ESCUELA SUPERIOR DE INGENIEROS
Departamento de Organización Industrial y Gestión de Empresas

Máster de Organización Industrial y Gestión de Empresas

Aplicación de Sistemas Multiagentes en el Modelado de Organizaciones

Tesis del máster realizada por
Mª del Carmen Romero Ternero

Dirigida por
Dr. Pablo Cortés Achedad
Profesor Titular de Universidad

Sevilla, junio de 2008

Prólogo

Este documento ha sido elaborado por la Dra. María del Carmen Romero Ternero, profesora del Departamento de Tecnología Electrónica de la Universidad de Sevilla, como trabajo de tesis del máster en el Máster Oficial de Organización Industrial y Gestión de Empresas de la Universidad de Sevilla.

Este trabajo recoge parte de los conocimientos adquiridos durante el máster, así como los adquiridos durante la estancia postdoctoral realizada en la Electronic and Computer Science (ECS) High School de la Universidad de Southampton. Allí trabajó con el profesor Dr. Richard Crowder, que es senior lecturer en el Departamento de ECS de la Universidad de Southampton, y miembro del grupo de investigación denominado Intelligence, Agents, Multimedia Group.

Este trabajo se enmarca en el Proyecto HIPARSYS (High Performance And Robust Systems), que comenzó en enero de 2006 y concluye a finales de 2008. El ECS está colaborando en este proyecto con otras entidades para investigar sobre las prácticas actuales en el modelado de sistemas y procesos organizacionales en entornos de ingeniería e incluso en el desarrollo de métodos alternativos para llevarlos a cabo. La Universidad de Sheffield también colabora en este proyecto, y los principales patrocinadores son Jaguar y Rolls Royce.

Todos los sitios web usados en la sección de referencias han sido accedidos desde marzo de 2006 hasta junio de 2008 y han sido incluidas todas las referencias consultadas aunque no se hayan utilizado como referencias directas en el texto. Se ha hecho así porque toda la literatura referenciada ha sido fuente, directa o indirectamente, de la información utilizada en la investigación realizada, y con idea de que pueda servir como base documental en los temas tratados a lo largo de este trabajo.

Índice

1. Introducción: Proyecto HIPARSYS

El proyecto HIPARSYS (HIgh Performance And Robust SYStems) supone un reto en el campo del modelado organizacional, ya que sus objetivos son [37] [145]:

- ✓ Modelar y simular procesos en una organización.
- ✓ Unir la experiencia en prácticas organizacionales, modelado de agentes y psicología organizacional.
- ✓ Desarrollar y explotar las capacidades a largo plazo.
- ✓ Centrarse en las cuestiones humanas y organizacionales.
- ✓ Resolver problemas específicos de las empresas.

Este proyecto está financiado por el DTI (Department of Trade and Industry) y Rolls Royce, y algún trabajo relacionado ha sido financiado por el EPSRC (Engineering and Physical Sciences Research Council). Los principales socios implicados son Rolls Royce, Jaguar, la Universidad de Southampton y la Universidad de Sheffield.

Se trata de una investigación que es parte de un programa multidisciplinar y multiorganizacional a gran escala, cuyo tema común son los sistemas robustos y de alto rendimiento (high performance and robust systems – HIPARSYS). El programa incluye tres paquetes de trabajo diferentes. El modelado organizacional constituye el paquete 3 (WP3) en los que las dos universidades (Sheffield y Southampton) y las dos compañías (Rolls Royce y Jaguar) están implicadas.

Tanto Rolls Royce como Jaguar han planteado un problema actual de investigación. El caso de estudio de Jaguar se enfoca en un trabajo de ingeniería relacionado con la combustión de motores de pistón y el caso de estudio de Rolls Royce se centra en un trabajo de ingeniería relacionado con el módulo de turbina completa. En ambos casos, los procesos organizacionales –tales como la interacción entre miembros de un mismo equipo como entre equipos diferentes– implicados en dichos trabajos de ingeniería se modelarán a lo largo de este proyecto.

El trabajo de investigación inicial se centra en el caso de estudio de Jaguar y después se validarán y refinarán los modelos organizacionales desarrollados mediante la investigación en el caso de estudio de Rolls Royce.

Las tareas de investigación planificadas son [145]:

- ✓ Sheffield y Southampton investigarán las prácticas actuales asociadas a los problemas de estudio e incluso desarrollarán métodos alternativos.
- ✓ Sheffield identificará las reglas subyacentes en tales prácticas de trabajo.
- ✓ Estas reglas serán utilizadas para modelar las prácticas de trabajo actuales, con el soporte de la universidad de Southampton y las compañías.
- ✓ La complejidad de tales modelos se incrementará gradualmente, manteniendo su precisión predictiva, mediante la inclusión de más datos y reglas.
- ✓ Los modelos serán validados frente a las prácticas de trabajo actuales.
- ✓ Una vez validados, los modelos se utilizarán para modelar nuevas prácticas de trabajo, con el objetivo de asegurar que son óptimamente robustos y de alto rendimiento.

