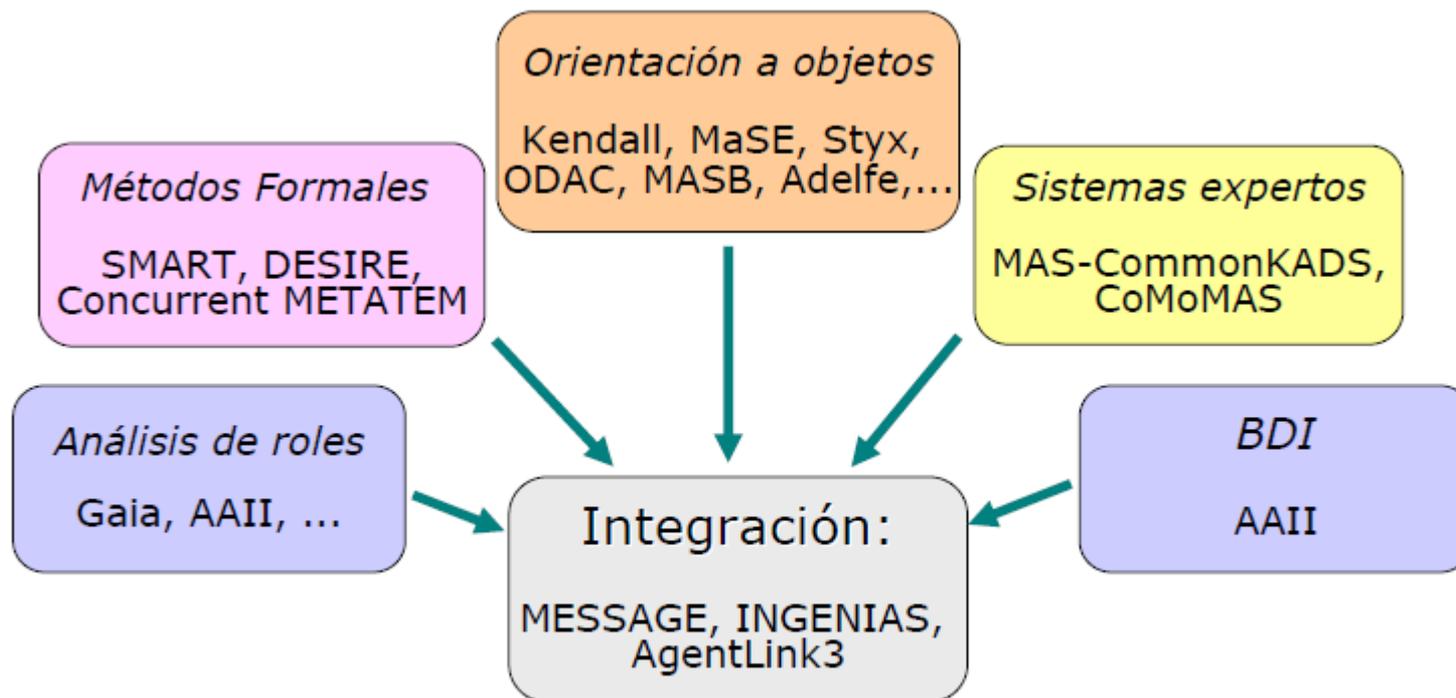
- **Técnicas de ingeniería del software** : estructurar el proceso de desarrollo
- **Técnicas de inteligencia artificial** : dotar a los programas de capacidad para tratar situaciones imprevistas y tomar decisiones
- **Programación concurrente y distribuida** : tratar la coordinación de tareas ejecutadas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



