Ingeniería de Agentes Software y Físicos

Metodologías de desarrollo de Sistemas Multi-Agente

Juan Pavón Mestras

Dep. de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense Madrid

http://grasia.fdi.ucm.es

MINISTERIO DE EDUCACION Y CIENCIA

Financiado por la Dirección General de Universidades

Sistemas multi-agente (SMA)

Para un desarrollador,
¿qué son los agentes software?

¿Cómo se construye un sistema multi-agente?

- *Cómo se concibe un SMA*
- *Cómo se programa*
- *Qué herramientas hay de desarrollo*
- *Qué plataformas hay de ejecución*

Sistemas multi-agente (SMA)

- Los agentes software tienen un conjunto de características que habrá que tener en cuenta para su desarrollo
  - No son simplemente objetos distribuidos pero se pueden implementar como tales
  - Tampoco son sistemas expertos pero su comportamiento podría implementarse con conceptos similares
  - Podrían verse como un tipo de componentes software pero no se plantean como las técnicas actuales de componentes (J2EE, .NET, CCM)
  - Son estas cosas y algo más…

SMA vs. OO

Objetos

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

Agentes

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

SMA vs. Sistemas Expertos

Sistemas Expertos	Agentes
Sistemas cerrados	Interactúan con el entorno
Sistemas de decisión centralizados	Distribución de la toma de decisiones: *Comportamiento emergente*
Interacción con el usuario bajo petición del usuario	Mayor grado de interacción con el usuario Interacción con otros agentes

Sobre la utilidad de los agentes

- La primera pregunta que habrá que plantearse cuando se va a realizar un sistema es:

¿Hace falta utilizar agentes?

¿O bastaría con objetos, componentes, ...?

Sobre la utilidad de los agentes

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

Sobre la utilidad de los agentes

- 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 abstraccion que los objetos o componentes:
  - Mayor autonomía y capacidad de decisión
  - Varios componentes heterogéneos que mantienen relaciones entre ellos y con escalas de tiempo diferentes
  - Modelado de sistemas naturales y sociales
- Facilitan la evolución:
  - Adaptación a modificaciones y al entorno
  - Escalabilidad: añadir agentes para soportar mayor carga de trabajo
  - Añadir/quitar funcionalidad en tiempo de ejecución
  - Desarrollo incremental
  - Sistemas abiertos: capacidad de aceptar nuevos elementos
- Pero no siempre son la solución ideal
  - Ausencia de control/visión global del sistema

Aplicaciones

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

Personalización de servicios

Flexibilidad de la distribución

Delegación de tareas

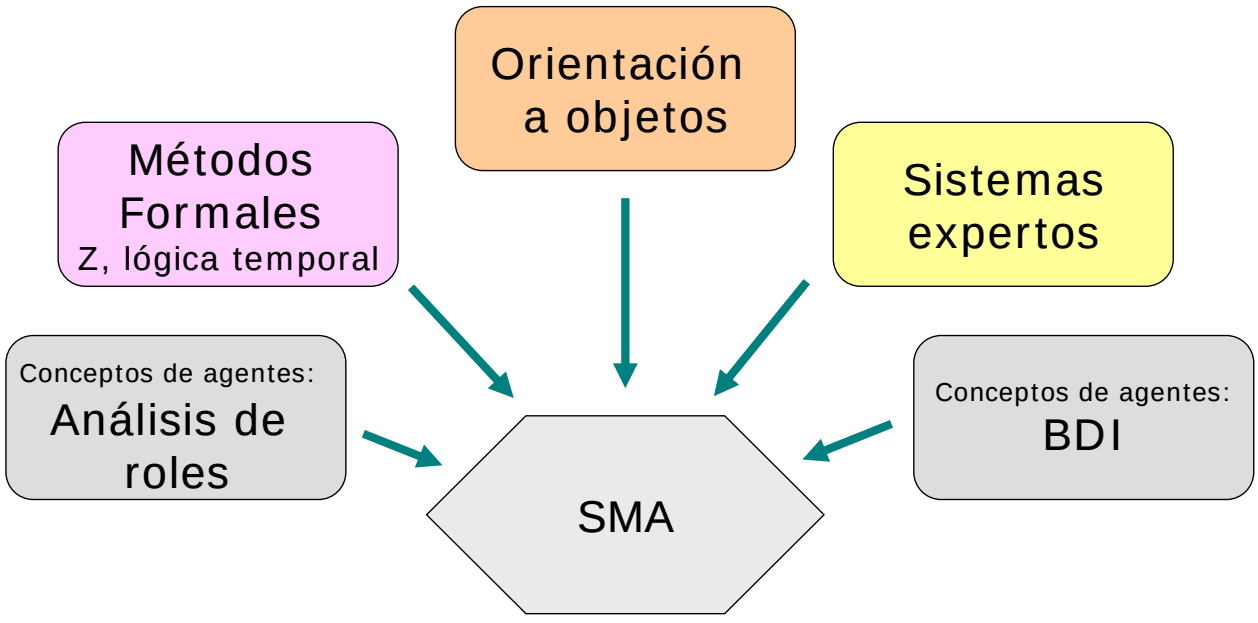
Metodologías de desarrollo de SMA

- Una vez decidido construir un SMA habrá que ver
  - Qué resultados producir
    - Documentación
    - Código, prototipos, pruebas
  - Con qué lenguaje especificar el SMA
    - Visuales: Data Flow Diagrams, Entity-Relationship diagrams, Message Sequence Charts, UML
    - Formales: Z, redes de Petri
  - Qué actividades para producir los resultados
    - De análisis, diseño, implementación, validación, ...
  - Cómo: Guías
  - Métricas
  - Con qué herramientas
    - Desarrollo
    - Entorno de ejecución

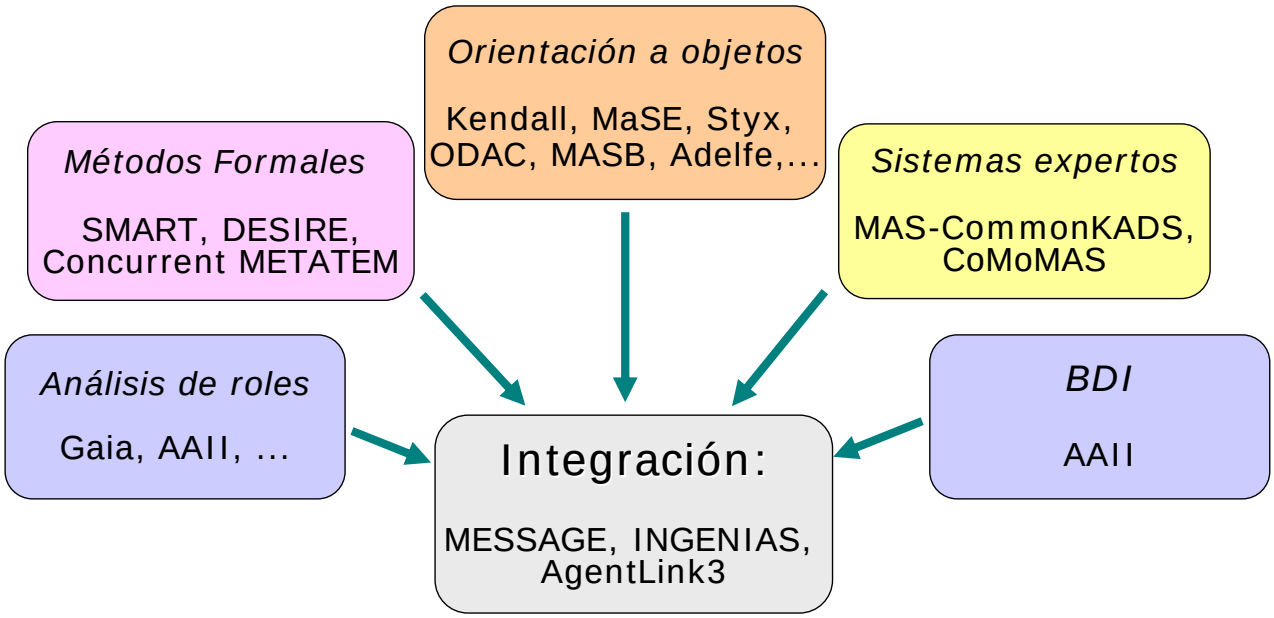
Metodologías de desarrollo de SMA

- La respuesta:
  - Metodologías de desarrollo de SMA
    - Pero... *¿cuál elegir?*