La innovación en este proyecto adquiere una doble dimensión. La primera es la integración y aplicación de una serie de tecnologías dispares, que actualmente están en fase de investigación, a un conjunto de problemas reales que las demandan. Esto dará como resultado el posterior desarrollo y validación de tales tecnologías. La segunda, y el aspecto más arriesgado, es extender el modelado y simulación del diseño (Design Simulation and Modelling, DSM) a sistemas organizacionales, explotando la sinergia entre la tecnología multiagente y la comprensión de la psicología de trabajo en la operación de individuos y organizaciones. La mayor innovación probablemente proceda de cuestiones como la variabilidad e incertidumbre en este nuevo dominio de modelar organizaciones al mismo tiempo que las conduce en el mundo más establecido de modelar la física de los productos. El proyecto en su totalidad adopta un enfoque socio-técnico, combinando experiencia en cuestiones técnicas y sociales.

Este estudio cubre la fase inicial del proyecto HIPARSYS y su principal objetivo es dar soporte a la primera tarea de investigación planificada. Sin embargo, en el último apartado se expone brevemente un primer modelo desarrollado como prototipo inicial.

2. Modelado y Simulación del Diseño

Hoy día no es suficiente simplemente con optimizar un diseño de sistema nominal. Por un lado, los sistemas están creciendo en complejidad y son no lineales en su comportamiento, de ahí que exista más susceptibilidad a la variación e incertidumbre. Efectivamente el proceso de optimización en sí mismo puede conducir a diseños dentro del área del espacio de soluciones donde el rendimiento es alto, pero también son sensibles a la variación. Por otro lado, el cambio fundamental emergente en muchas industrias (desde la venta de productos a la provisión de servicios) es migrar el riesgo de nuevos sistemas (desde el comprador del producto hasta el proveedor del servicio). Por tanto, la presión comercial está creciendo para desembocar en la necesidad paradójica de sistemas robustos, que son insensibles a la variación en sus entornos de fabricación y operación, y, además, tienen altos los niveles de rendimiento e innovación esenciales para la competitividad del mercado. El modelado y simulación del diseño (Design Simulation and Modelling, DSM) es clave para resolver esta paradoja.

Cuando un nuevo problema de diseño aparece, se utiliza el conocimiento de los diseñadores con experiencias en problemas previos ya resueltos. Es decir, el proceso empieza usando el conocimiento de los diseñadores de forma similar o relacionada con otro problema y se trata de aplicar, si es posible, realizando las modificaciones oportunas para resolver el nuevo problema. Como algunos autores afirman (Alder et al, 1989) [1], cuanto más experiencia posea un experto más rápida y fácilmente resuelve en nuevo problema. La información sobre cómo se resuelven los problemas se documenta y, en general, está disponible para otros diseñadores que quieran consultarlo, a veces a través de alguna herramienta específica (Shadbolt y Milton, 1999) [154]. El proceso seguido por un diseñador cuando quiere resolver un problema de diseño es la parte importante para este trabajo que es el contenido de la caja Proceso de Diseño en la Figura 1. Generalmente, un diseñador sigue una serie de pasos para resolver el problema de diseño. Tenemos que determinar cuáles son esos pasos con idea de extraer las reglas de operación en un proceso de diseño, que es uno de los objetivos del proyecto HIPARSYS.

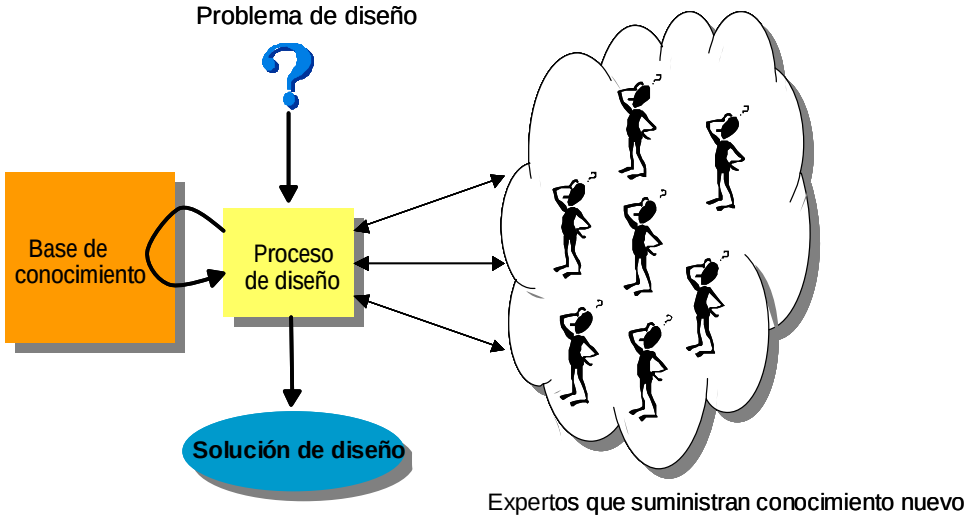

Figura 1. Proceso de diseño genérico (Romero et al., 2008) [147].