# Metodología 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*

# Metodologías de desarrollo de SMA

## MASINA

### 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).

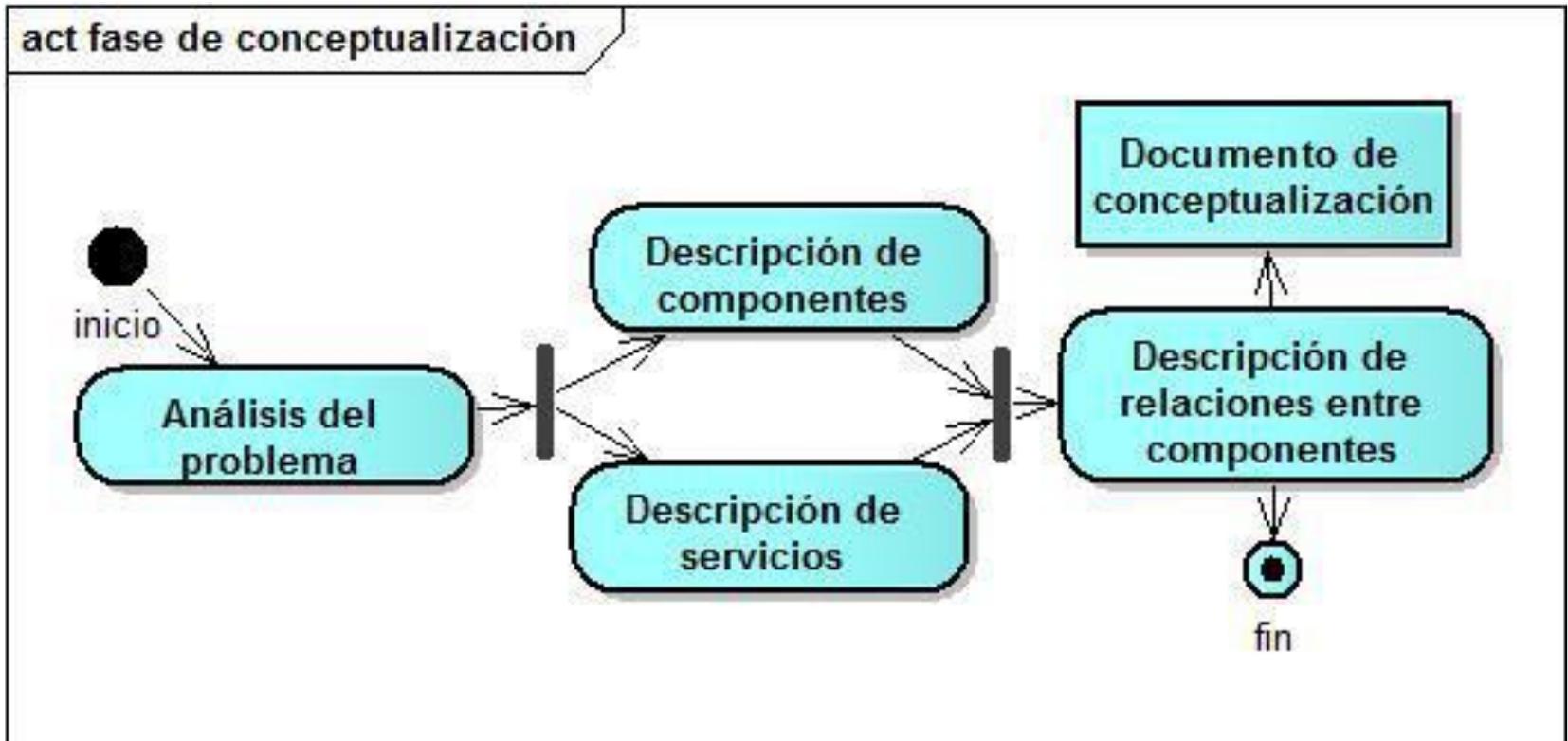
#### *Codificación y prueba*

#### *Integración*

#### *Operación y mantenimiento*

# MASINA

## Fase de Conceptualización



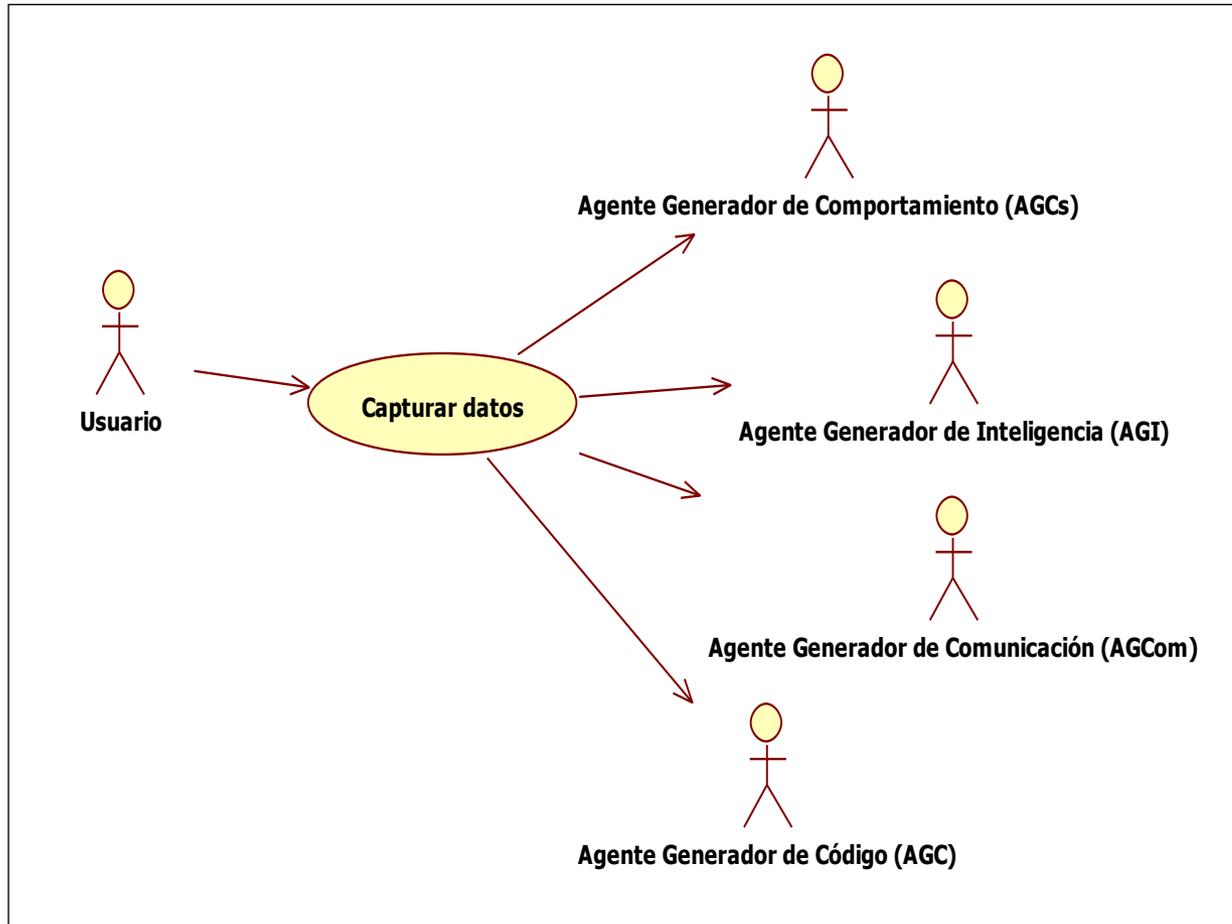
# MASINA

- 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**

# MASINA

## Fase de Conceptualización

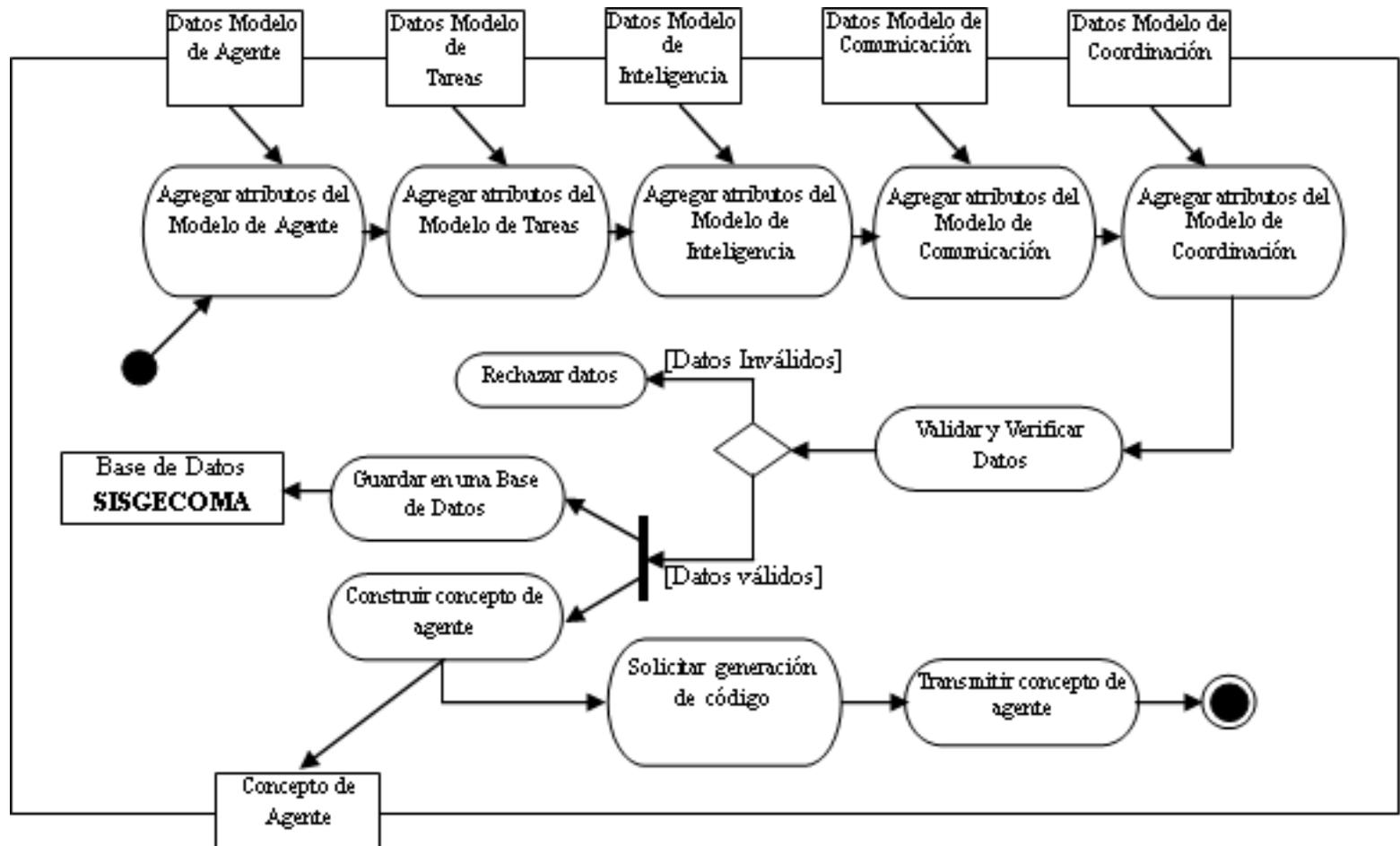
Casos de  
uso



# MASINA

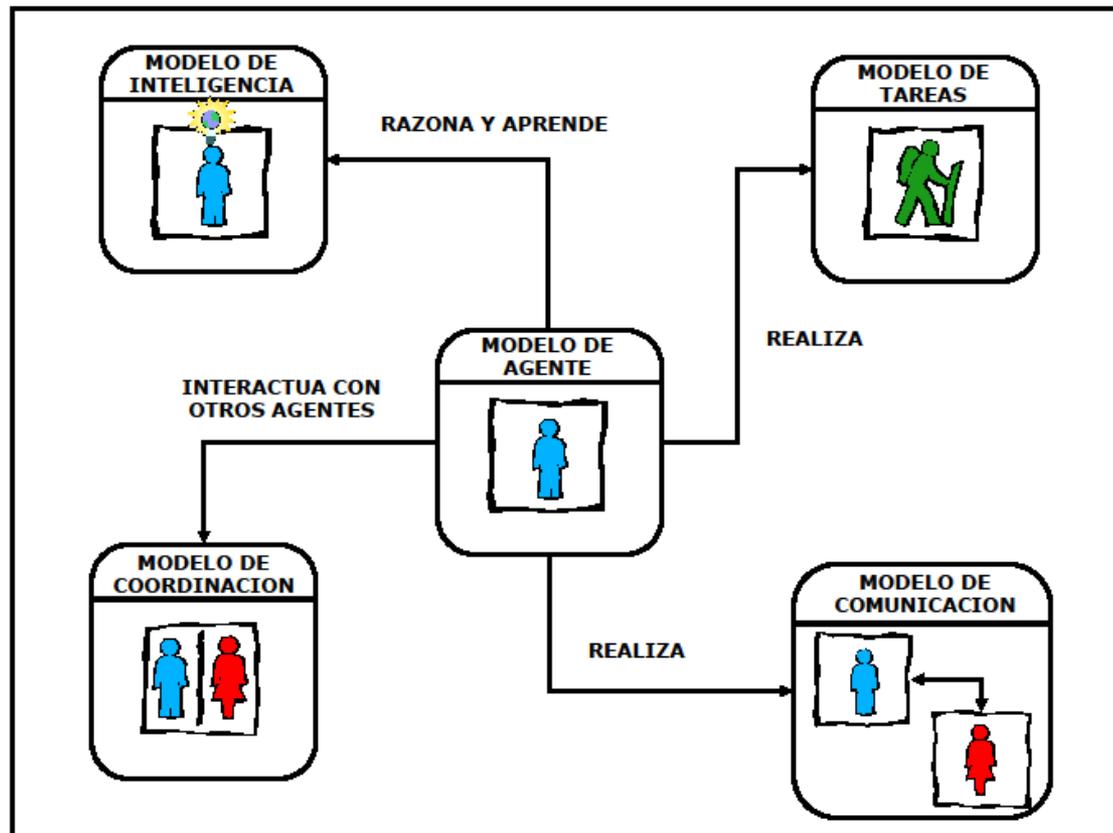
## Fase de Conceptualización

Diagrama de actividades



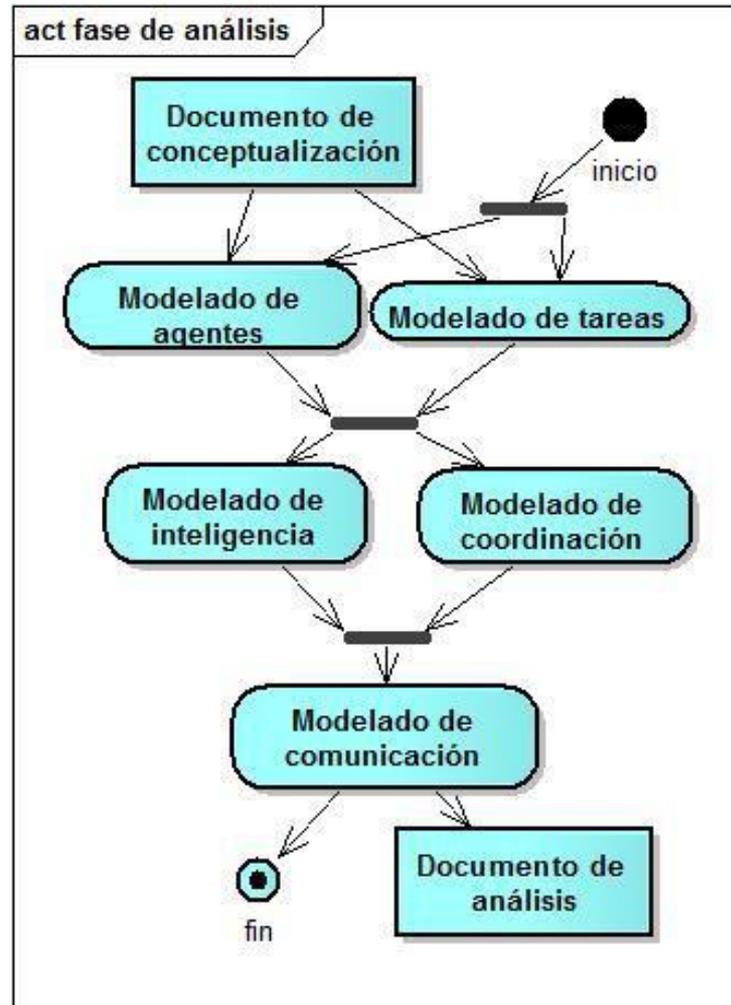
# MASINA

## Fase de Análisis



# MASINA

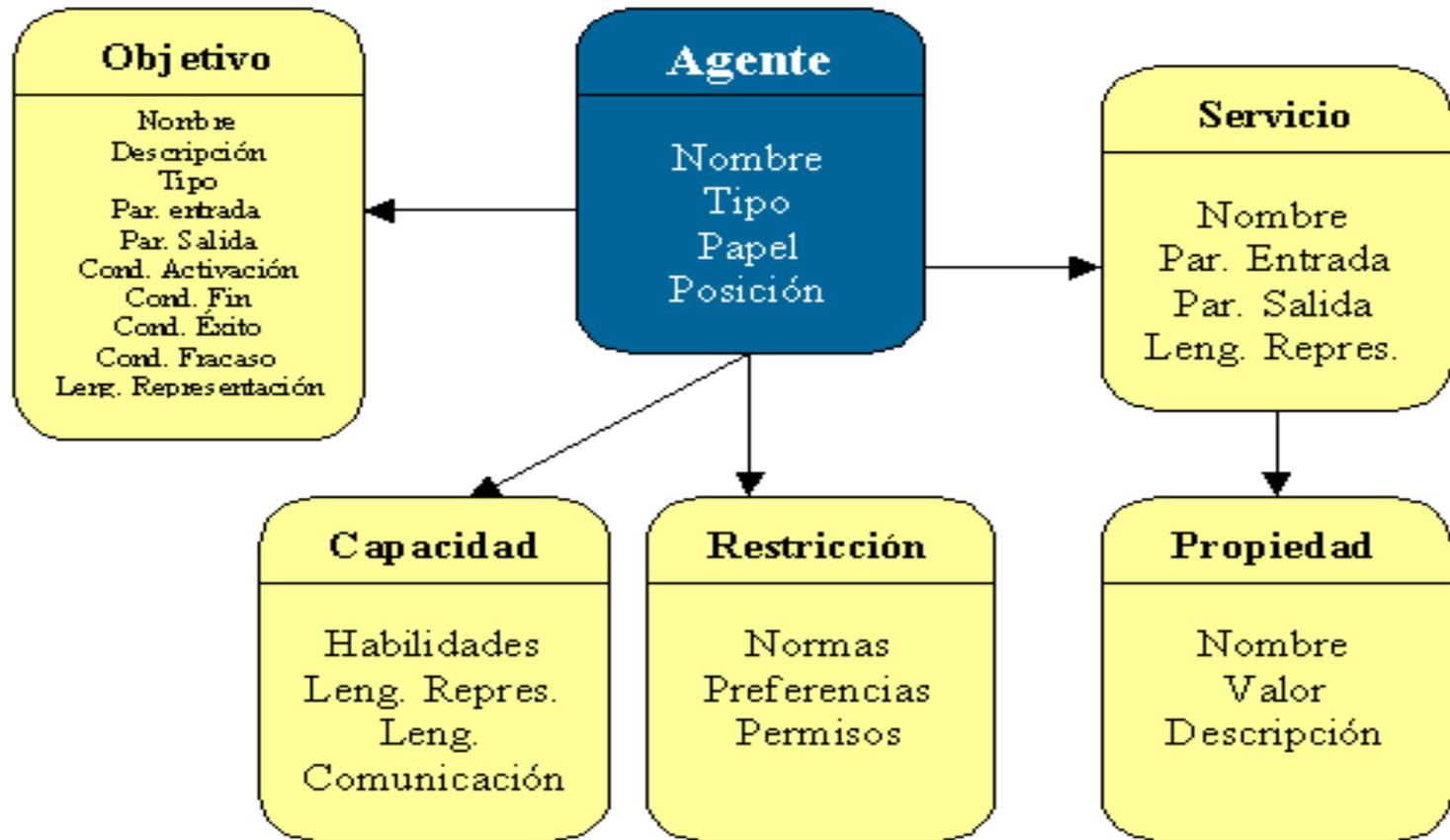