Metodologías para desarrollar SMA



Metodologías para desarrollar SMA

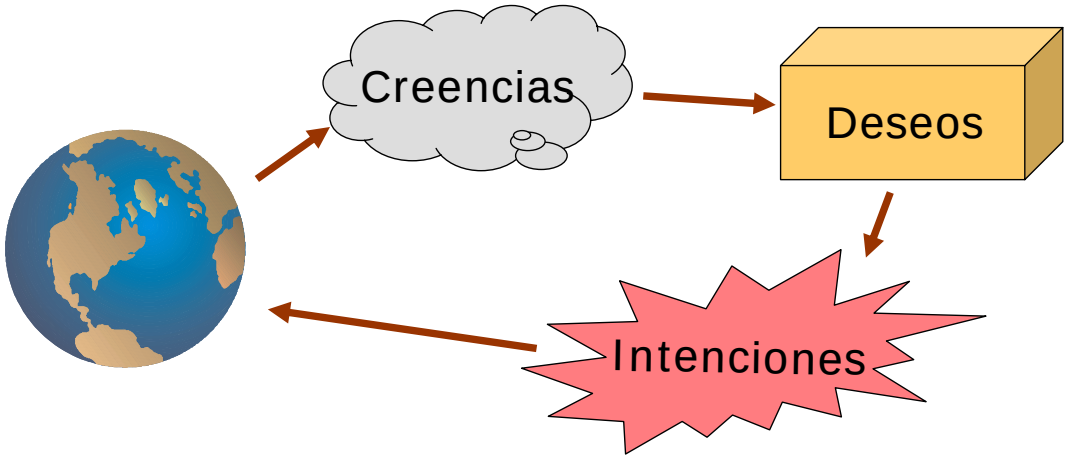


Algunas metodologías

- AAII (Australian Artificial Intelligence Institute)
  - Basada en el modelo BDI
- Gaia
  - SMA como conjunto de entidades que interactúan
- MaSE
  - OO con conversaciones entre objetos
- Tropos
  - Énfasis en la gestión de requisitos
- Zeus
  - Entorno visual de desarrollo de agentes
  - Prototipos de agentes
- MAS-CommonKADS
  - CommonKADS extendido con OO, SDL y MSC
- MESSAGE, INGENIAS
  - Meta-modelado, agentes que siguen el principio de racionalidad de Newell

Metodología AAII [Kinny, Georgeff y Rao 96] (Australian Artificial Intelligence Institute)

- Modelo BDI (Beliefs, Desires, Intentions) [Bratman87]



Metodología AAII [Kinny, Georgeff y Rao 96] (Australian Artificial Intelligence Institute)

- Punto de vista externo
  - Modelo de agentes: jerarquía de clases
  - Modelo de interacciones: responsabilidades de los agentes, servicios que proporcionan, interacciones asociadas, relaciones de control entre agentes
- Basado en análisis de roles:
  1. Identificar roles del dominio de aplicación
     - Primera definición de clases de agentes
  2. Para cada role, identificar las responsabilidades asociadas y servicios que proporciona
     - Descompone las clases de agentes hasta el nivel de servicios
  3. Para cada servicio,identificar las interacciones asociadas
     - Modelo interno de cada clase de agente
  4. Refinar la jerarquía de agentes
     - Definir superclases cuando hay clases de agentes con similitud
     - Componer clases de agentes con herencia o agregación
     - Introducir clases concretas de agente teniendo en cuenta aspectos específicos de implementación

Metodología AAII [Kinny, Georgeff y Rao 96] (Australian Artificial Intelligence Institute)

- Punto de vista interno (basado en BDI)
  - Modelo de creencias: información sobre el entorno, estado interno del agente y acciones que puede realizar
  - Modelo de objetivos: objetivos que puede adoptar el agente y eventos a los que puede responder
  - Modelo de planes: secuencias de acciones que puede emplear el agente
- Basado en el análisis del propósito de los servicios y su descomposición hasta llegar a planes:
  1. Analizar los medios para alcanzar los objetivos
     - Descomposición de cada objetivo en acciones y subobjetivos
     - Generar planes
  2. Construir las creencias del sistema
     - A partir de las condiciones que controlan la ejecución de actividades, y requisitos de entrada y salida para cada objetivo
- El refinamiento de los modelos internos realimenta los modelos externos

GAIA [Wooldridge, Jennings y Kinny, 2000]

- Extiende la metodología Fusion [Coleman et al. 94]
- SMA como organización de entidades que interactúan
- Análisis: Basado en análisis de roles en interacción
  - Modelo de roles:
    - Para cada rol (un individuo, departamento u organización de la vida real):
      - Responsabilidades: funcionalidad del agente (como propiedades de viveza y seguridad)
      - Permisos: derechos a información y recursos
      - Actividades: acciones privadas (sin interactuar con otros agentes)
      - Protocolos
  - Modelo de interacciones: define los protocolos (entre roles)
    - Atributos: propósito, iniciador, respondedor, entradas/salidas, proceso
- Diseño: Basado en agrupación de roles en agentes
  - Modelo de agentes: tipos de agentes (agente=conjunto de roles) y sus instancias
  - Modelo de servicios: funciones de cada rol
  - Modelo de conocidos: con quienes puede interaccionar

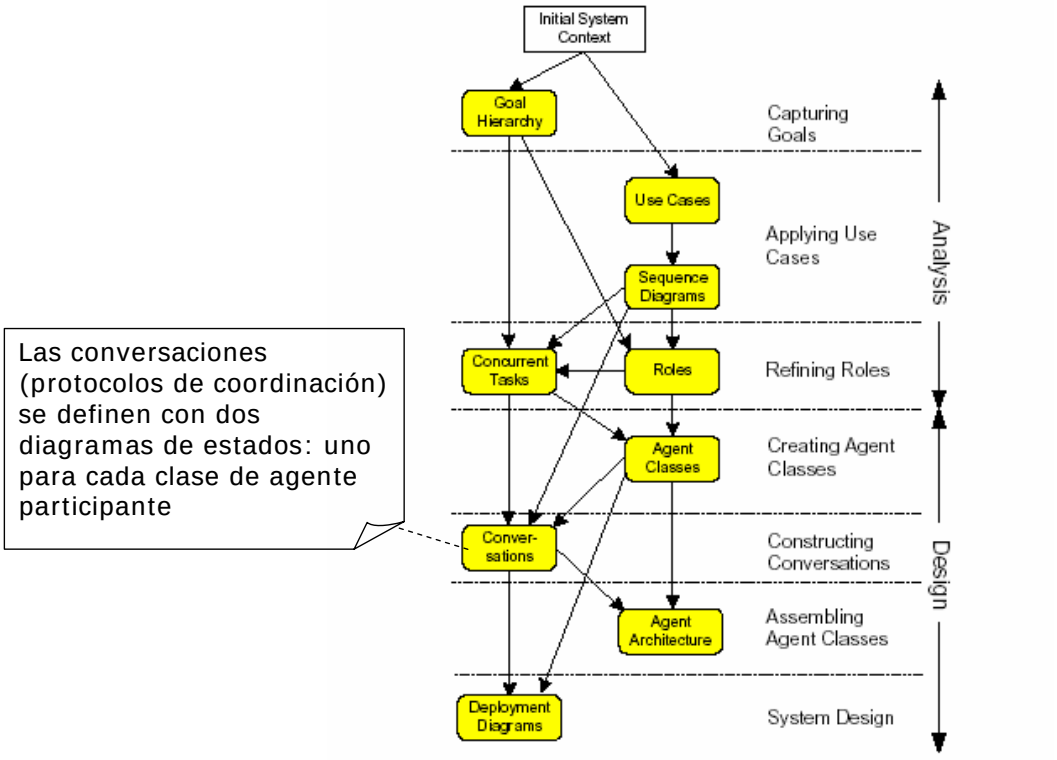
GAIA [Wooldridge, Jennings y Kinny, 2000]