## Fase de Análisis



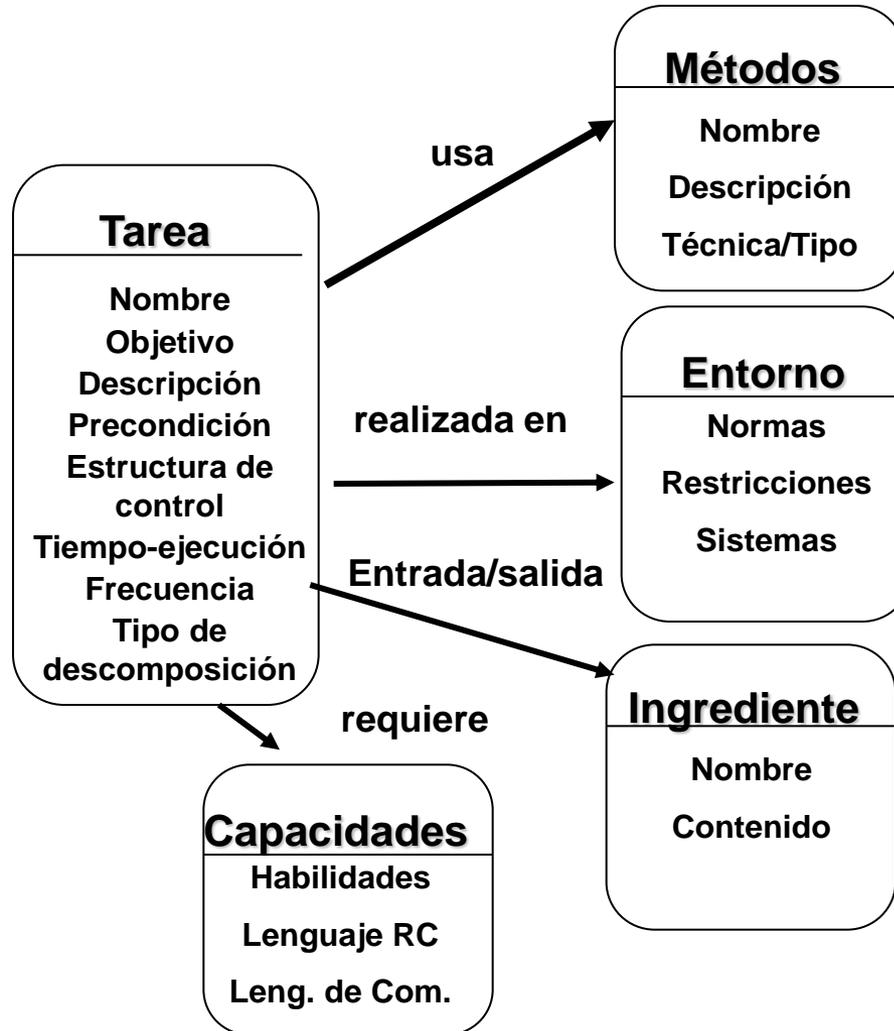
# MASINA

- 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.**

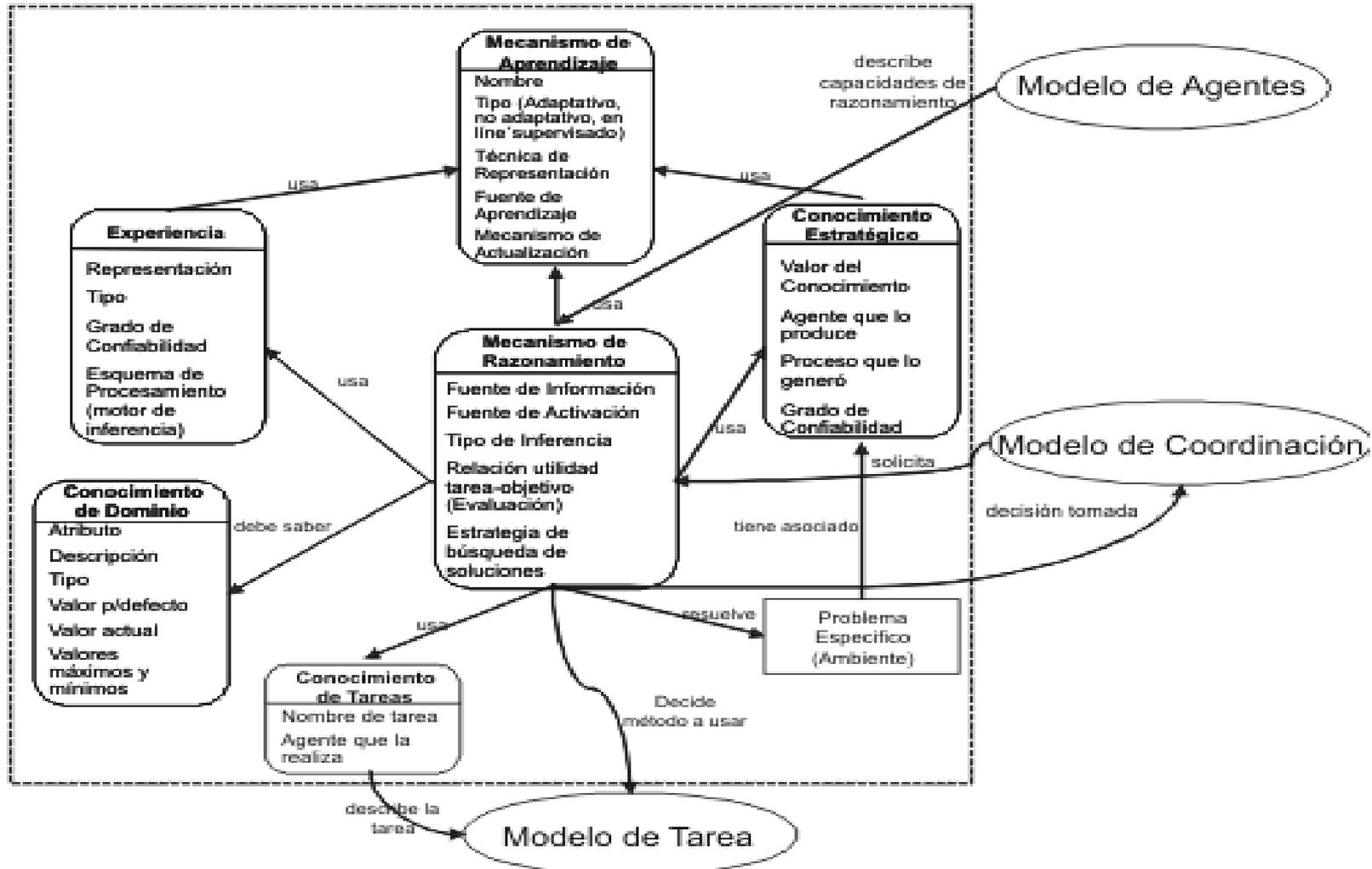
# Modelo de Agente



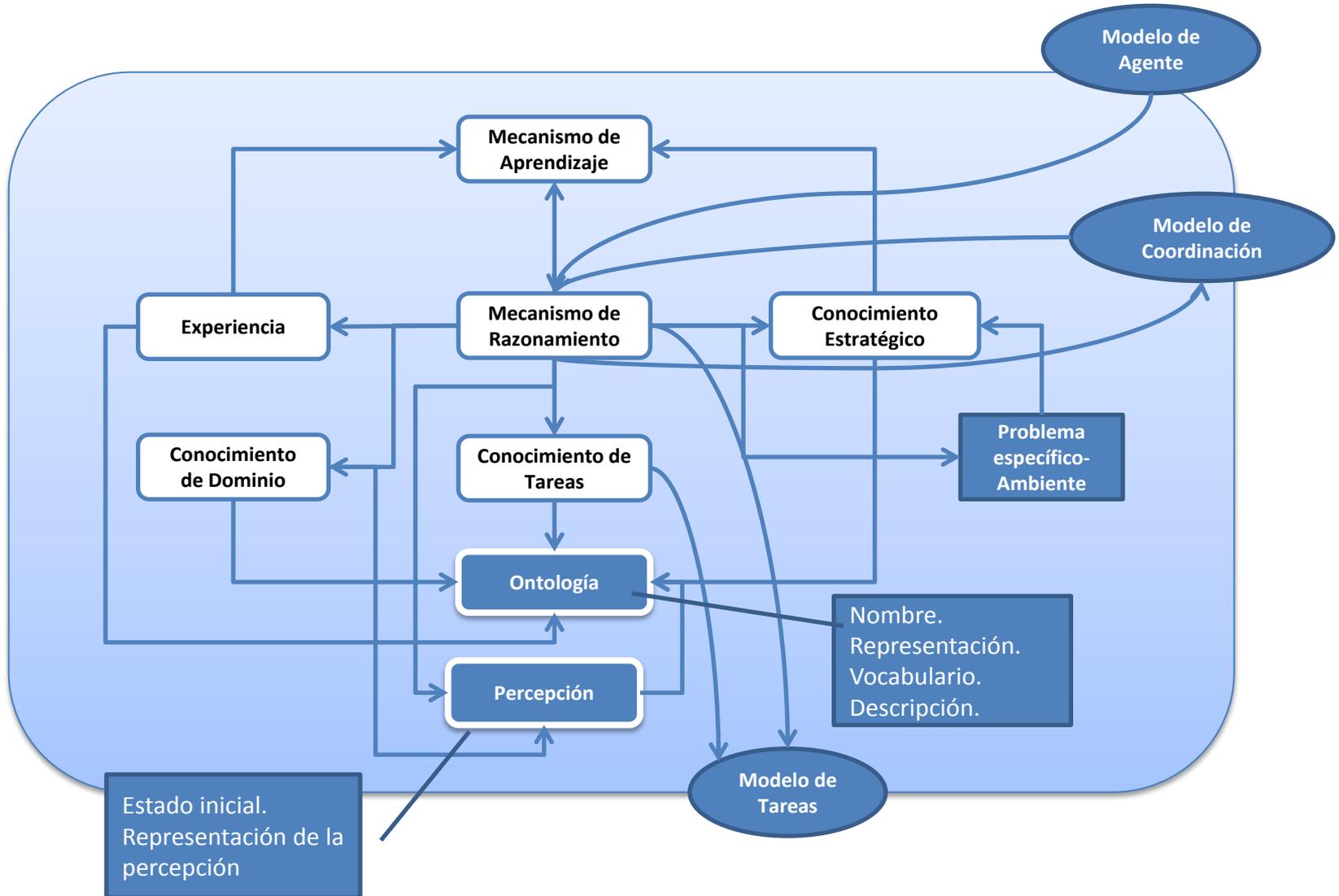
# Modelo de Tareas



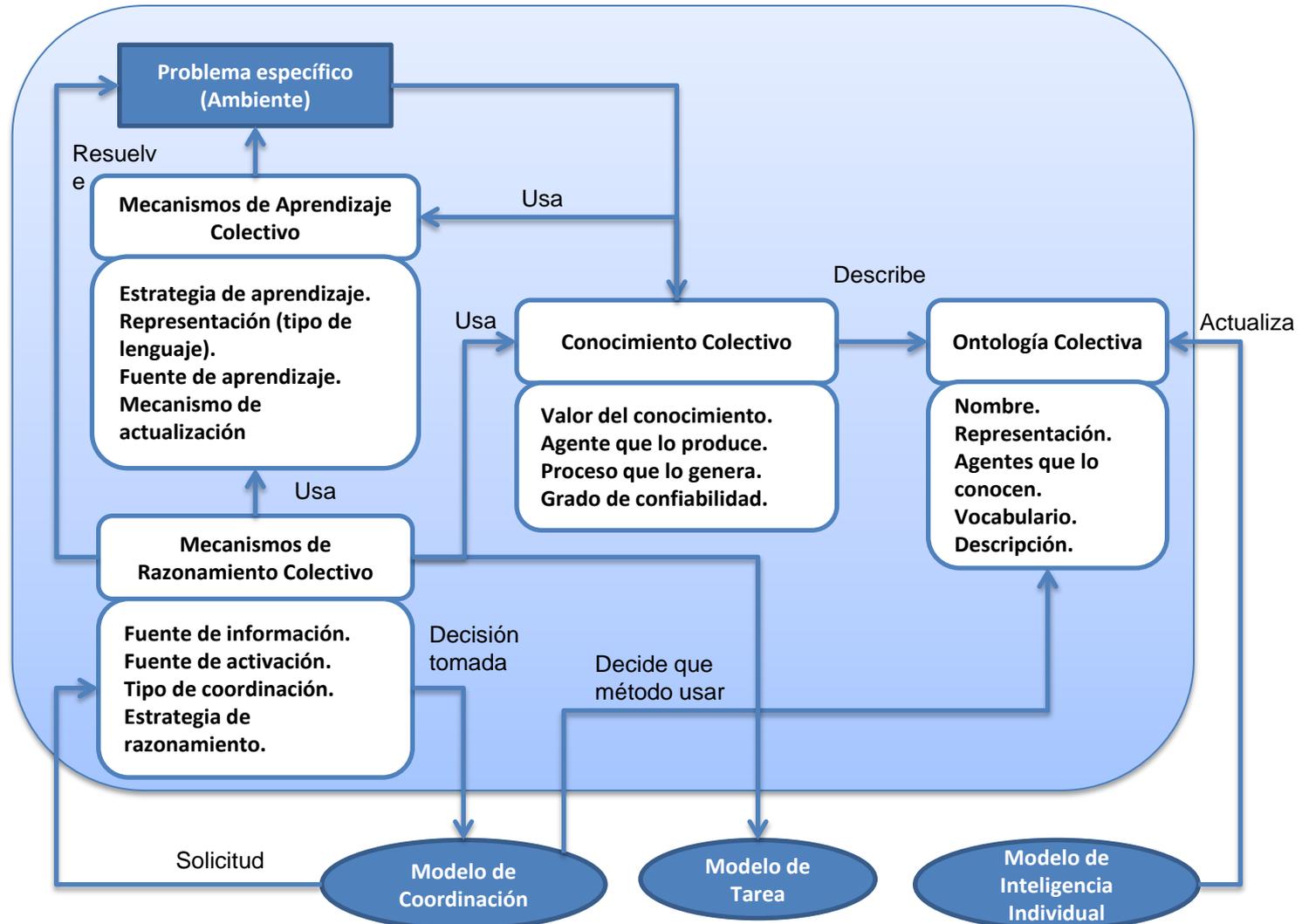
# Modelo de Inteligencia



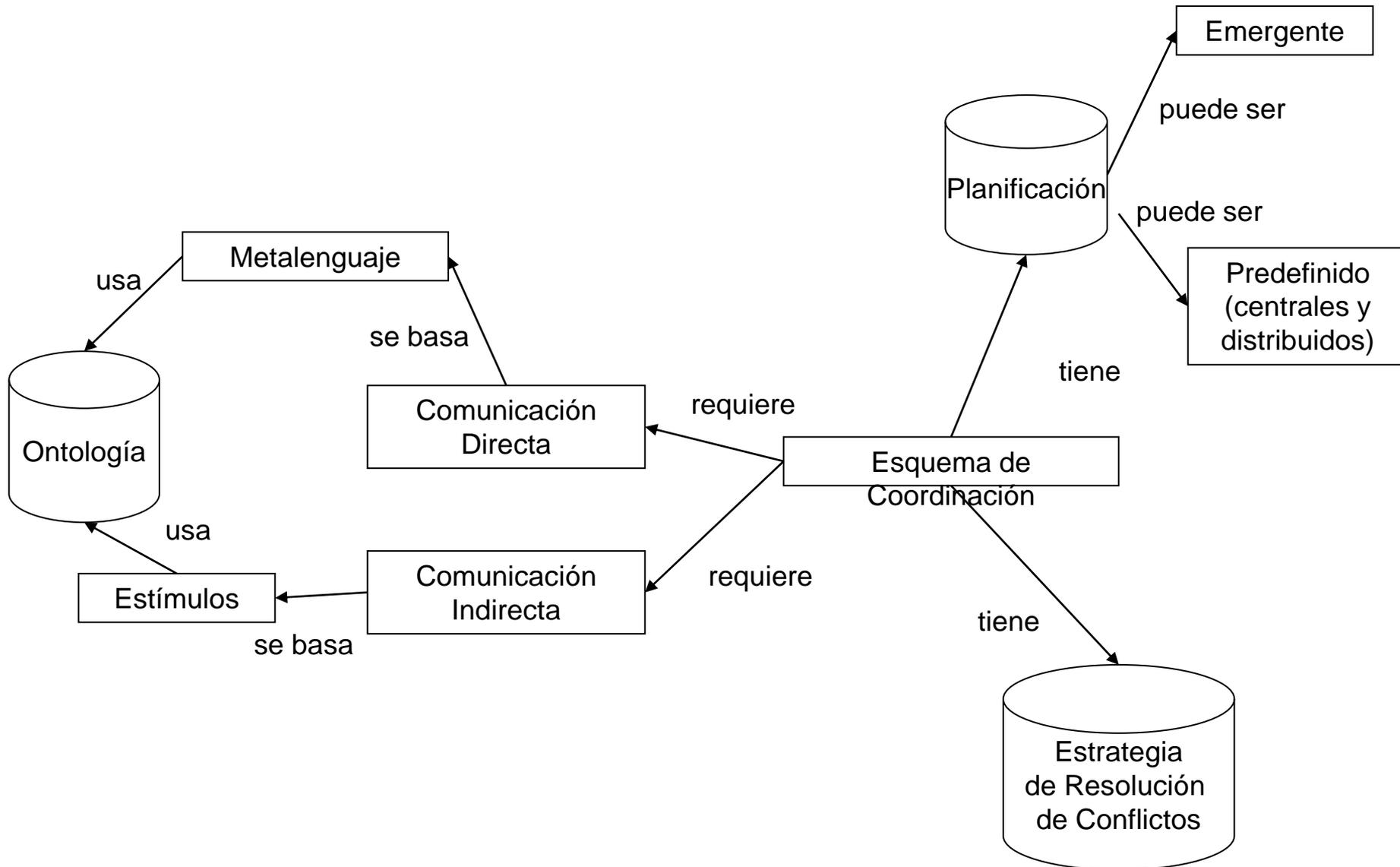
# Modelo de Inteligencia individual de MASINA



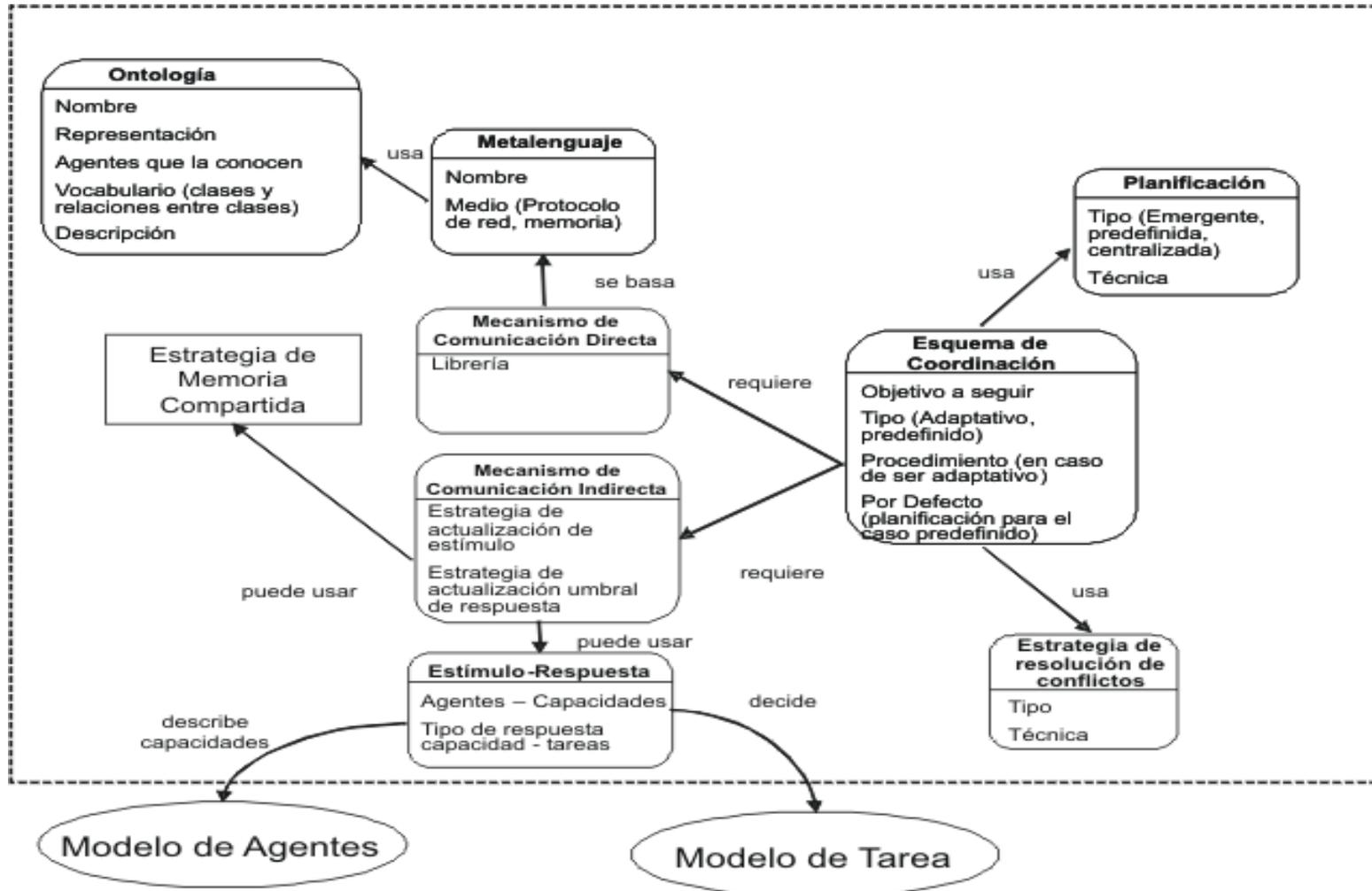
# Modelo de Inteligencia Colectiva de MASINA



# Problema de Coordinación



# Modelo de Coordinación



# Modelo de Coordinación y Comunicación

## Conversación

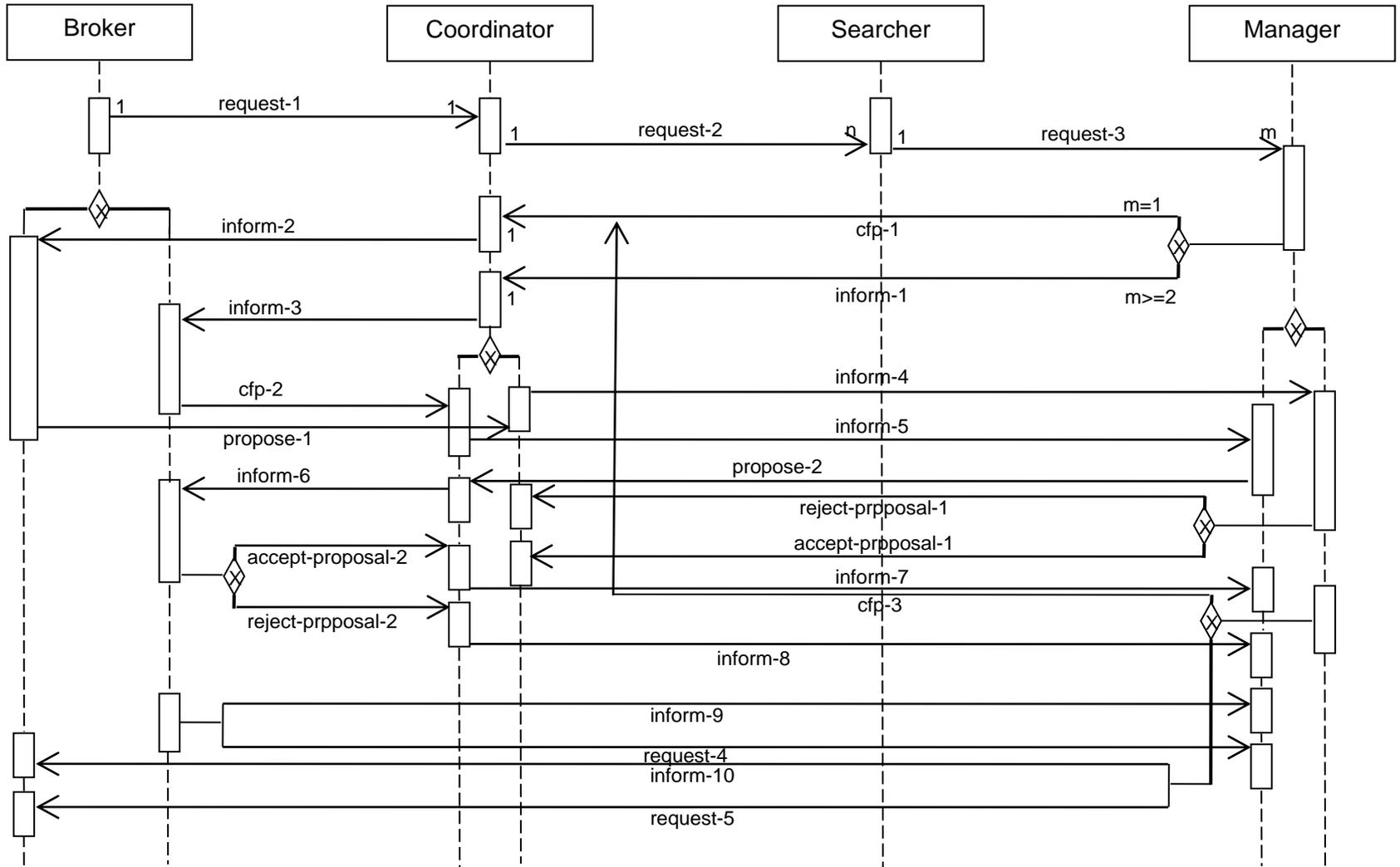
Nombre  
Tipo  
Objetivo  
Agentes  
Iniciador  
Servicio  
Actos de Habla  
Descripción  
Precondición  
Condición de Terminación