- Extensiones
  - ROADMAP [Juan, Pearce y Sterling, 2002]
    - En el análisis añade:
      - Captura de requisitos utilizando casos de uso
      - Modelo de entorno
      - Modelo de conocimiento, derivado de los dos anteriores
    - Especificación de interacciones utilizando AUML
  - Gaia II [Zambonelli, Jennings y Wooldridge, 2003]
    - En el análisis añade:
      - Modelo del entorno: representación computacional abstracta del entorno del SMA, como una lista de recursos abstractos
      - Reglas organizacionales: restricciones para las actividades de los roles
    - En el diseño considera la definición de la estructura organizacional
      - Definida a partir de un conjunto de patrones organizacionales
    - Sigue sin considerar la implementación

MaSE [DeLoach et al. 01]

- Agentes como objetos con capacidad de coordinarse mediante conversaciones
- Agentes: clases cuyo comportamiento está definido por autómatas
- Soportado por la herramienta agentTool [DeLoach y Wood 01]
  - Generación automática de código
  - Notación UML
- Basado en el RUP, trata especialmente las actividades de análisis y diseño

MaSE [DeLoach et al. 01]

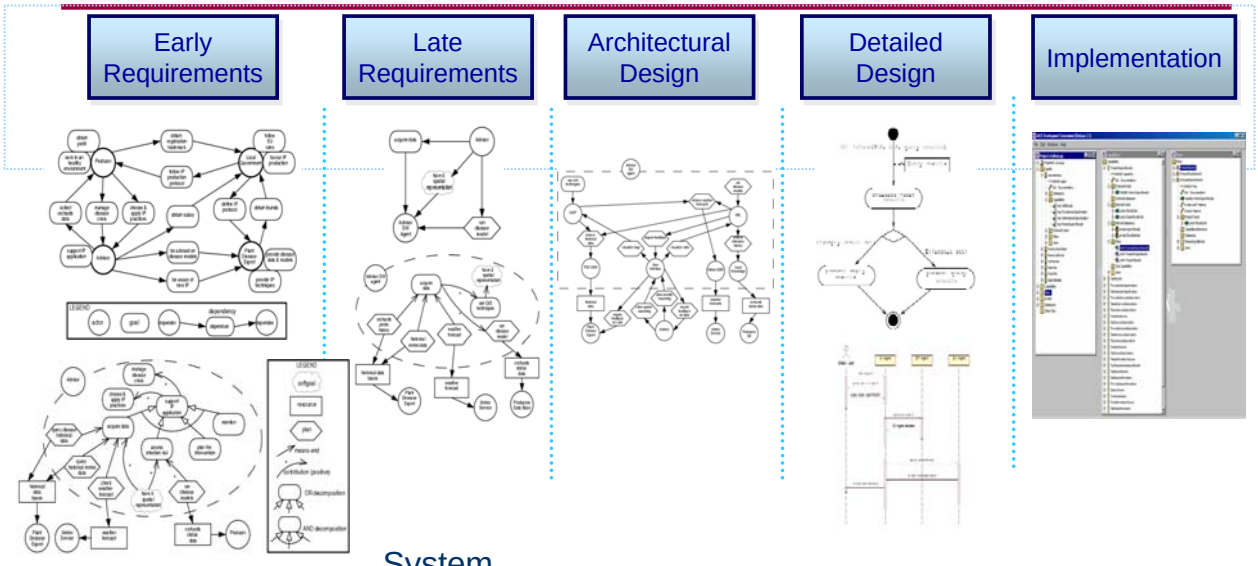


Tropos

- Integración de trabajo teórico previo: i*, KAOS
- Proceso de desarrollo
  - Análisis y Diseño basado en refinamiento de diagramas i* ampliados
  - Diseño detallado empleando técnicas adicionales
    - AUML para modelar protocolos
    - Diagramas de Planes
  - Implementación mediante plantillas de traducción a plataformas de agentes BDI
- Existen ejemplos de aplicación
- Hay herramientas de soporte

Tropos

De la presentación en AgentLink3, Ljubljana, 28 feb. 2005 por
D. Bertolini, A. Perini, A. Susi and H. Mouratidis. TROPOS meta-model
http://www.pa.icar.cnr.it/~cossentino/al3tf2/contributions.html



Actors in the organizational setting

System **Actor**

Sub-system **Actors**

Agents

Sw Agents

Requirement driven approach

Tropos

recurso

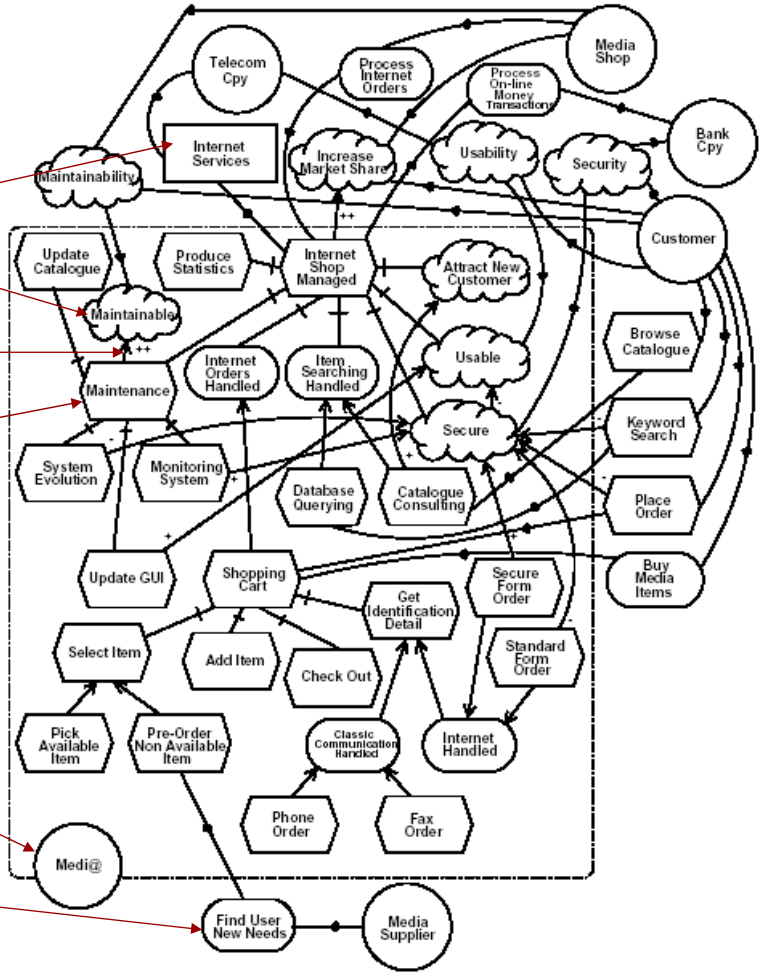
objetivo

contribuye

plan

actor

objetivo fuerte



Zeus [Nwana et al. 99]

- Zeus: entorno visual de desarrollo de SMA
  - http://more.btexact.com/projects/agents/zeus/
- Proporciona una plataforma de ejecución de agentes, prototipos de agentes, y componentes para su realización
  - Agentes de utilidad
    - Servidor de nombres
    - Agentes facilitadores
    - Agente visualizador
  - Herramienta de construcción de agentes
  - Librería de componentes de agentes
- El desarrollador configurará agentes genéricos mediante la definición de
  - Ontologías
  - Agentes
  - Tareas
  - Organización
  - Coordinación

Zeus [Nwana et al. 99]

- Etapas de desarrollo
  - Análisis del dominio
    - Basado en el modelado de roles, utilizando diagramas de clase UML y patrones
  - Diseño de los agentes
    - Identificación de ontologías, servicios, tareas y relaciones entre agentes
  - Realización de los agentes
    - Definición de los elementos identificados en el diseño y su implementación
    - Guiado por las herramientas gráficas de construcción de agentes de Zeus
  - Soporte en tiempo de ejecución
    - Depuración y optimización de código con herramientas de visualización y monitorización

Zeus [Nwana et al. 99]

- Etapa de realización de agentes
  - Creación de la ontología (Zeus Ontology Editor)
    - Conocimiento declarativo que representa los conceptos significativos dentro del dominio de la aplicación
  - Creación de agente (Zeus Agent Editor)
    - Configuración de un agente genérico de Zeus: definición de agente, descripción de tareas, organización del agente, coordinación del agente
  - Configuración de agentes de utilidad (Code Generation Editor)
    - Atributos de los agentes de utilidad (plataforma de agentes)
  - Configuración de agentes de tarea
    - Parámetros de ejecución de los agentes de tarea
  - Implementación de agentes
    - Utilizando la herramienta de generación de código

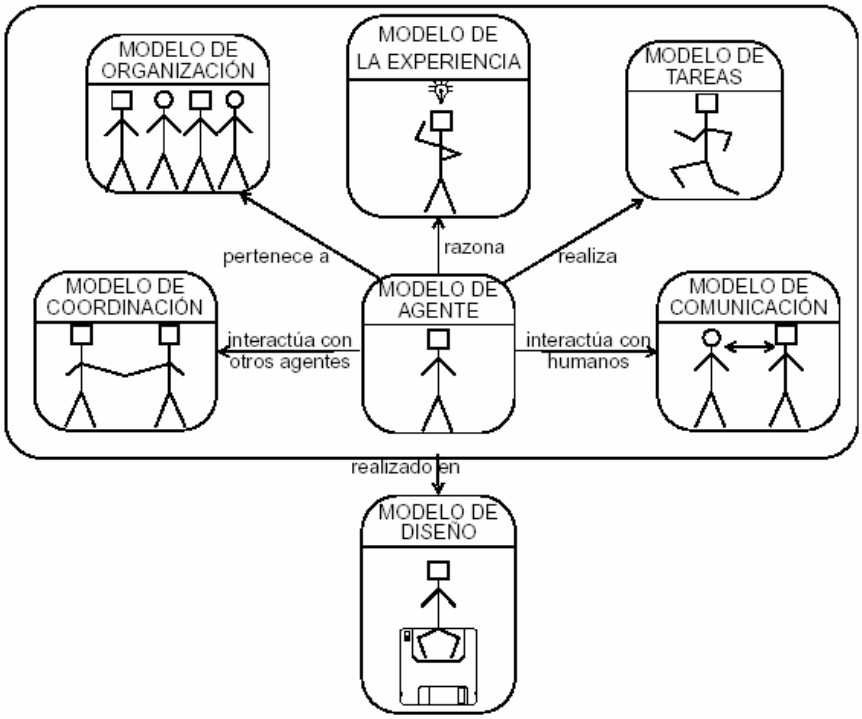
MAS-CommonKADS [Iglesias 98]

- Extiende la metodología CommonKADS [deHoog et al. 93] de desarrollo de sistemas expertos con:
  - Técnicas de orientación a objetos (OMT y OOSE)
  - Técnicas de ingeniería de protocolos: SDL y MSC
- Modelo de ciclo de vida en espiral dirigido por riesgos
  - Y modelo en cascada con reutilización para proyectos pequeños
- El desarrollo de un SMA consiste en rellenar un conjunto de plantillas de un número de modelos interrelacionados
  - Asociada a cada plantilla hay un estado que caracteriza los hitos en el desarrollo de cada modelo
    - Para cada variable de estado se pueden asociar varios valores: vacío, identificado, descrito y validado
    - Ejemplo de estados hito: identificación inicial de los agentes, descripción de objetivos y servicios, validación de relaciones entre un modelo y los demás

MAS-CommonKADS [Iglesias 98]

- Fase de conceptuación
  - Definición de casos de uso
- Especificación del sistema (Análisis)
  - Modelo de agente
    - Un agente es cualquier actor: humano, agente software, sistemas software (e.g. BD)
    - Capacidades de razonamiento, habilidades, servicios, sensores, efectores, grupos de agentes a los que pertenece y clase de agente
  - Modelo de tareas
    - Qué tareas pueden realizar los agentes, cómo se estructuran, objetivos, ...
  - Modelo de experiencia (o modelo de conocimiento)
    - Conocimiento necesario por los agentes para alcanzar sus objetivos
  - Modelo de organización de la sociedad de agentes:
    - Organización de los agentes y su relación con el entorno
  - Modelo de comunicación con el usuario
    - Factores de la interacción humano-agente
  - Modelo de coordinación
    - Interacciones entre agentes software
- Modelo de diseño
    - Arquitectura y diseño del SMA como paso previo a su implementación

MAS-CommonKADS [Iglesias 98]



MESSAGE [Message 2000]

- Methodology for Engineering Systems of Software AGents) Proyecto Eurescom P907
- Extiende ingeniería de software OO con conceptos del área de agentes
  - Que se definen en 5 meta-modelos:
    - Agente
    - Organización
    - Dominio
    - Tareas-objetivos
    - Interacciones
  - Adopta el Proceso Unificado de Desarrollo de Software (también conocido como Rational Unified Process)

INGENIAS

Evolución de MESSAGE

Abordando con mayor profundidad los distintos aspectos que definen la metodología:

- Especificación y notación para los elementos que constituyen un SMA
  - Agentes, organización, roles, servicios, objetivos, tareas, entorno, ...
- Modelo de ciclo de desarrollo de un SMA
  - Etapas del ciclo de vida, actividades, productos, métodos, ...
- Herramientas de ayuda
  - INGENIAS Development Kit (IDK)
    - Modelado
    - Generación de documentación
    - Generación de código
    - Validación y verificación