## Acto de Habla

Objetivo  
Tipo  
Agentes Participantes  
Comunicación  
Emisor  
Receptor  
Conversación  
Servicio  
Datos Intercambiados  
Descripción  
Precondición  
Condición de Terminación  
Performativa  
Medio de Comunicación

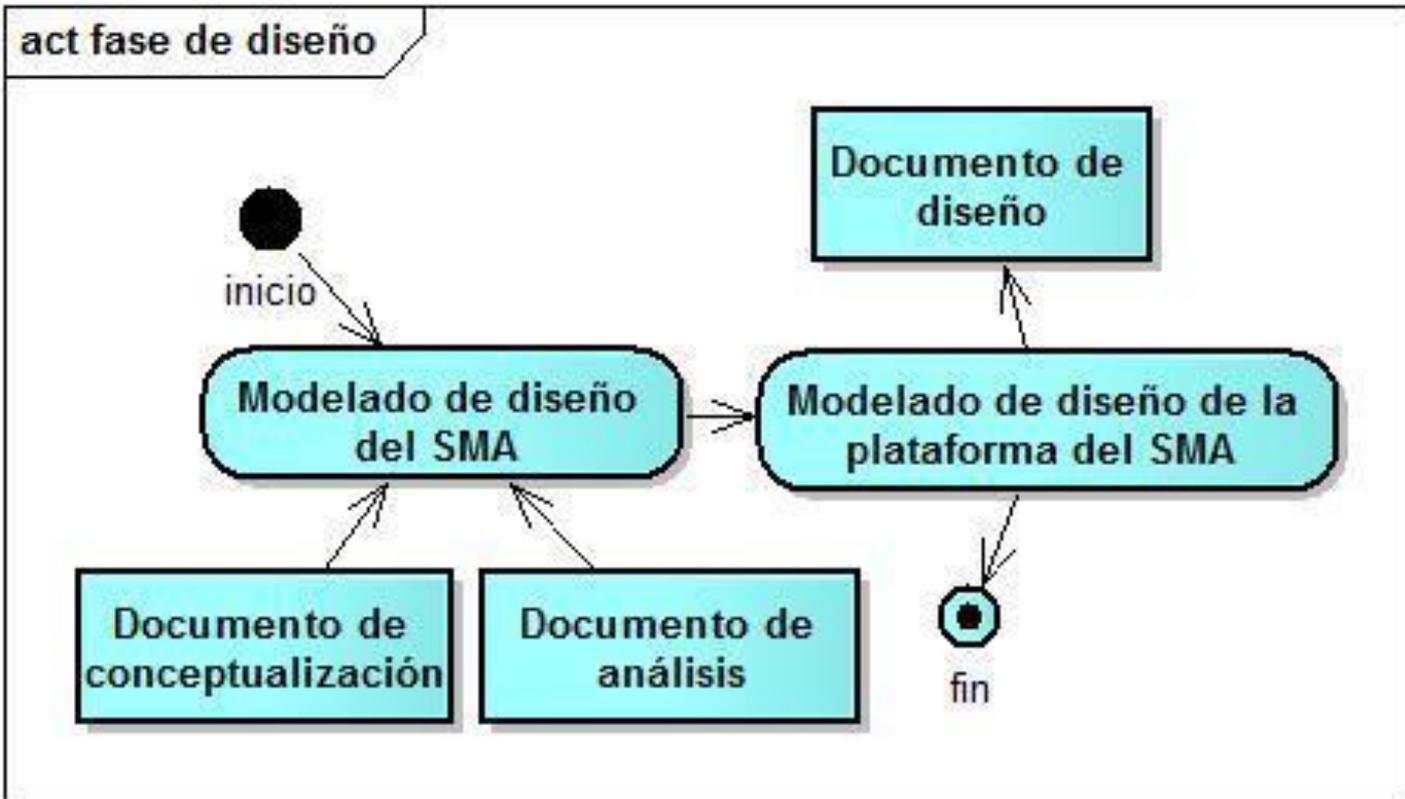
# Diagrama Interacción: Protocolo de interacción FIPA

Metascheduler Protocol



# MASINA

## Fase de diseño



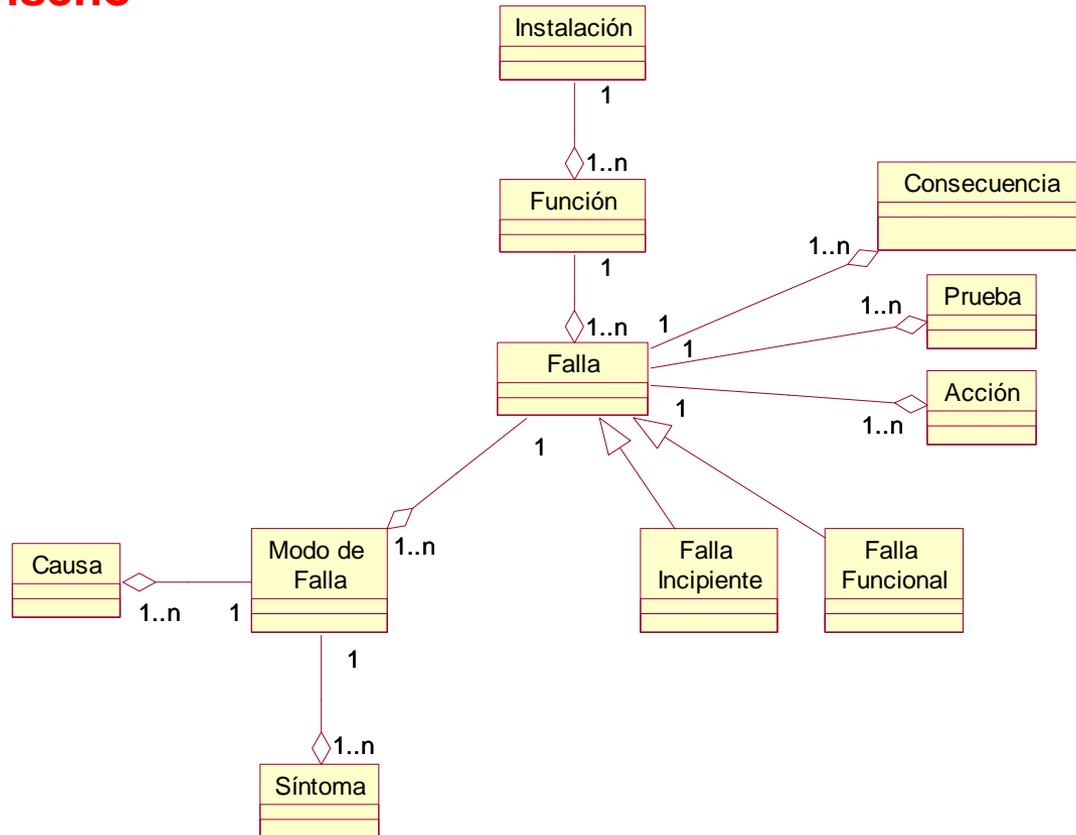
# MASINA

- 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.

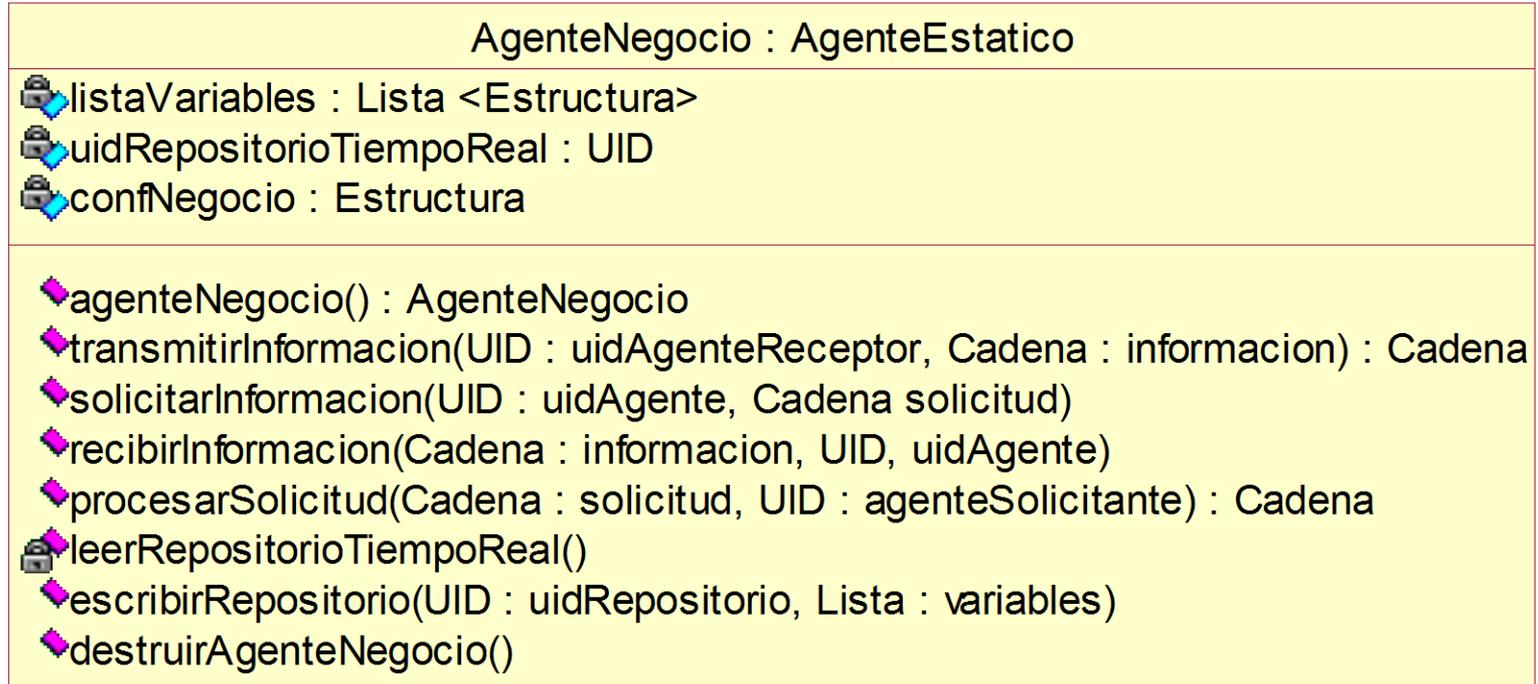
Nosotros particularmente usamos TDSO

# MASINA

## Modelo de Diseño



# Diagrama de clases del Agente de Negocio



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

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

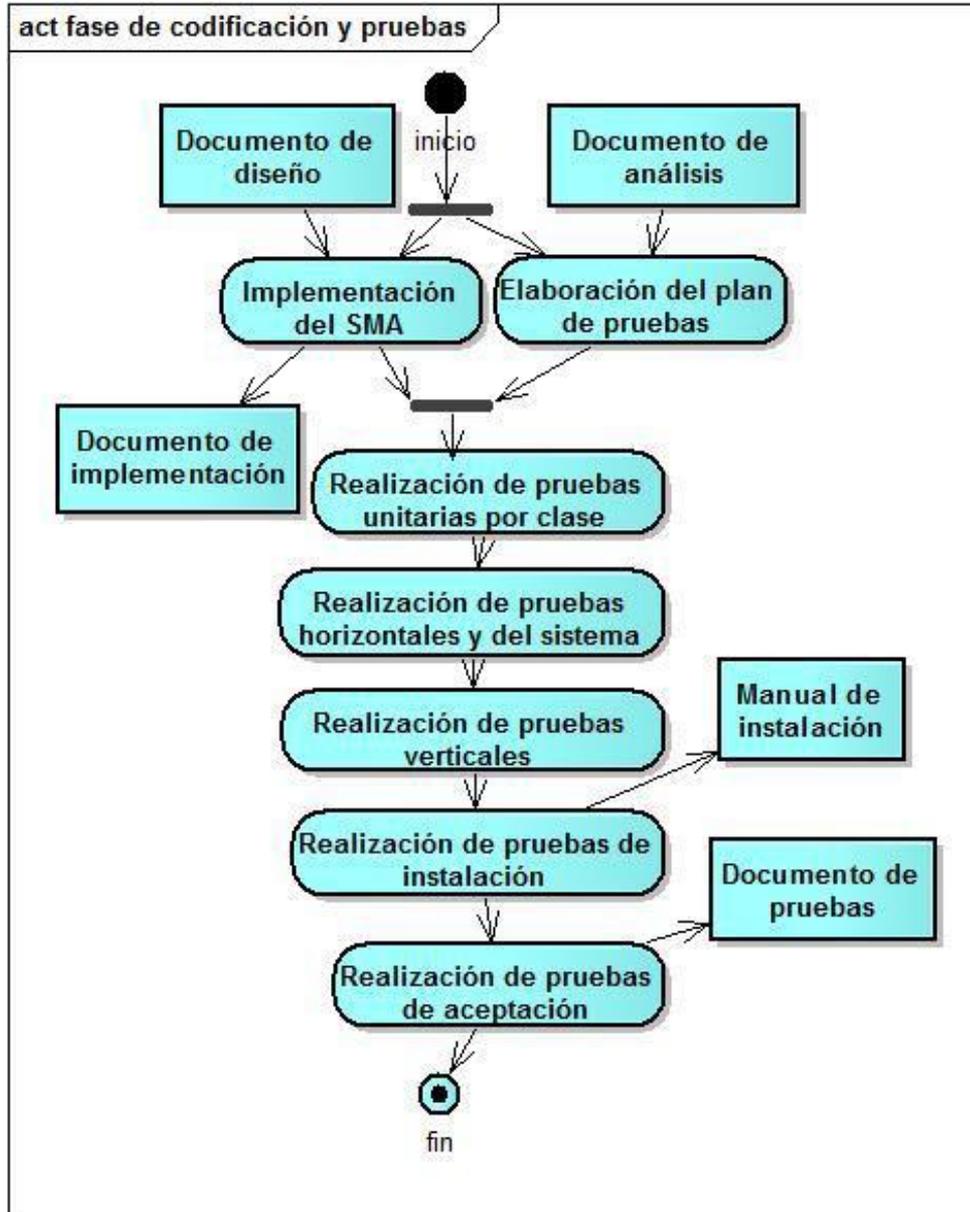
{**pre:** existencia de memoria y {**pos:** se crean componentes del agente o error}  
 Negocio.dat distinto de Null}

1	abrirArchivoConf("Negocio.dat")	<i>asignaUid()</i> : método que asocia el AN con un único repositorio datos de tiempo real <i>listaVariables</i> : Lista que contiene los datos del proceso real
2	Leer(Negocio.dat, ConfNegocio)	
3	uidRepositorioTiempoReal	
4	asignaUid( ) listaVariables: Lista<TablaTiempoReal>	

1	agenteNegocio x $\Rightarrow$ se creó el AN	Se instancia el agente x, si éste se puede crear hay éxito, por el contrario hay error.
2	agenteNegocio x $\Rightarrow$ error	

# MASINA

## Fase de Codificación Y pruebas



# MASINA

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

# Sistema Generador de Código para la Metodología MASINA

Menú

Área de Trabajo

The screenshot shows the SISGECOMA application window. The title bar reads "SISGECOMA". The menu bar contains "Archivos", "Editar", and "Ayuda". The left sidebar, titled "SISGECOMA", shows a tree view with "MGS" expanded, containing "Actos de Habla", "Agentes", "Conversaciones", and "Tareas". The main workspace, titled "servicio", contains three buttons: "Agregar Nuevo Servicio", "Agregar Propiedades", and "Guardar Servicio". Below these are form fields for "Nombre" (Enrutar Mensaje), "Tipo de Servicio" (Dual), "Par. de entrada" (Solicitud de Envío o Recepción de Mensajes), "Par. de salida" (No Aplica), and "Descripción del Servicio" (Recibe y envía mensajes de los agentes de Aplicaciones, de Negocio y del MGS). A status bar at the bottom shows "agente".

Archivos Editar Ayuda

SISGECOMA

MGS

- Actos de Habla
- Agentes
- Conversaciones
- Tareas

servicio

Agregar Nuevo Servicio Agregar Propiedades Guardar Servicio

Nombre: Enrutar Mensaje

Tipo de Servicio: Dual

Par. de entrada: Solicitud de Envío o Recepción de Mensajes

Par. de salida: No Aplica

Descripción del Servicio: Recibe y envía mensajes de los agentes de Aplicaciones, de Negocio y del MGS

agente

Árbol de Búsqueda

# SISGECOMA

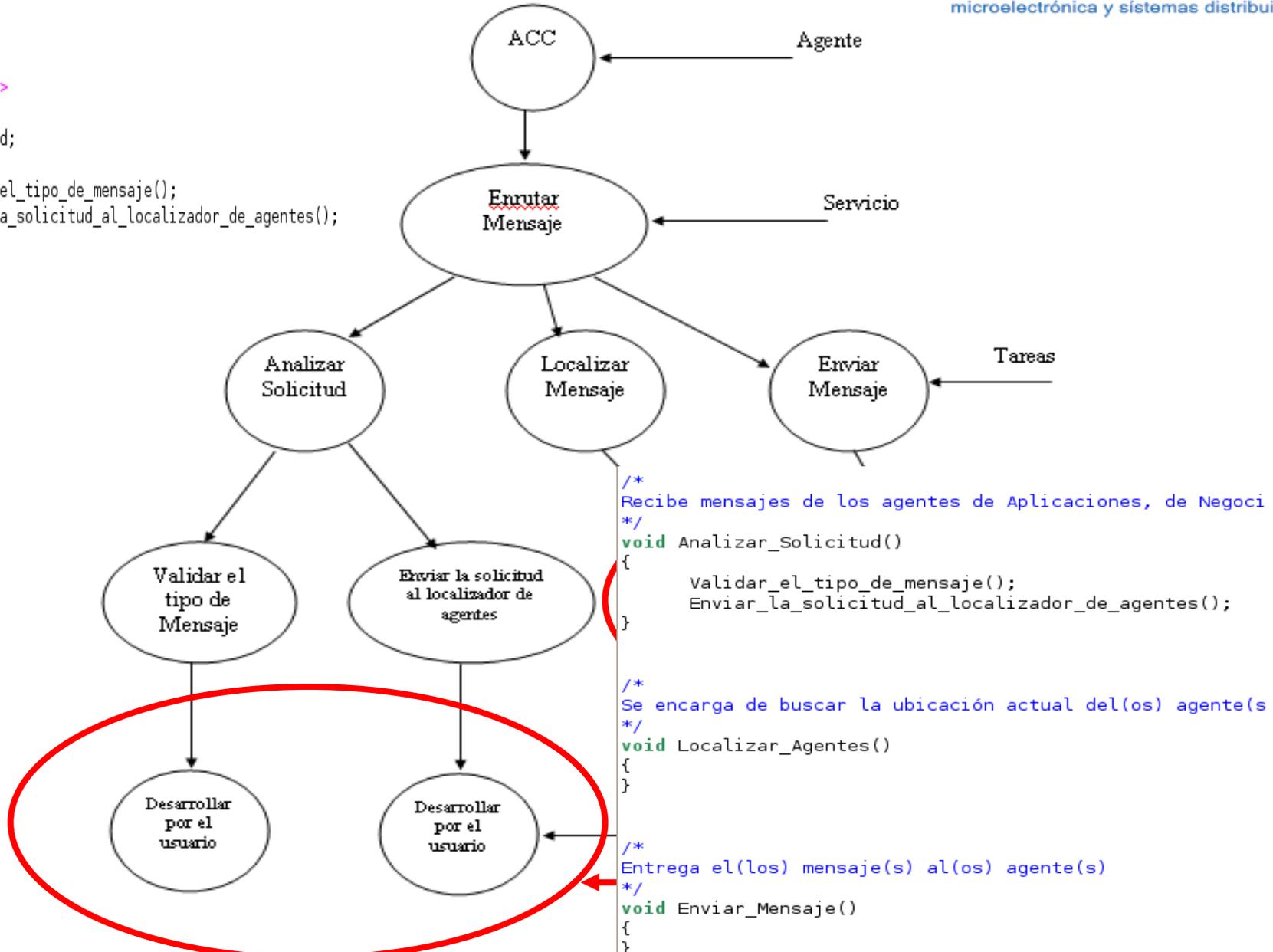
```
#ifndef SUBTAREA_H
#define SUBTAREA_H

#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

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	<pre>for(int i=1; i&lt;11; i++)     cout&lt;&lt;"estoy dando el     paso"&lt;&lt;i&lt;&lt;endl;</pre>



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;  
}
```

# 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.** intentar 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

# Gaia

- Su intención es la de permitir a un analista ir sistemáticamente **desde los requisitos a un diseño suficientemente detallado como para implementar directamente.**
- Los principales conceptos que aparecen en la metodología se dividen en dos:
  - abstractos y
  - concretos
- **Las entidades abstractas** son aquellas que son empleadas durante el análisis para la conceptualización del sistema.
- **Las entidades concretas** son empleadas en el proceso de diseño.

# Gaia

- La entidad más abstracta de un sistema es la “sociedad” u “organización”.
  - Idea: ver un sistema informático como un conjunto de roles teniendo en cuenta una visión organizacional del mundo.
  - **Organización** = una colección de roles que toman parte en patrones sistemáticos de interacción con otros roles
- ☐
- **Roles** = pueden ser instancias de agentes; no fijos necesariamente
  - Roles consisten en 4 atributos: responsabilidades, permisos, actividades y protocolos

# Gaia

- Un rol es asociado a:
  - **un conjunto de permisos**, derechos asociados al rol, identifican los recursos que están disponibles al rol para realizar sus responsabilidades.
  - **Un conjunto de protocolos**, los cuales definen la manera de interactuar del rol con otros roles
- Las **responsabilidades** determinan la funcionalidad, podemos ver dos tipos:
  - **propiedades de viveza**: describen aquellos estados de los asuntos que un agente debe efectuar. Decir que “algo será hecho”
  - **propiedades de seguridad**: son invariantes. condiciones de seguridad

# Gaia

- En el **Proceso de Análisis:**

- Encontrar los roles en el sistema
- Modelar las interacciones entre los roles
- Para cada papel identificar y documentar los protocolos asociados

Role Schema: COFFEEFILLER			
Description:			
This role involves ensuring that the coffee pot is kept filled, and informing the workers when fresh coffee has been brewed.			
Protocols and Activities:			
Fill, InformWorkers, <u>CheckStock</u> , AwaitEmpty			
Permissions:			
	reads	supplied <i>coffeeMaker</i>	// name of coffee maker
		<i>coffeeStatus</i>	// full or empty
	changes	<i>coffeeStock</i>	// stock level of coffee
Responsibilities			
Liveness:			
COFFEEFILLER = (Fill. InformWorkers. <u>CheckStock</u> . AwaitEmpty) <sup>ω</sup>			
safety:			
	•	<i>coffeeStock</i> > 0	

- En el **Proceso de Diseño:**

- Mapear los roles en tipos de agentes
- Crear el número correcto de instancias de agente de cada tipo
- Modelar los servicios necesarios para desarrollar el rol (modelo de servicios), examinando protocolos y propiedades de viveza y seguridad
- Crear el modelo de conocimiento

# Gaia

- El *análisis* incorpora los siguientes modelos:
  - **El Modelo del Prototipo del Rol:** consiste en identificar los roles y dar una breve descripción de los mismos
  - **El Modelo de Interacción:** captura la interacción secuencial entre los roles
  - **El Modelo de Roles Elaborado:** documenta los roles del sistema, sus protocolos y actividades, permisos y responsabilidades
- El *diseño* incluye:
  - **El Modelo de Agente,** en el que se los roles son agregados en tipos de agentes
  - **El Modelo de Servicios,** en el que se identifican todos los servicios (funciones) asociados a cada rol de agente
  - **El Modelo de Interacción,** que simplemente define la interacción existente entre los agentes

# AUML

AUML = AGENT UNIFIED MODELING LANGUAGE

([www.auml.org](http://www.auml.org))

*“Reutilizar UML sólo donde tenga sentido”*

- **UML es insuficiente para modelar sistemas multiagentes**
  - Comparados con los objetos, los agentes son activos ya que actúan por razones que emergen de ellos mismos
  - Sus actividades incluyen objetivos y condiciones que guían la ejecución de las tareas definidas.
  - Toman la responsabilidad de sus necesidades.
  - Los agentes actúan de igual forma solos o con otros agentes.
  - Forman una comunidad social de miembros
  - interdependientes que actúan de forma autónoma.

# AUML

- No se trata de una metodología, sino de un **lenguaje para la documentación de modelos de sistemas.**
- AUML sintetiza el interés por disponer de **metodologías de desarrollo orientadas a agentes** con la aceptación de UML
- Presenta en la actualidad un conjunto de extensiones de UML para:
  - la especificación de protocolos de interacción de Agentes
  - la representación de estructuras sociales y organizativas entre agentes

# AUML

- Inicialmente se identifican dos áreas para el desarrollo detallado de especificaciones:
  - **Diagramas de clases**
    - Especifican el comportamiento interno de un agente y su relación con el exterior usando diagramas de clases UML adaptados
    - Las clases de UML tienen “una extensión” que son los *AgentifiedClass*, *AgentClass* y un *AgentRoleClass*
  - **Diagramas de interacción**
    - Término genérico que se aplica a diversos tipos de diagramas centrados en la interacción entre agentes
    - Similar a los diagramas de interacción usados en UML
    - Varios usos: Diagramas de secuencia, Diagramas de descripción de interacciones, Diagramas de colaboración, diagrama de comunicación, Diagramas de estado

# AUML

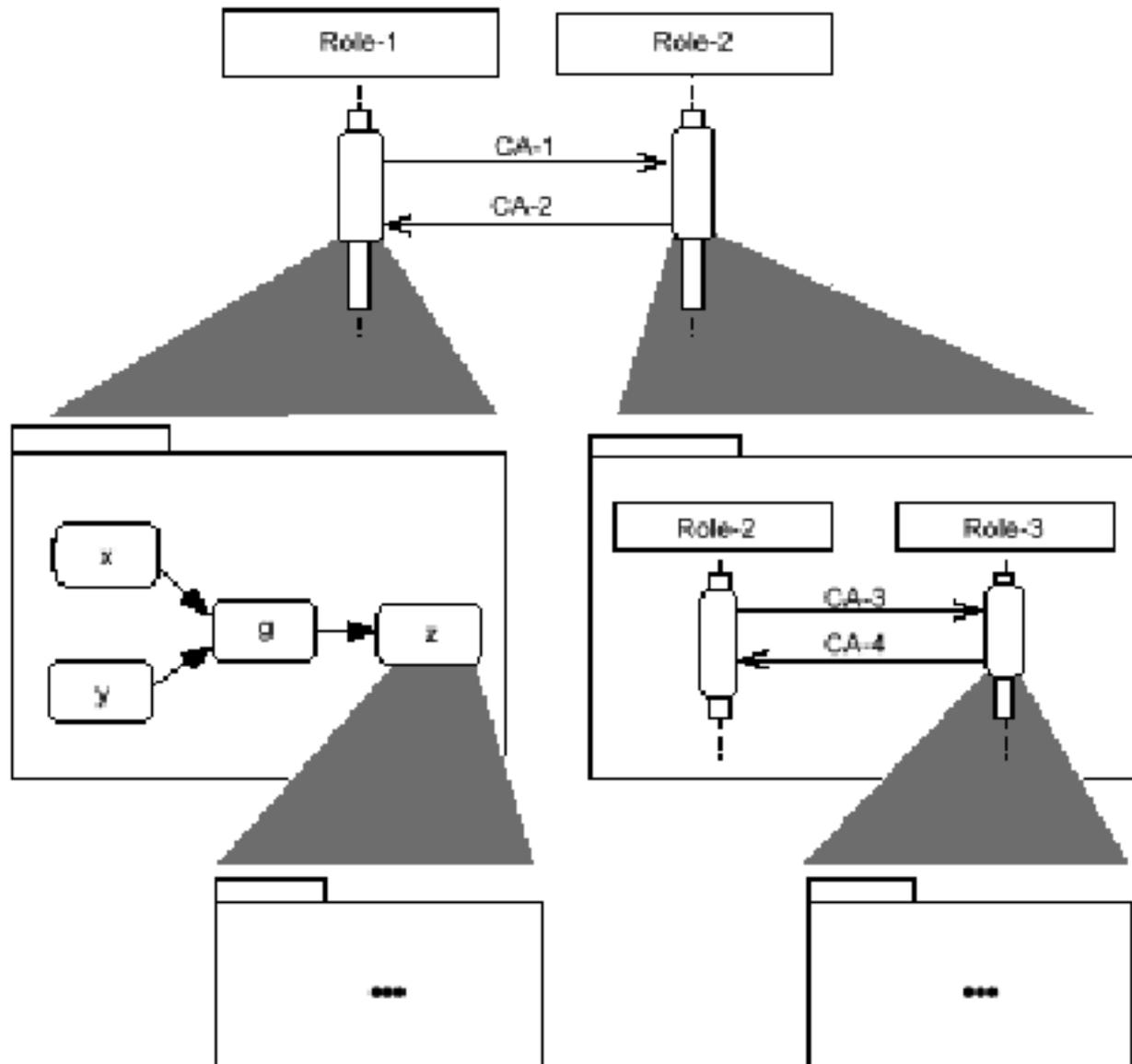
AUML adopta un enfoque por capas de protocolos:

Nivel 1 - Representar el protocolo general (diagramas de secuencia, paquetes, plantillas)

Nivel 2- ¿Representa a las interacciones entre los agentes (diagramas de secuencia, colaboración)

Nivel 3- ¿Representa la parte del Agente interna (diagramas de actividad y estados, de clases)

# AUML: enfoque por capas de protocolos



# Quienes estan interesados en AUML

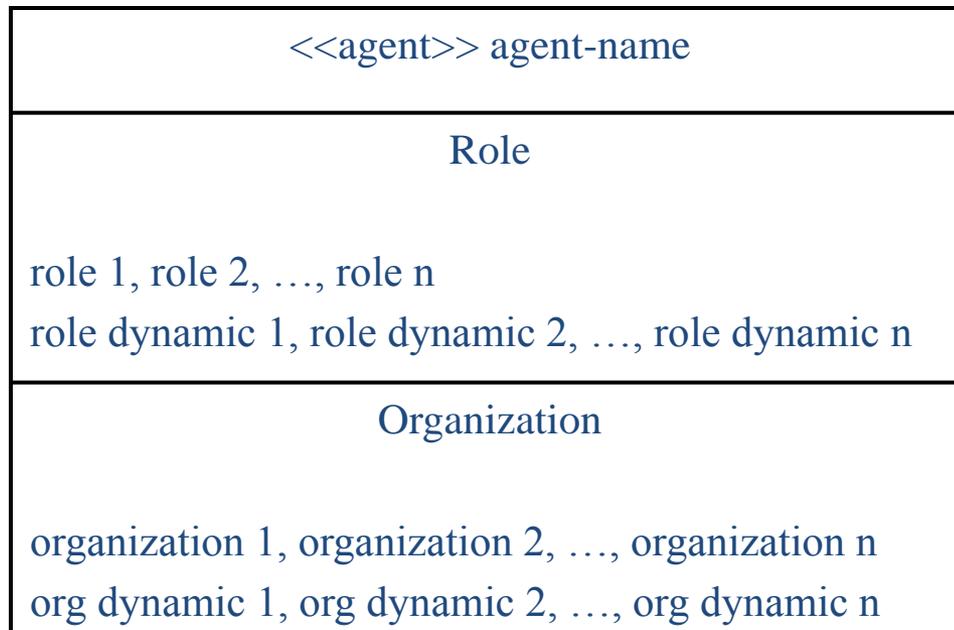
- **OMG Special Interest Group**: recomienda estándares para tecnologías de agentes ([www.omg.org](http://www.omg.org))
- **FIPA Modeling Technical Committee**: trabaja en AUML como estándar ([www.auml.org](http://www.auml.org))
- **Otras metodologias**: MESSAGE, Gaia, Tropos, Prometheus, MaSE, MASINA...

# Agentes comparten

- Identificador
  - Identifica a cada agente en un SMA
- Rol
  - define el comportamiento de un agente en la sociedad (es. Vendedor, etc.)
- Organización
  - Define las relaciones entre los roles (como en los humanos o animales: jerarquías, grupos de interés, etc.)
- Capacidad
  - Especifica que un agente puede hacer y bajo que condiciones
  -
- Servicio
  - Describe la actividad que un agente puede realizar y proveer a otros agentes

# Representación de agentes

Diagrama de clases UML se puede usar para representar parte estática de un agente.