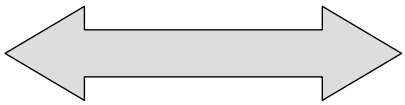
INGENIAS

- Planteamiento de INGENIAS
  - Agentes como paradigma de modelado
    - Conceptos de más alto nivel que en objetos y más cercanos al dominio
    - Se pueden considerar adaptaciones específicas a dominios de aplicación particulares
    - Los aspectos organizativos e intencionales reducen el salto de especificación de requisitos a implementación
  - Implementación sobre distintos tipos de plataforma
    - Un modelo de SMA se puede implementar sobre una plataforma de agentes o sobre un entorno de objetos tradicional
    - La metodología facilita y promueve el desarrollo de herramientas de generación de código que faciliten el paso del modelo (análisis y diseño) a la implementación
  - Contempla la evolución de la tecnología de agentes
    - Adaptabilidad a nuevos lenguajes y estándares (p.ej. AUML)
- Todo ello basado en la utilización y manipulación de meta-modelos

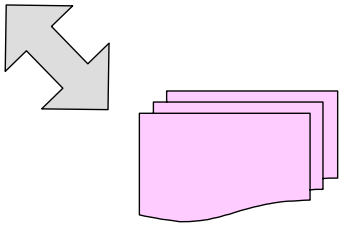
INGENIAS

- El modelo de desarrollo con INGENIAS
  - El desarrollador de SMA: realiza la aplicación

Modela con el editor del IDK



Simula y valida el modelo

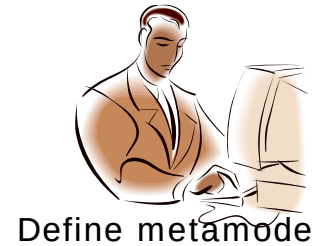


Genera código y valida el modelo

Implanta el SMA en la red

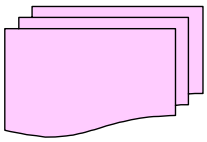
INGENIAS

- El modelo de desarrollo con INGENIAS
  - Ingeniero INGENIAS: prepara las herramientas



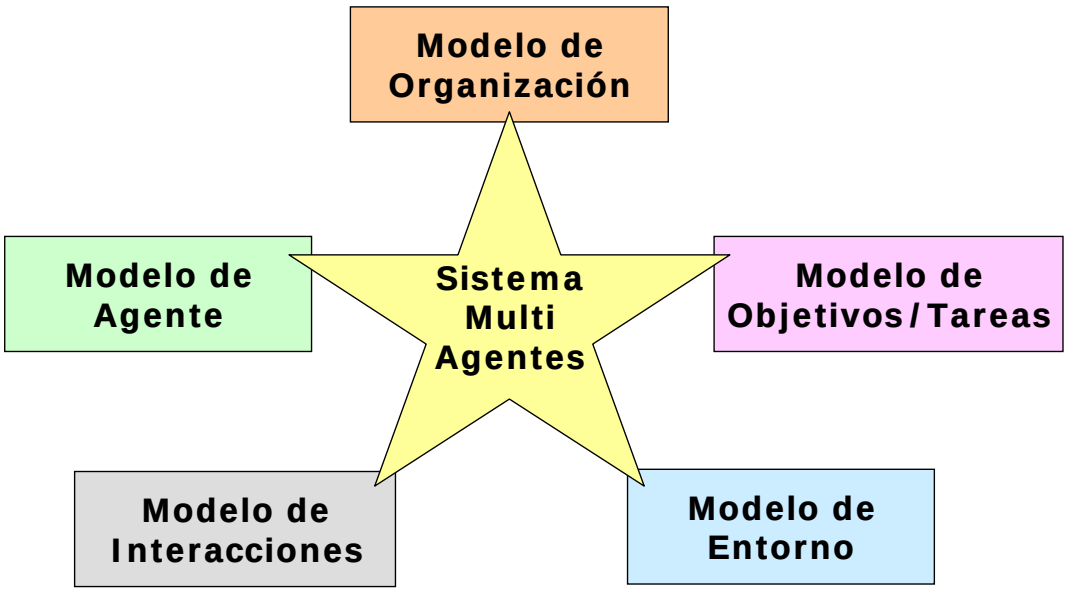
Define metamodelo del dominio de aplicación y puede personalizar el editor

Prepara generador de código para simulador



Crea generador de código para plataforma final

Modelado de un SMA con INGENIAS



Modelado de un SMA con INGENIAS

- Modelo de organización
  - Estructura del SMA, roles, relaciones de poder, workflows
- Modelo de agente
  - Los agentes realizan tareas o persiguen objetivos
  - Responsabilidades, control y estado mental del agente
- Modelo de objetivos y tareas
  - Identificación de objetivos generales y descomposición en objetivos más concretos que se pueden asignar a agentes
  - Similarmente con tareas
  - Objetivos: motivación ⇔ Tareas: actividad
- Modelo de interacción
  - Qué interacciones existen entre agentes/roles
- Modelo de entorno
  - Entidades y relaciones con el entorno del SMA

Integración de metodologías

- El paradigma de agente extiende al de objeto:
  - Aprovechar las metodologías OO
    - Más fácil de aceptar por los ingenieros software
    - Aprovecha herramientas y experiencia
  - Ciclo de desarrollo iterativo e incremental, basado en casos de uso
  - Extensiones:
    - Aspectos sociales (organización, interacciones, negociación)
    - Comportamiento (autonomía, estado mental, objetivos, tareas)
    - Concurrencia y distribución
- Modelado desde varios puntos de vista (Vowel Engineering, AAII, MAS-CommonKADS, MESSAGE)
  - Para poder gestionar la complejidad del SMA
  - Modelos: Entorno, dominio/ontología, roles, objetivos/tareas, interacciones/protocolos, organización, agente

Integración de metodologías

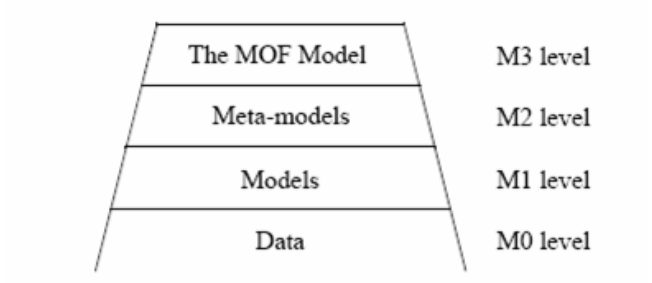
- Análisis
  - Casos de uso para capturar requisitos funcionales
  - Roles y servicios para agrupar las distintas funcionalidades asociadas a un agente o grupo de agentes
- Diseño
  - Independiente de la arquitectura (Gaia)
  - Basado en una arquitectura concreta (MaSE, AAII, Zeus)
  - Define el modelo computacional del agente => arquitectura del agente (MAS-CommonKADS)
- Relevancia de las herramientas (Zeus, MaSE, MESSAGE)

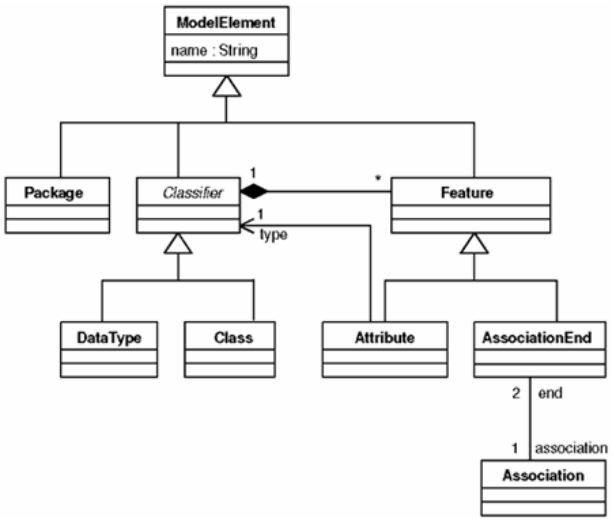
Integración de metodologías: AgentLink3

- Tanto en FIPA TC Methodologies como en AgentLink3 se está intentando definir un meta-modelo de SMA común a varias metodologías
  - ADELFE
  - GAIA
  - INGENIAS
  - PASSI
  - TROPOS
  - Los que se quieran aportar
- Los metamodelos permitirán
  - Clarificar conceptos
  - Definir un lenguaje común
  - Construir herramientas
- Primera propuesta conjunta:
  Bernon C., Cossentino M., Gleizes M-P., Turci P., and Zambonelli F., *A Study of some Multi-agent Meta-models*. In Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004. Revised Selected Papers. Editors: James Odell, Paolo Giorgini, Jörg P. Müller. LNCS Series, pp62-77, Vol. 3382 / 2005