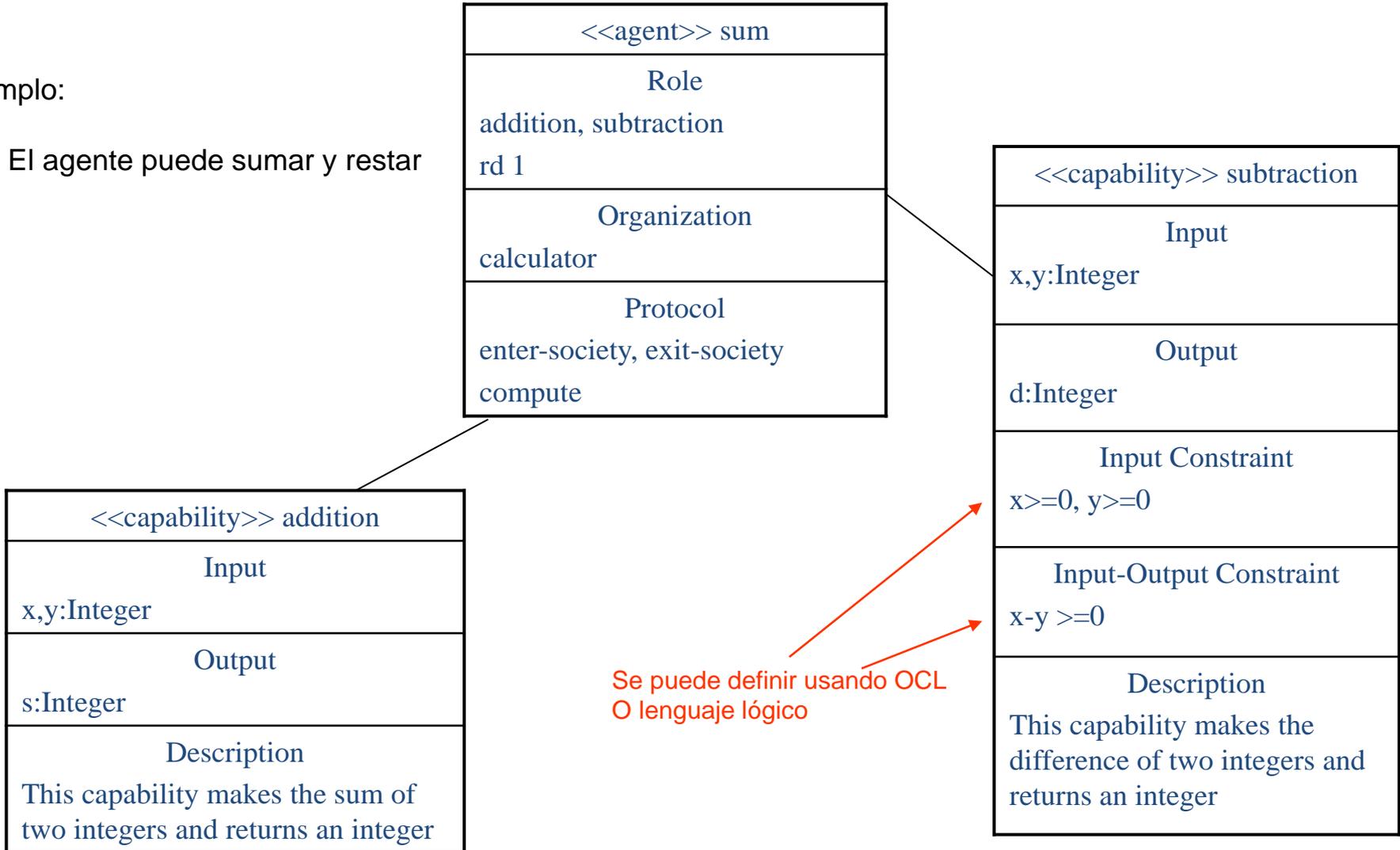
# Capacidades

- Esta compuesta por:
  - Input
    - Entradas para realizar sus tareas
  - Output
    - Qué genera como resultados
  - Input constraints
    - Restricciones para poder realizar sus tareas
  - Output constraints
    - Restricciones para poder usar sus resultados
  - Input-output constraints
    - Restricciones que deben cumplirse en ambos casos
  - Description
    - Descripción en lenguaje natural de las capacidades

# Capacidades

- Ejemplo:

El agente puede sumar y restar



# Servicio

- Esta compuesto por:
  - Name
    - Nombre del servicio
  - Description
    - La descripción
  - Type
    - El tipo
  - Protocol
    - Protocolos de interacción soportados por el servicio
  - Agent communication language
    - Lenguaje de comunicación usado
  - Ontology
    - Lista de ontologías usadas
  - Content language
    - Lenguaje de contenido
  - Properties
    - Propiedades del servicios

# Servicio

<<agent>> sum
Role addition, subtraction rd 1
Organization calculator
Protocol enter-society, exit-society compute

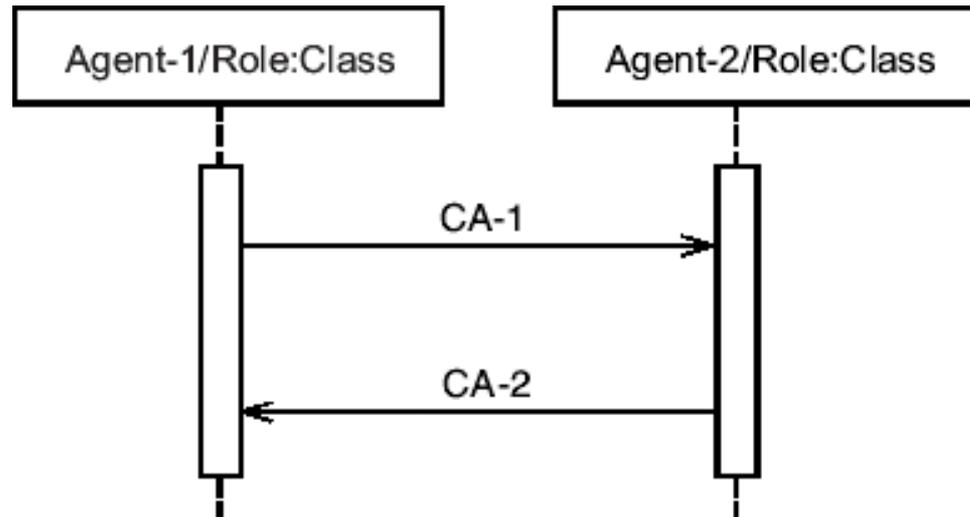
<<service>> computation
Description This service makes an addition when requested by the request addition protocol and makes a subtraction when requested by the request-subtraction protocol
Type computation
Protocol request-addition request-subtraction
Agent Communication Language FIPA ACL
Ontology computation ontology
Content Language FIPA SL

- Ejemplo:

El agente da servicios de sumar y restar

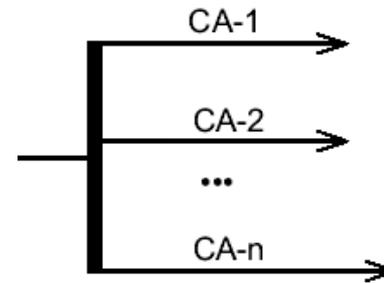
# Interacciones

Interacciones entre agentes se define usando diagrama de secuencia de UML

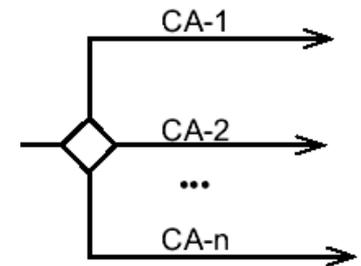


# Interacciones Concurrentes

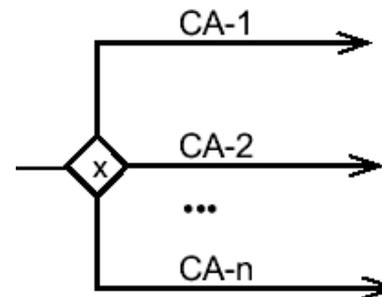
- A) Los actos de comunicación simultáneos procedentes de CA-1 a CA-n se envían en paralelo.
- b) Una selección de las n actos se envía en paralelo (cero o más).
- c) Elección exclusiva: sólo uno de los actos de comunicación se envía.



(a)



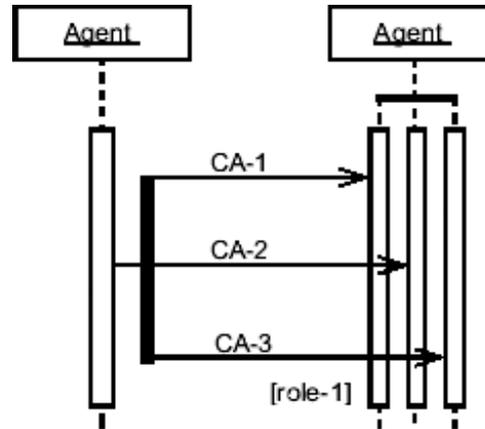
(b)



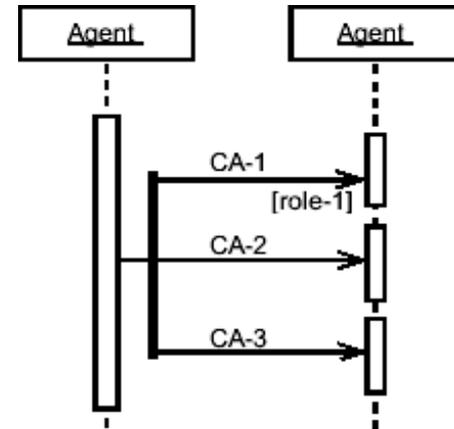
(c)

# Interacciones Concurrentes

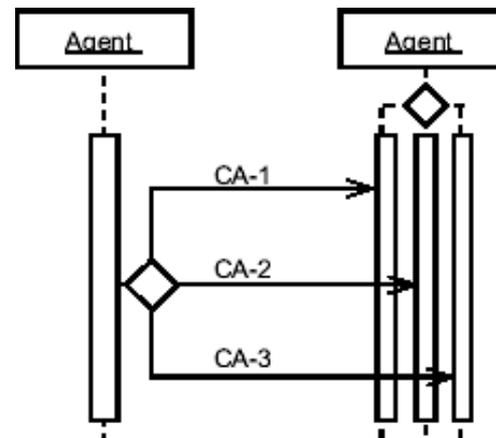
- a) Un agente envía 3 CA concurrente a otro agente. El diagrama se puede interpretar de dos maneras diferentes:  
Cada CA es procesado desde el mismo agente/papel con un hilo diferente de la ejecución  
Cada CA es procesada por un papel diferente del agente (en este caso los mensajes pueden ser anotada especificando el papel)
- b) La misma semántica de (a), pero con una notación más simple
- c) Elección de tres actos de comunicación diferente recibido por tres hilos diferentes (o roles)



(a)



(b)

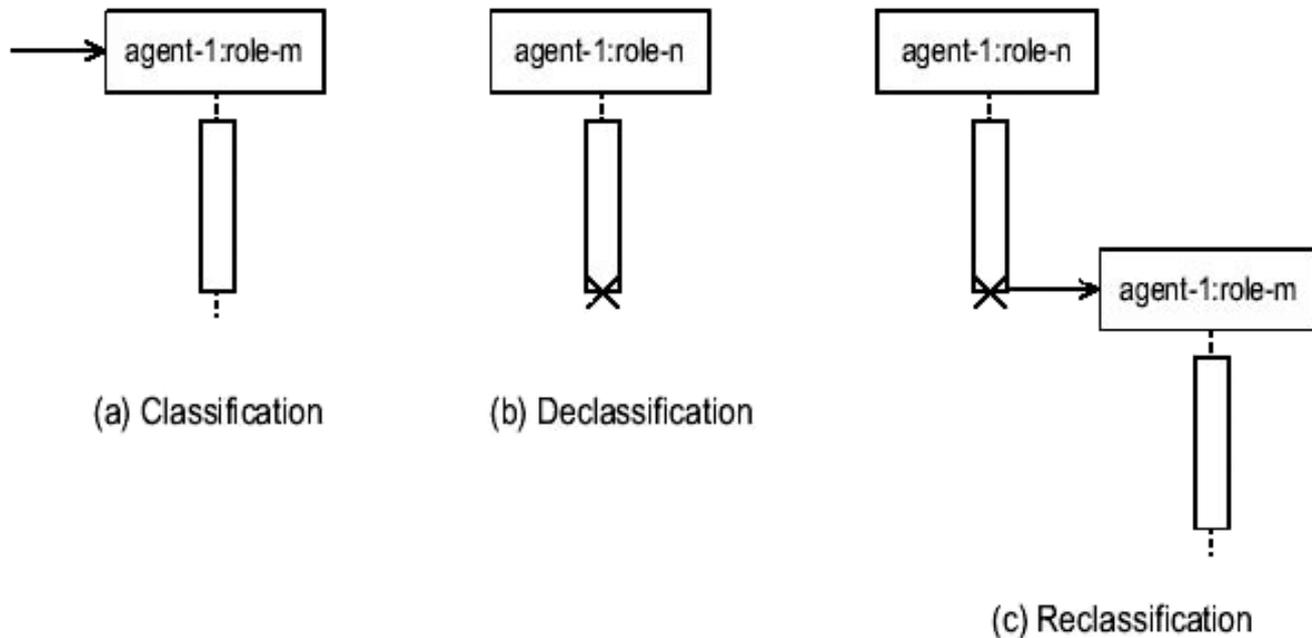


(c)