Metamodelado

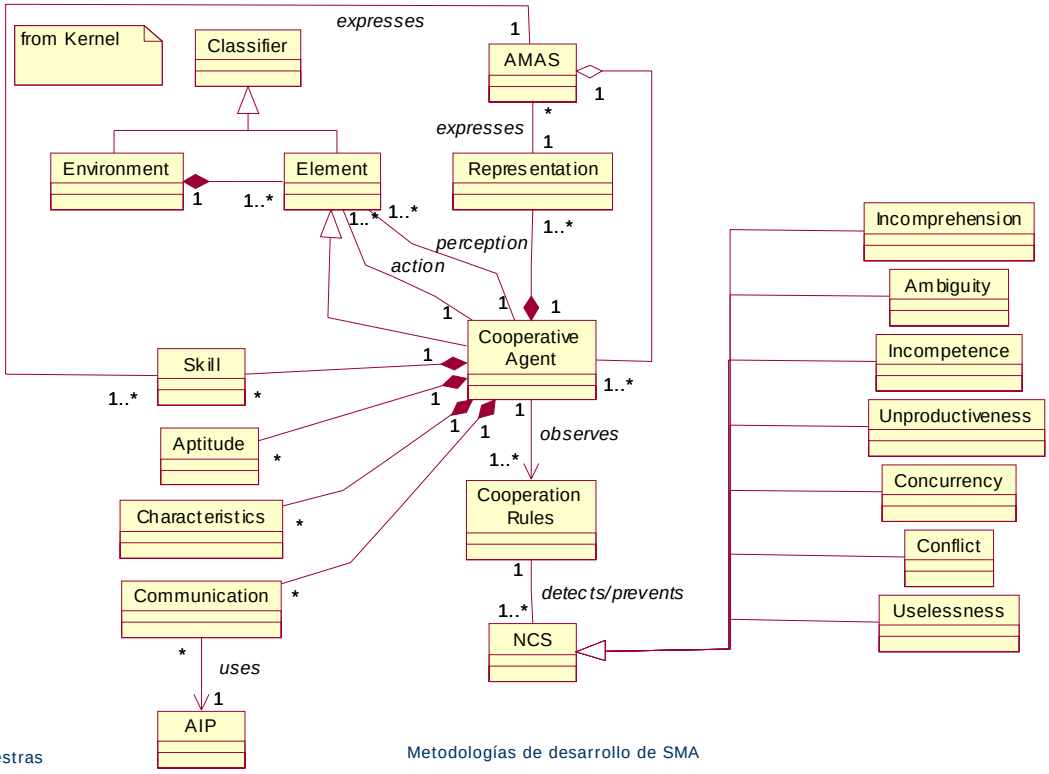
- OMG define la Meta Object Facility (MOF)
  - Lenguaje abstracto para definir lenguajes de modelado (por ejemplo, UML)





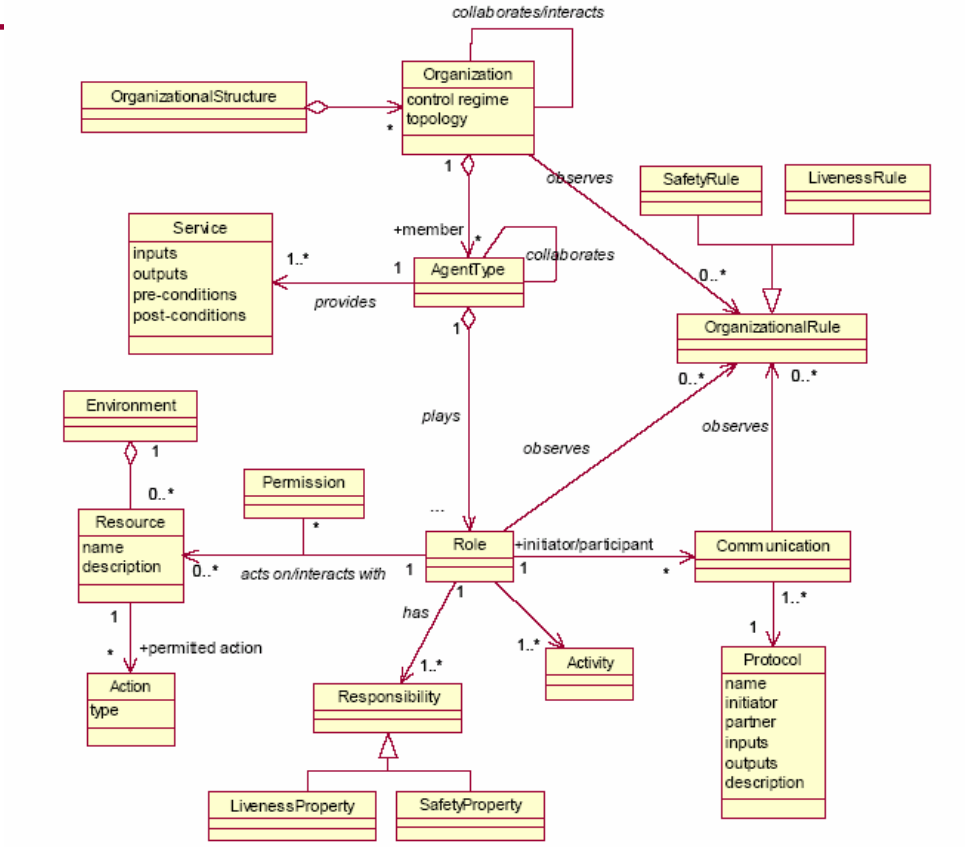
Integración de metodologías: AgentLink3

- ADELFE



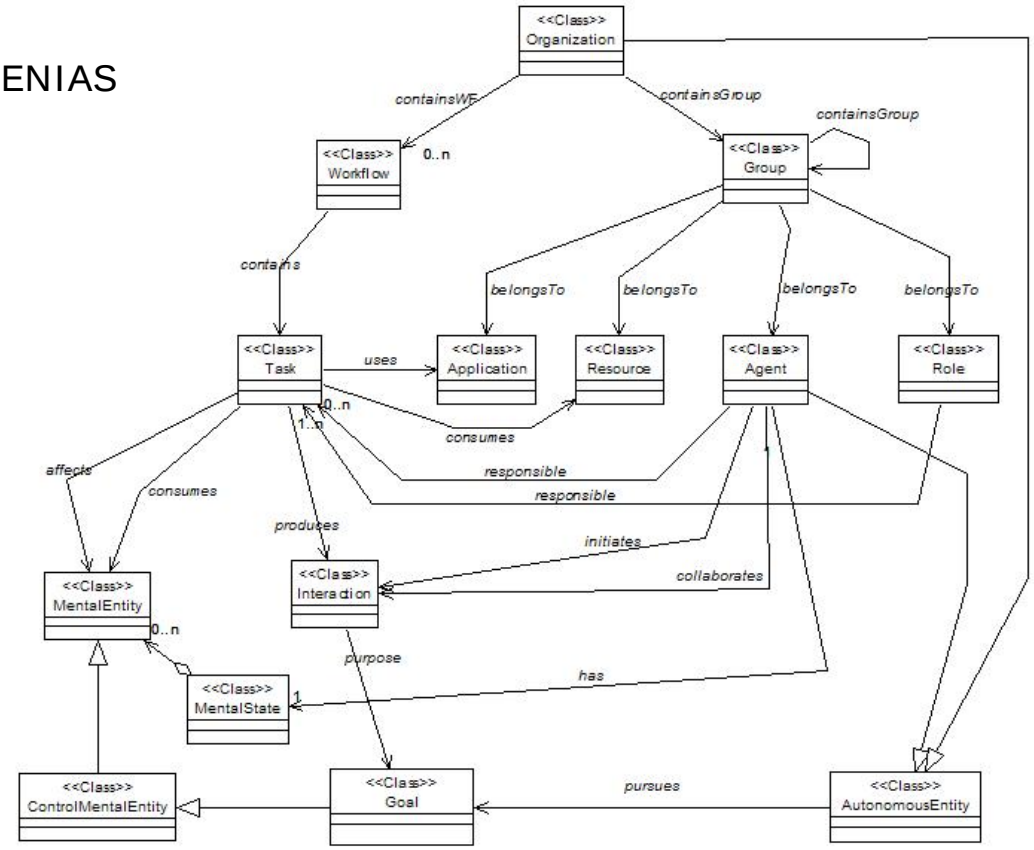
Integración de metodologías: AgentLink3

- Gaia



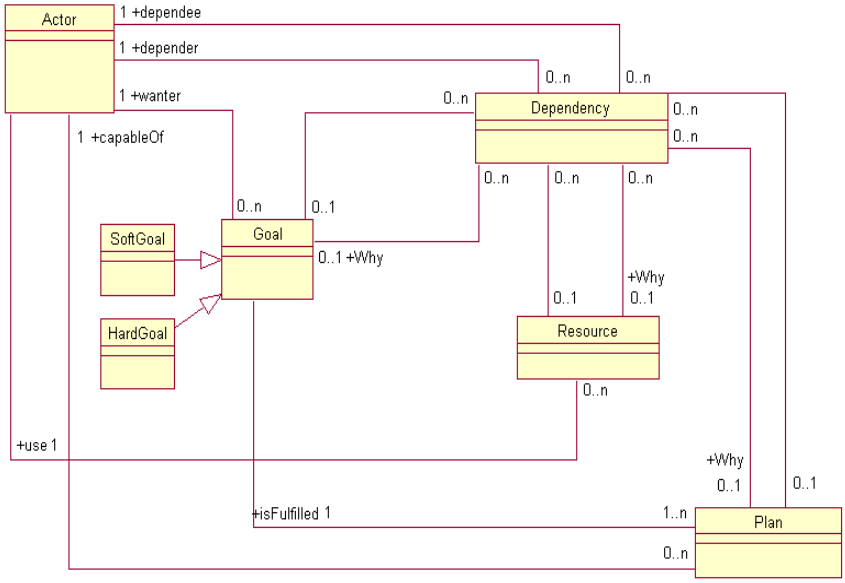
Integración de metodologías: AgentLink3

- INGENIAS



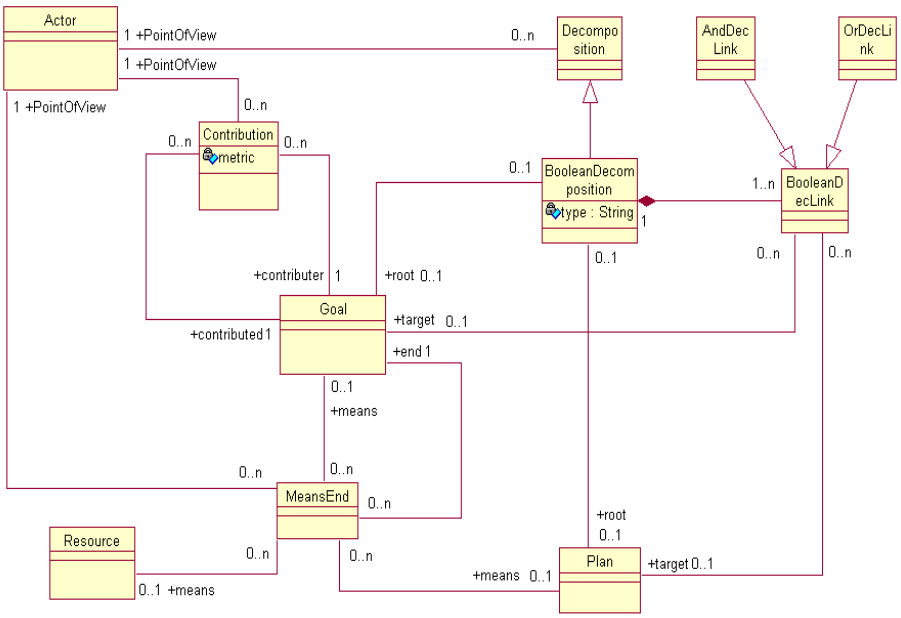
Metamodelo Tropos 1/2

- **Actor**: an entity that has strategic goals and intentionality;
- **Goal**: actors' strategic interests;
- **Resource**: a physical or an informational entity;
- **Plan**: a way of doing something;
- **Dependency**: depender → dependum → dependee.