NOTA: cada CA concurrente puede ser enviado a diferentes agentes

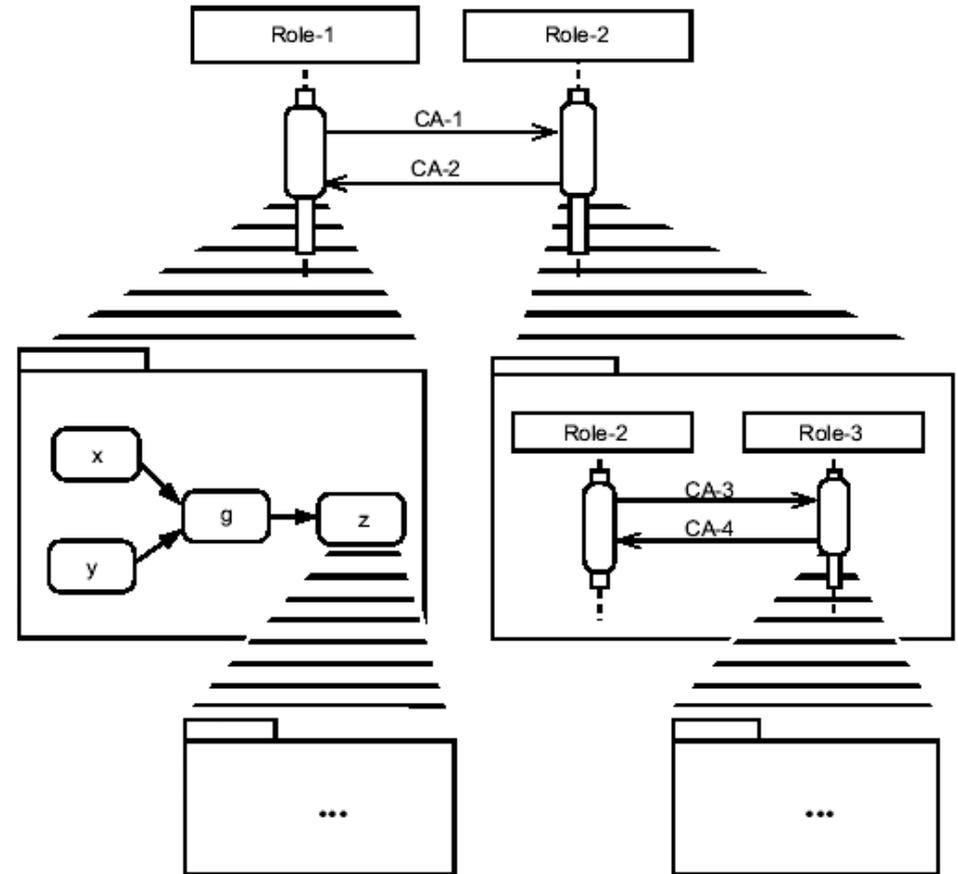
# Roles

- diagrama de secuencia de UML para representar cambios en roles



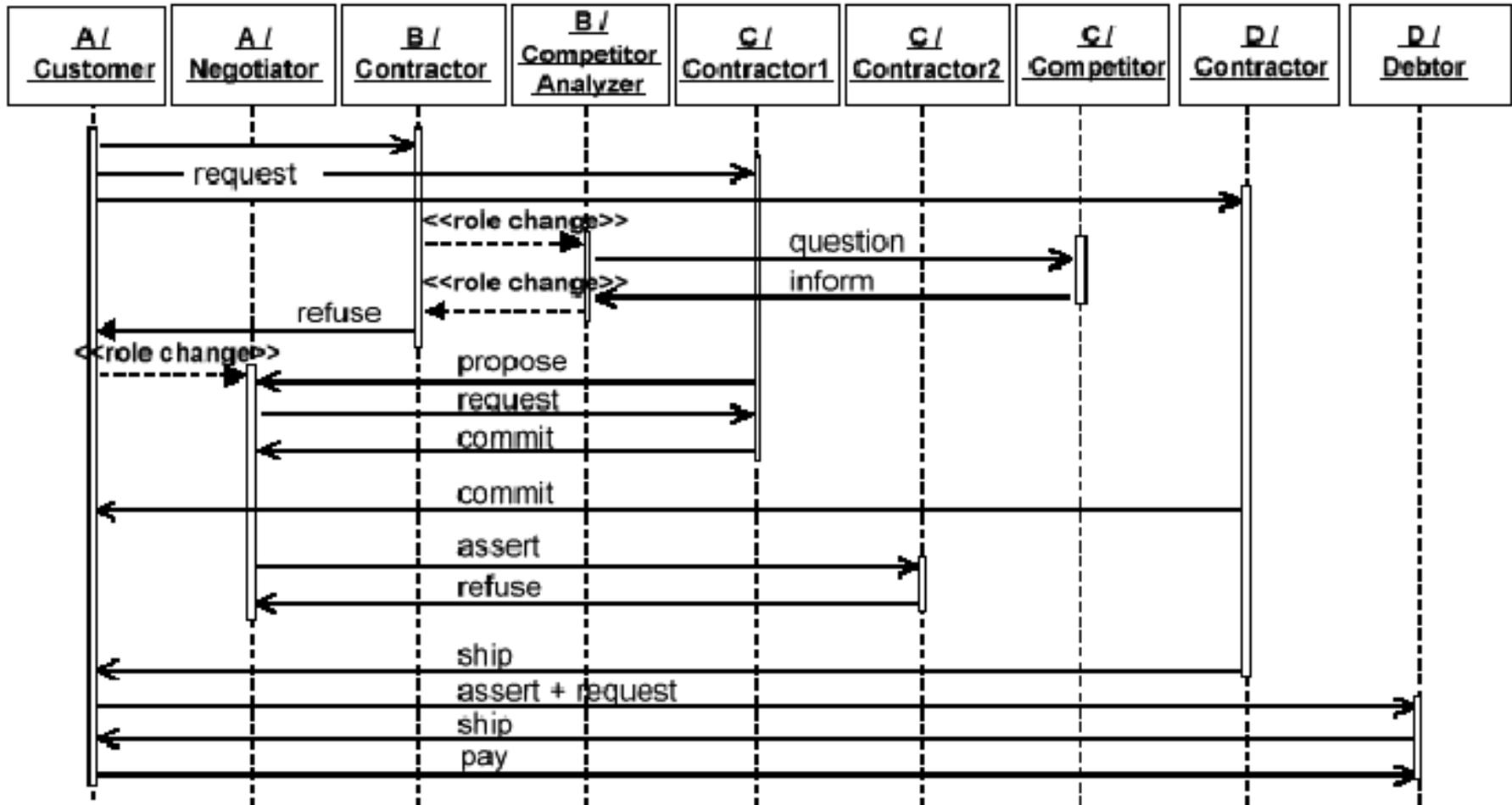
# Especificación

- Cualquier proceso de interacción se puede expresar en más detalle.
- La "nivelación" puede continuar hasta que el problema se ha especificado adecuadamente para generar código.
- También los diagramas de actividad y gráficos de estado pueden ser utilizados.



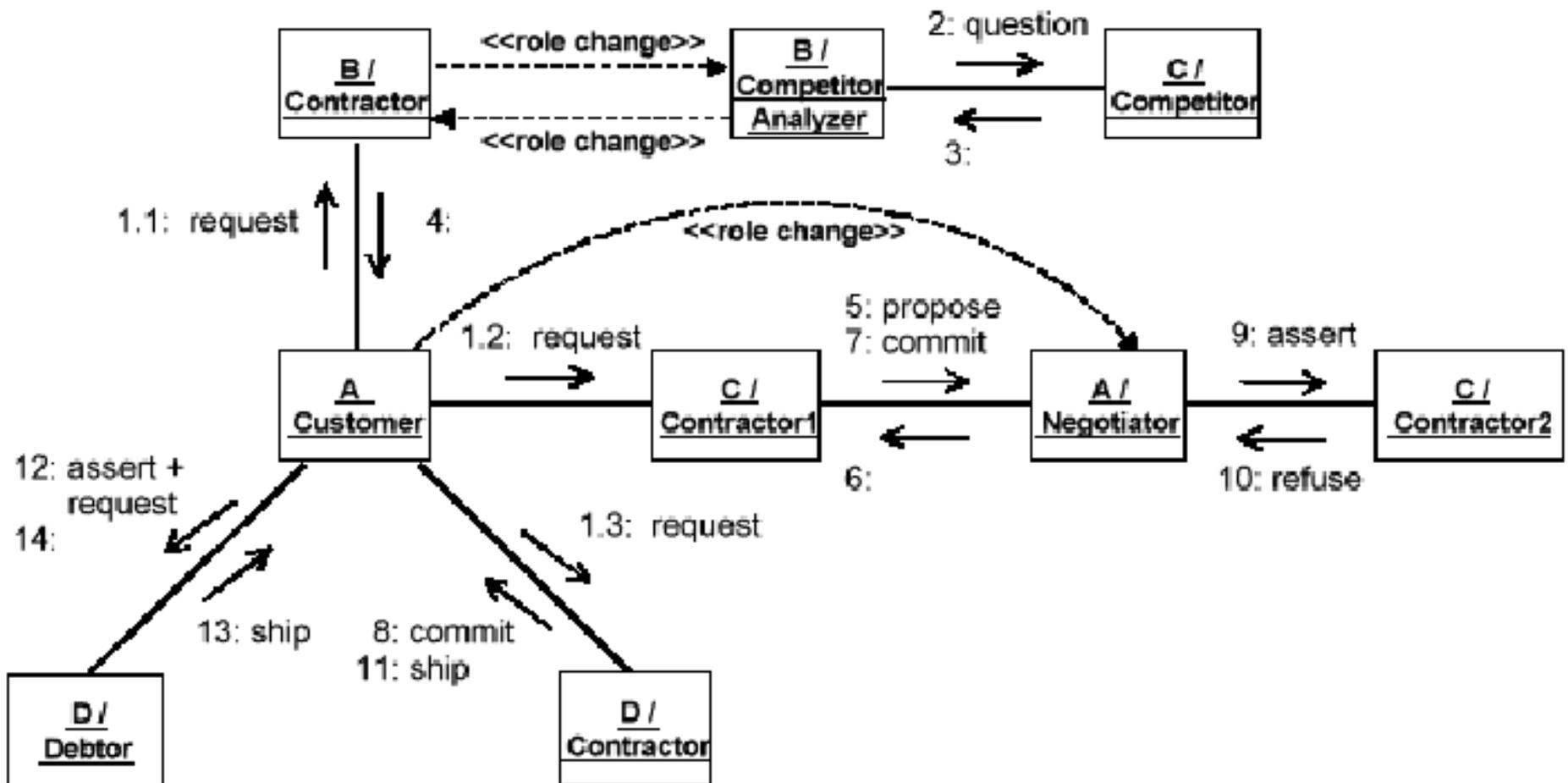
# Diagramas de secuencia Extendidos

Expresan roles de los agentes



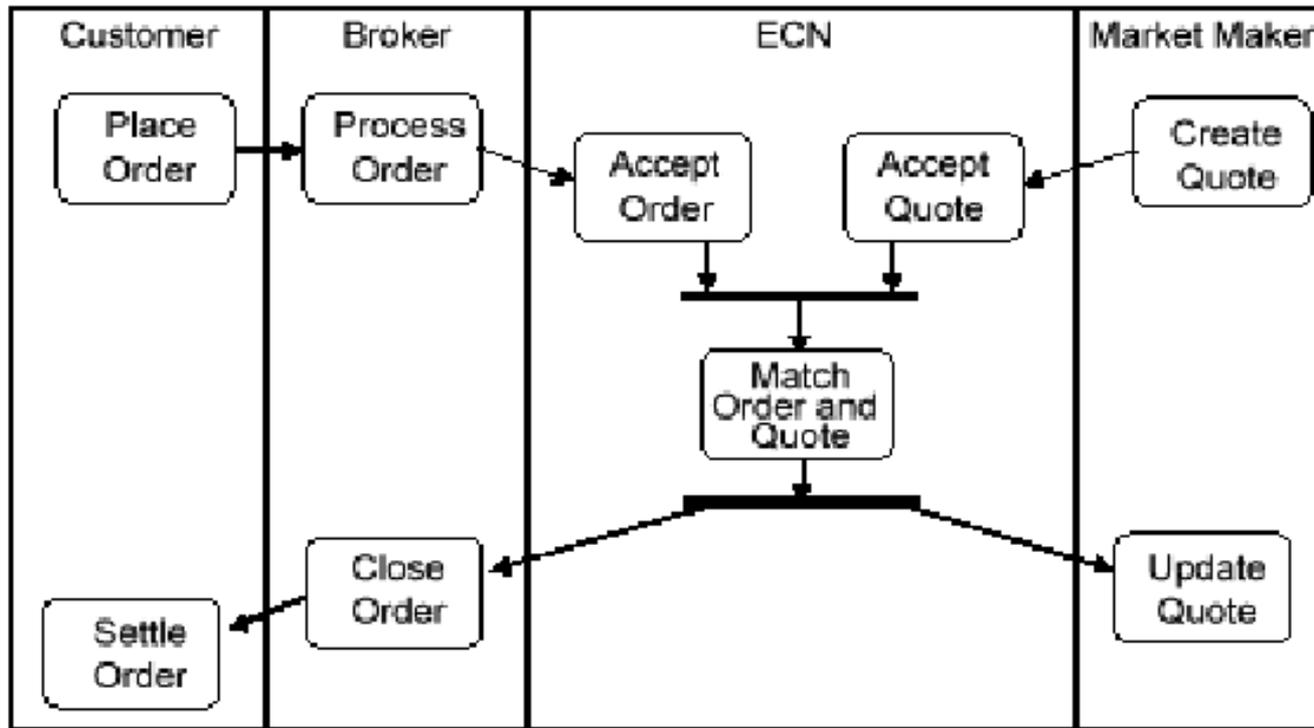
# Diagrama de colaboración

otra forma de mostrar interacción entre agentes



# Diagram de Actividades Extendido

- Proporciona un hilo explícito de control?
- Útil para los protocolos de interacción complejos que involucran concurrencia



# Multiagent System Engineering (MaSE)

- Diseñada para desarrollar multiagentes de propósito general
- **Análisis**
  - Definir metas a partir de los requerimientos
  - Definir roles necesarios para satisfacer las metas

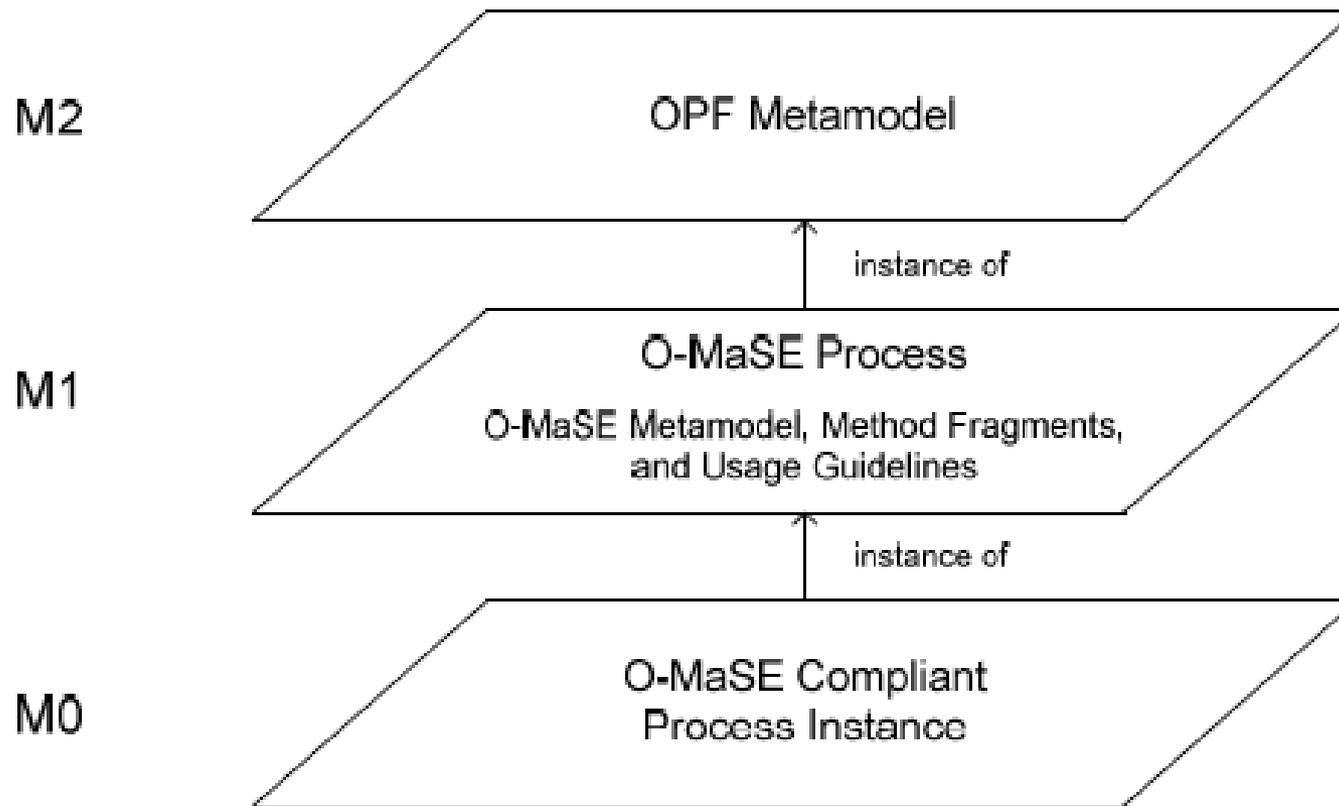
# MaSE

- **Diseño**
  - Definir clases de agentes basado en los roles
    - Un agente puede ejecutar varios roles
    - Un rol puede ser dividido y ejecutado por varios agentes
  - Construir conversaciones
  - Ensamblar agentes
    - Definir la arquitectura
    - Definir los componentes de la arquitectura
  - Diseño del sistema

# Problemas de MaSE

- No provee un mecanismo para modelar la interacción de los agentes con su ambiente
- MaSE produce un sistema multiagente con una organización definida. Estos sistemas deben ser capaces de diseñar y adaptar su organización dinámicamente
- MaSE no incluye el concepto de sub-equipos, todos los agentes pertenecen a una misma capa
- Las conversaciones entre agentes se diseñan a muy bajo nivel. Entender el proceso de comunicación entre agentes se vuelve complicado

# O-MaSE Framework



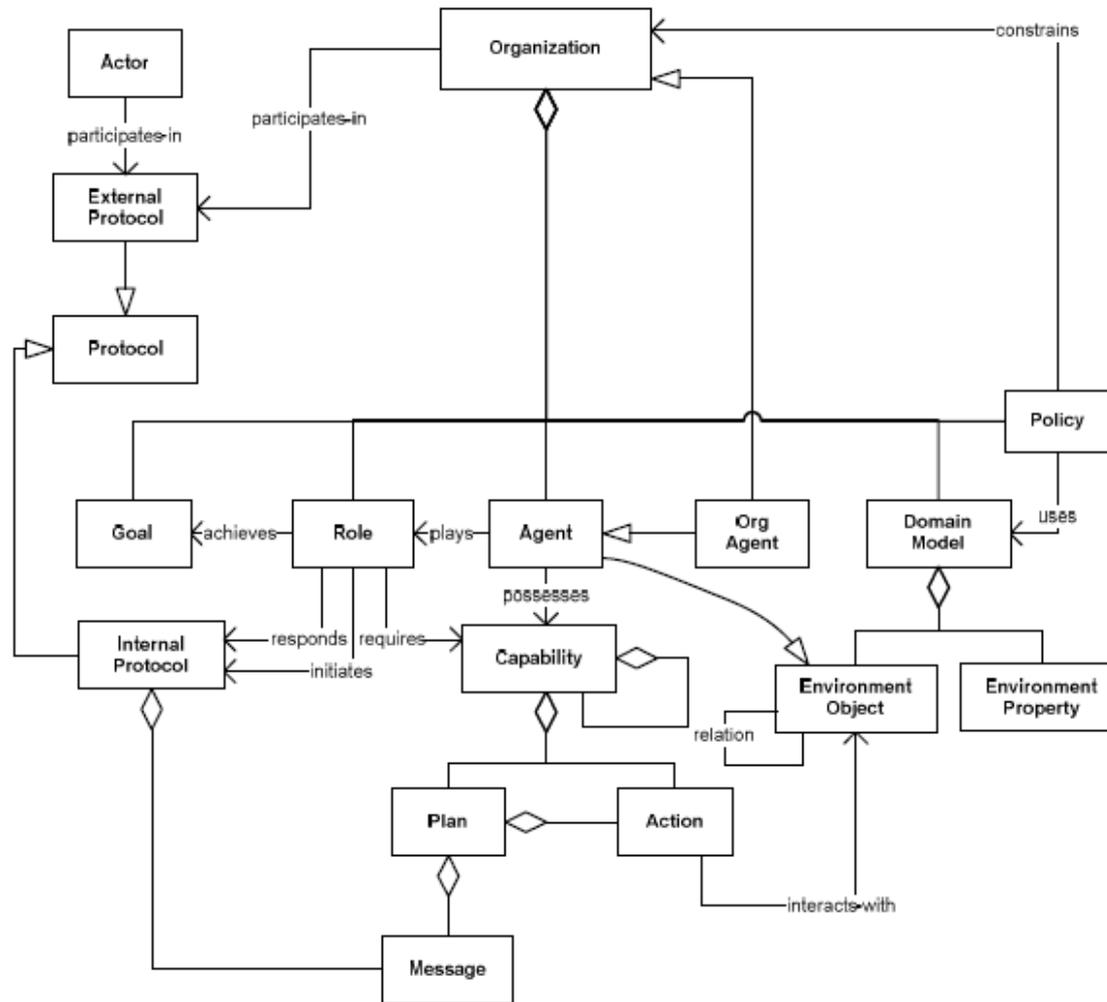
# O-MaSE: Metamodelo

- Define los conceptos principales utilizados en los sistemas multiagente
- Basado en un enfoque organizacional
- Organización
  - Metas
  - Roles
  - Agentes
  - Modelo de dominio
  - Políticas