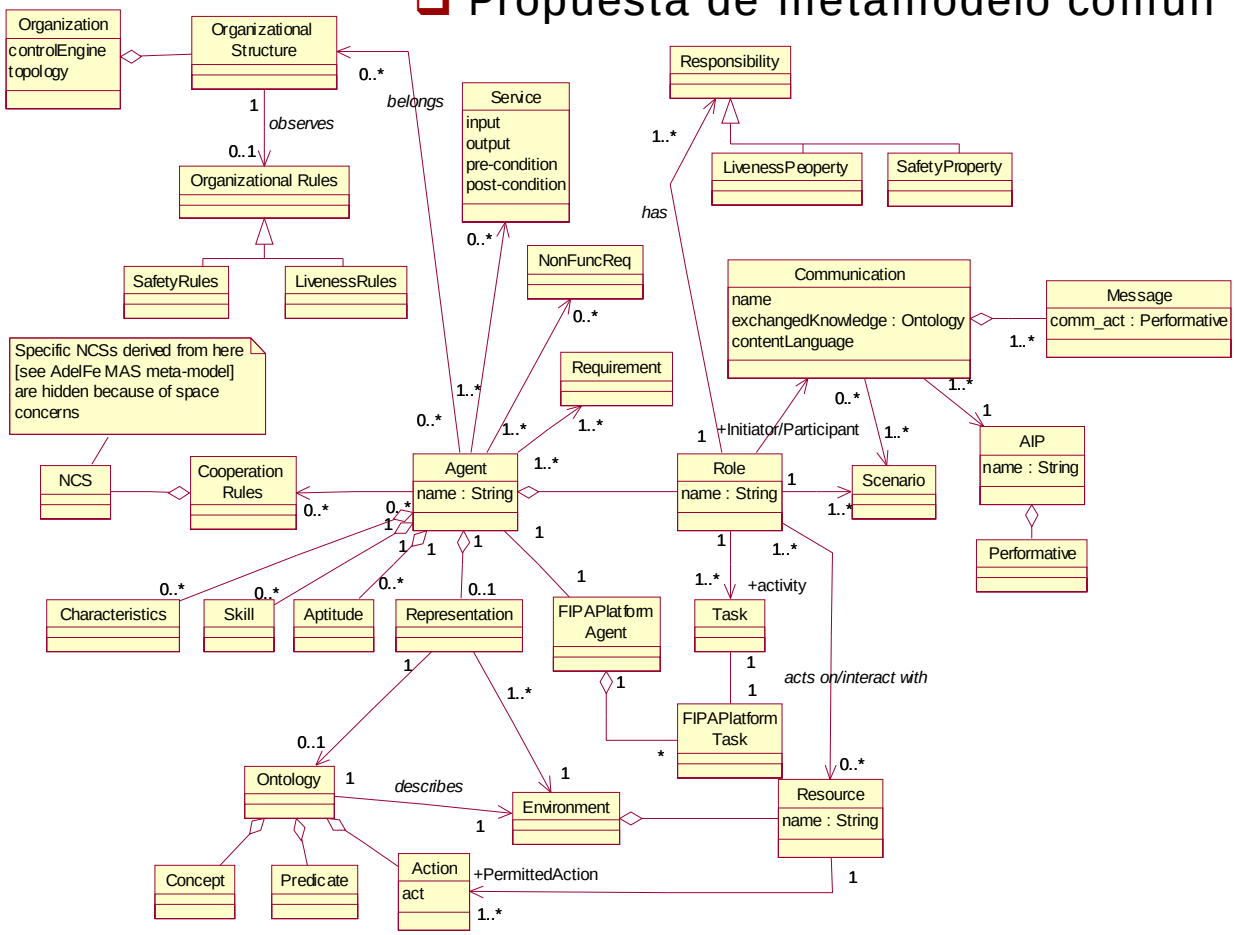


Metamodelo Tropos 2/2

- **AND/OR decomposition:** root(Goal) → sub(Goals)
- **Contribution:** towards the fulfillment of a goal
- **Means-end analysis**: a means to satisfy the goal.



Propuesta de metamodelo común



Entornos de desarrollo de agentes

- EIDE
  - ISLANDER: modelado de instituciones electrónicas
  - AMELI: plataforma de ejecución de instituciones electrónicas
  - ABUILDER: diseño de agentes que participan en instituciones electrónicas
  - http://e-institutor.iiia.csic.es/
- INGENIAS Development Kit (IDK)
  - Editor gráfico de modelos, generación automática de código y documentación, validación y verificación de propiedades
  - http://ingenias.sourceforge.net
- Otras
  - agentTool: metodología MaSE
    - http://www.agentbuilder.com/
  - MadKit: modelo de organización AGR (Agent/Group/Role)
    - http://www.madkit.org

Conclusiones

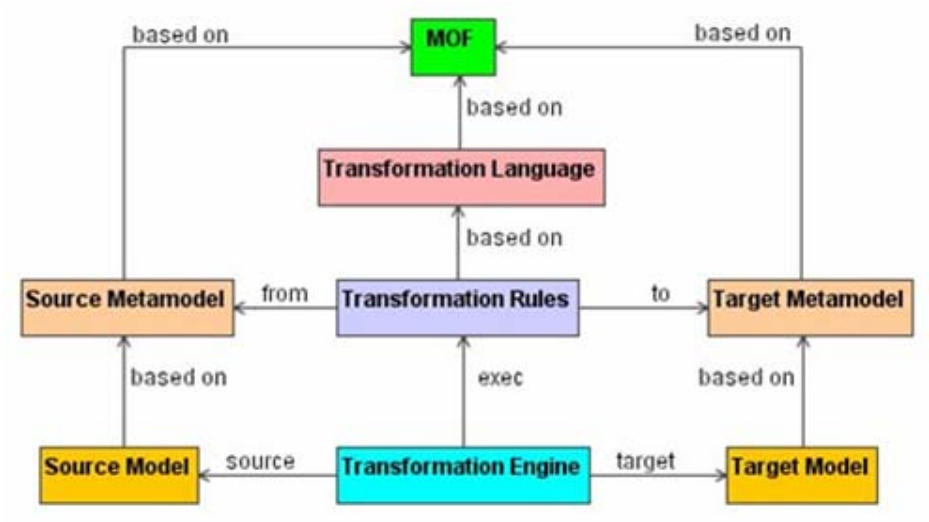
- El paradigma de agente extiende al de objeto:
  - Aprovechar las metodologías OO
    - Más fácil de aceptar por los ingenieros software
    - Aprovecha herramientas y experiencia
  - Ciclo de desarrollo iterativo e incremental, basado en casos de uso
  - Extensiones:
    - Aspectos sociales (organización, interacciones, negociación)
    - Comportamiento (autonomía, estado mental, objetivos, tareas)
    - Concurrencia y distribución
- Modelado desde varios puntos de vista (Vowel Engineering, AAII, MAS-CommonKADS, MESSAGE)
  - Para poder gestionar la complejidad del SMA
  - Modelos: Entorno, dominio/ontología, roles, objetivos/tareas, interacciones/protocolos, organización, agente

Conclusiones

- Análisis
  - Casos de uso para capturar requisitos funcionales
  - Roles y servicios para agrupar las distintas funcionalidades asociadas a un agente o grupo de agentes
- Diseño
  - Independiente de la arquitectura (Gaia)
  - Basado en una arquitectura concreta (MaSE, AAII, Zeus)
  - Define el modelo computacional del agente => arquitectura del agente (MAS-CommonKADS)
- Relevancia de las herramientas (Zeus, MaSE, MESSAGE, INGENIAS)
- Intentos de estandarización: FIPA Methodology TC y AgentLink AOSE TFG

Conclusiones

- Propuestas estilo MDA
  - Separar la lógica de negocio de las tecnologías de soporte
    - Desarrollo centrado en modelos (Platform Independent Models, PIM)
    - La plataforma de destino no es lo importante (Platform Specific Models,(PSM)
    - Definición de transformaciones PIM a PSM (QVT)



Bibliografía

- Bratman, M. E. (1987) Intentions, Plans, and Practical Reason. Harvard University Press.
- DeLoach, S. 2001. Analysis and Design using MaSE and agentTool.. Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conferece (MAICS).
- Ferber, J., 1999. Multi-Agent System: An Introduction to Distributed Artificial Intelligence. Addison Wesley Longman.
- Iglesias, C. (1998). Definición de una metodología para el desarrollo de sistemas multiagente, Tesis Doctoral, UPM.
- Kinny, D., Georgeff, M. y Rao, A. (1996). A methodology and modelling technique for systems of BDI agents. En Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, LNAI 1038, 56–71.
- Nwana, H. S. et al. (1999) ZEUS: A Toolkit for Building Distributed Multi-Agent Systems, Applied Artificial Intelligence Journal, vol. 1, no. 13, pp. 129-185.
- Russell, S. y Norvig, P. (1995) Artificial Intelligence: a modern approach. Prentice Hall.
- Weiss, G. Multiagent Systems. The MIT Press, 1999.
- Wooldridge, M., Jennings, N. R. y Kinny, D (2000). The Gaia Methodology for Agent-Oriented Analysis and Design, Journal of Autonomous Agents and Multi-Agent Systems, 3 (3), 285-312.
- Zambonelli, F., Jennings, N. y Wooldridge, M. (2003) The Gaia methodology for agent-oriented analysis and design. ACM TOSEM.
- Web sobre metodologías de agentes: http://ma.ei.uvigo.es/isoa/
- Gómez Sanz, J.J. y Pavón, J. *Methodologies for Developing Multi-Agent Systems*, Journal of Universal Computer Science (10) 4, 359-374
- Ana Mas, Agentes Software y Sistemas Multi-Agente: Conceptos, Arquitecturas y Aplicaciones. Pearson Educación, Prentice Hall, 2005.

Bibliografía