# O-MaSE: Metamodelo (cont.)

- Meta: función u objetivo de la organización
- Rol: posición dentro de la organización que intenta alcanzar una meta
- Agente: percibe el ambiente y ejecuta acciones
  - Capacidades
    - Planes
    - Acciones
- Modelo del dominio: descripción del ambiente
- Políticas: reglas de la organización

# O-MaSE: Metamodelo (cont.)



# O-MaSE: Fragmentos de métodos

- Define actividades a realizar durante el proceso de desarrollo de software
- FIPA se encarga de desarrollar (agrupar) estos fragmentos [4]
- Definido en términos de:
  - Escenarios
  - Unidades de trabajo
    - Actividades
    - Tareas
    - Técnicas
  - Productores
  - Productos
  - Lenguajes

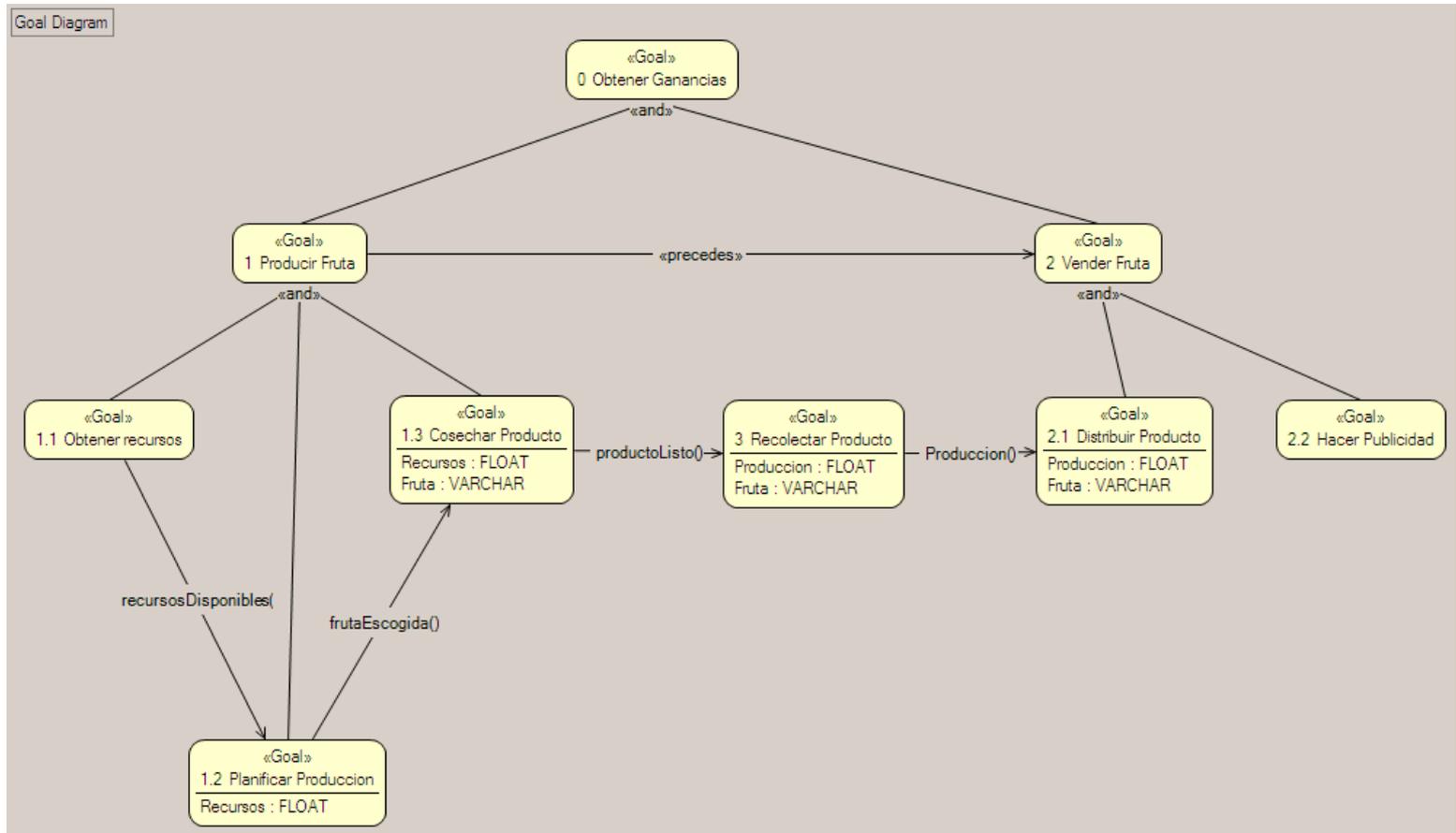
# O-MaSE: Guías

- Combinar los fragmentos de métodos para obtener procesos
- Los procesos describen la metodología final que se va a utilizar
- Se especifican como (Entrada, Salida, Precondiciones, Postcondiciones)
  - Entrada y salida: un conjunto de productos
  - Condiciones: estado de productos y productores

# O-MaSE: tareas

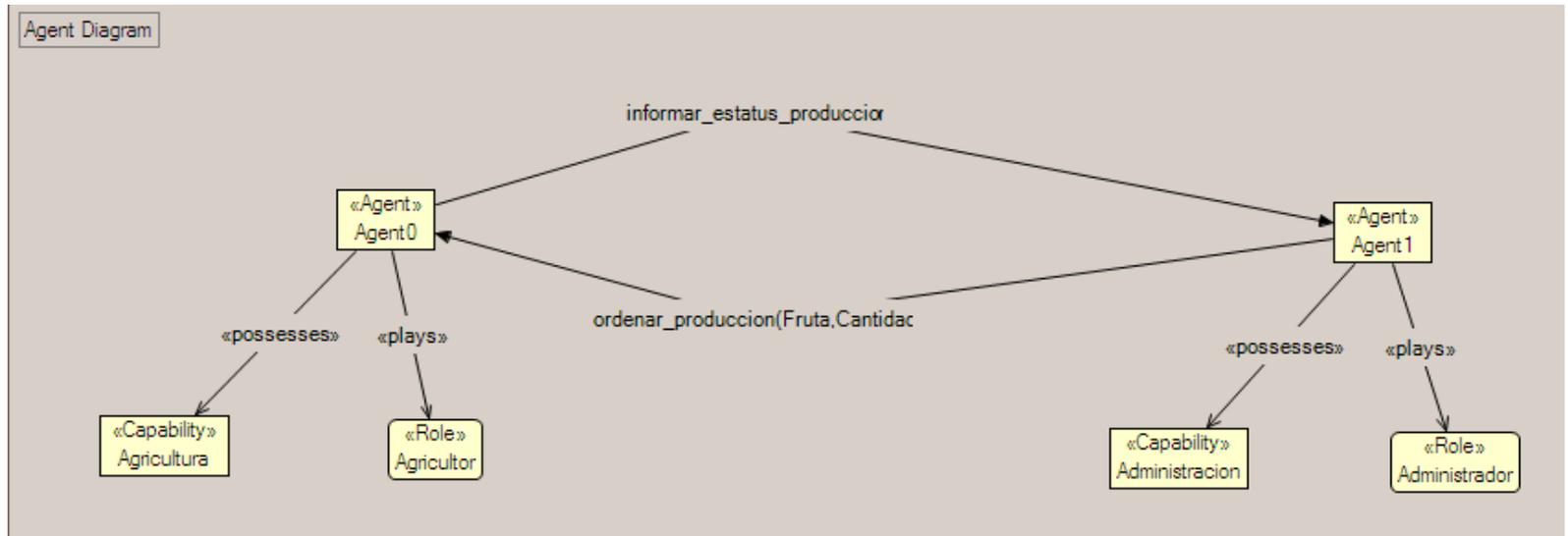
- Modelar metas
- Refinar metas
- Modelar clases de agentes
- Modelar protocolo (interacción)
- Modelar plan

# Modelo de Metas (con algo de refinamiento)

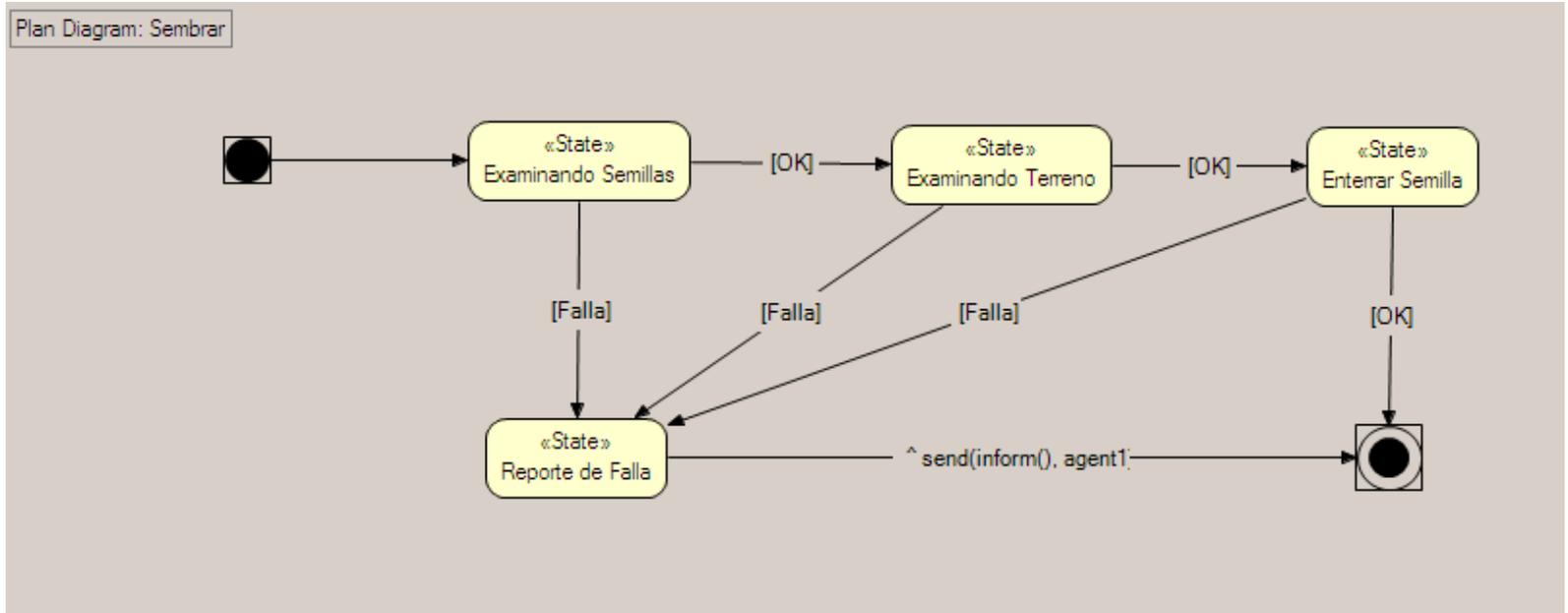




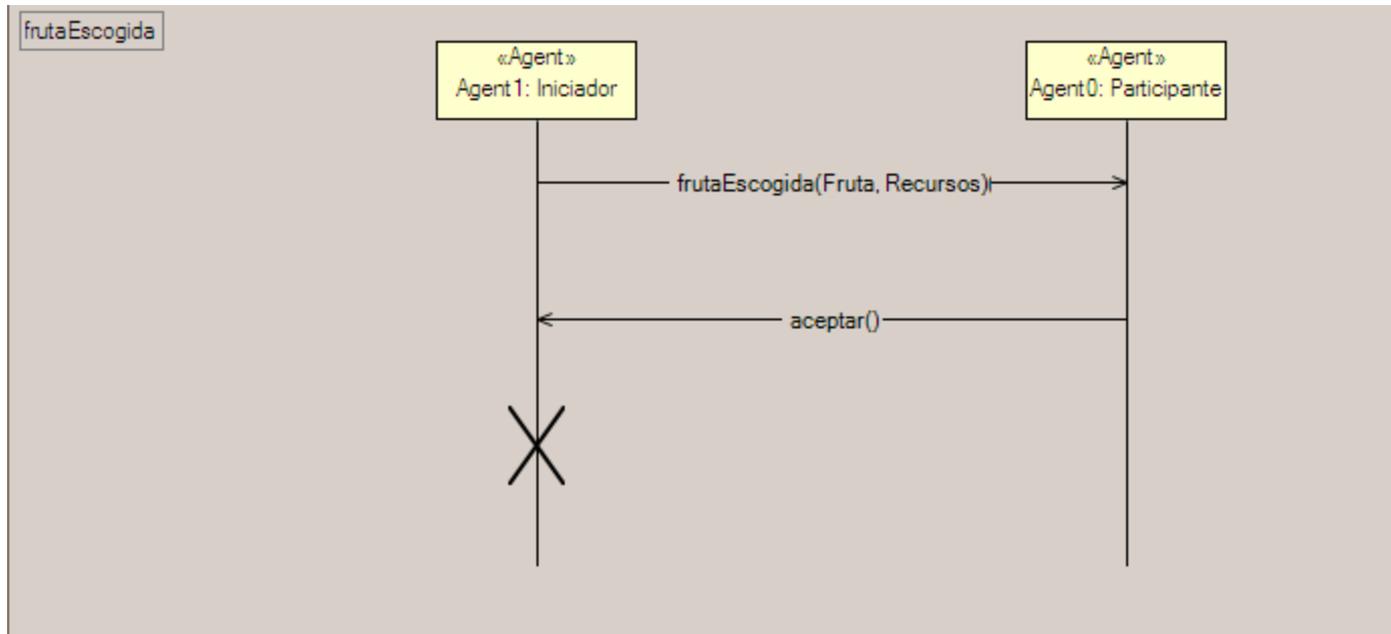
# Modelo de Agentes



# Modelo de Planes (Ej)



# Modelo de Protocolos (Ej)



# Comparación Metodologías

	ANÁLISIS	DISEÑO	UML	FASES
<b>GAIA.</b>	Modelo de Roles Modelo de Interacción	Modelo de Agente Modelo de Servicios Modelo de Conocimiento	-	AD
<b>MASSIVE.</b>	Vista de Tareas Vista de Entorno Vista de Sistema	Vista de Roles Vista de Interacción Vista de Sociedad Vista de Arquitectura Vista de Sistema	Si	ADI
<b>AUML</b>			Si	A

# Comparación Metodologías

	ANÁLISIS	DISEÑO	UML	FASES
<b>MaSE</b>	Captura de Objetivos Aplicación casos de uso Transformación de Roles Creación clases de agente	Construcción de conversaciones Ensamblado clases de agente Diseño del sistema	Si	ADI
<b>MASINA</b>	Fase de Conceptuación Modelo de Organización Modelo de Agente Modelo de Tareas Modelo de Inteligencia M. Comunicación y Coord.	Diseño de la Red de agentes Diseño de Agentes Diseño de Plataforma	Si	ADI

# 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