De acuerdo con un estudio realizado por ARC Advisory Group in 2003 [8], se estima que las industrias de procesos en la Unión Europea gasta aproximadamente 300 millones de euros al año en actividades de modelado y genera más de 1 billón de euros por año en valor añadido como resultado[1]. Estos beneficios se obtienen en términos de optimización del tamaño del equipo, optimización del rendimiento de la planta, reducción del gasto en ampliación y mejora de la calidad en la toma de decisiones. Además, hay enormes beneficios medioambientales y de seguridad que contribuyen al bien público y permite a los fabricantes mantener sus licencias para operar.

2.1. Modelado y Simulación de Diseño: Panorámica

Los ingenieros suelen trabajar con problemas en los que se pueden desarrollar muchas soluciones prácticas diferentes; buscan la mejor solución de entre todas las posibles alternativas.

Para determinar cuál es la mejor solución de entre un conjunto de alternativas, los ingenieros deben ser capaces de reconocer y desarrollar cada una de esas alternativas. Para realizar esta tarea de forma efectiva y eficiente, frecuentemente se sigue un procedimiento conocido como el proceso de diseño de ingeniería (Voland, 1999) [172]. La Figura 2 muestra este proceso dividido en seis tareas o fases principales.

Figura 2. El proceso de diseño de ingeniería, incluyendo la tarea de Reflexión (Voland, 1999) [172].

Durante la fase de reflexión, los ingenieros observan las lecciones aprendidas y el conocimiento adquirido como resultado del esfuerzo de diseño que se acaba de completar. Este periodo de reflexión formal puede ser útil para arrojar luz sobre aquellos aspectos de la experiencia que puedan ser utilizados en futuras tareas de diseño de una forma más efectiva. Esto es particularmente valioso si todos los miembros del equipo de diseño pueden compartir sus conclusiones sobre el proyecto una vez que éste ha finalizado. De ese modo, la base de conocimiento mostrada en la Figura 1, crece cuando el proyecto concluye.

Para obtener el mejor diseño hay muchas prácticas de ingeniería, entre las cuales destacan:

[1] En [53] Forsyth asume como razonable que haya un retorno de 4 a 10 veces la inversión.

- Diseño del ciclo de vida. Todo el ciclo de vida es relevante a la hora de diseñar un producto, desde la concepción hasta la fabricación y su uso final.
- Diseño para fabricación y ensamblaje. Los ingenieros de diseño deberían trabajar para garantizar que cualquier solución propuesta pueda ser fabricada adecuadamente.
- Diseño para calidad. Los principios de garantía de calidad se aplican a un diseño para asegurar bajas tasas de fallos junto con niveles de rendimiento altos.
- Ciclos de diseño más rápidos. Las nuevas tecnologías informáticas se aplican a la mejora tanto del diseño como del proceso de diseño en sí mismo.
- Ingeniería sin muros. Aprovechando las ventajas de Internet, el trabajo colaborativo distribuido es actualmente una de las principales herramientas para la producción de mejores diseños más rápidamente y con menores costes, compartiendo gastos y aunando sus recursos.
- Diseño para exportación. La tendencia es desarrollar estándares de productos para ampliar el mercado.

Cuando los diseñadores tienen que encontrar una solución para un problema específico, no debe buscar la generación y evaluación de cada posible solución al problema, ya que las limitaciones deben especificarse en cantidad de tiempo, esfuerzo y dinero que puede ser invertido en el proyecto. Con idea de estructurar el camino a seguir a la hora de elaborar una solución final de diseño, hay que considerar y comparar los aspectos del problema específico a tratar y los posibles estados de la solución:

– Eliminar caminos de soluciones imposibles, de acuerdo con los objetivos de diseño y las restricciones.
– Extraer la información más útil.
– Evaluar el estado de la solución final.

Volviendo a la Figura 2, en la fase de abstracción el ingeniero debe ser creativo, por lo que cuanto más experiencia tenga, mejores y más rápidos resultados puede lograr (Adler, 1999) [1]. El primer paso en la abstracción es dividir el problema, en la medida de lo posible, en distintas partes funcionales, subproblemas o unidades significativas. El diseñador identifica conjunto de objetivos que deben ser satisfechos para cualquier solución viable al problema. Posteriormente, tratará de clasificar estos aspectos funcionales del problema en categorías más generales de acuerdo con sus características distintivas. En la realización de esta tarea, Voland (1999) [172] basa su esquema de clasificación en variaciones de:

- el objetivo general que tiene que alcanzar la solución,
- los principios o enfoques que pueden ser utilizados para alcanzar este propósito, tal como una familia de invenciones que se han utilizado para resolver problemas similares,
- el contexto o los entornos de operación en los que la solución podría utilizarse, o
- las subtareas específicas que tienen que ser realizadas (ya sea secuencial o concurrentemente) para alcanzar el objetivo en su totalidad.

El modelado es parte del proceso de abstracción, ya que los ingenieros usan modelos para desarrollar y evaluar sus ideas. Los modelos permiten a los ingenieros organizar los datos, estructurar sus ideas, describir relaciones y analizar los diseños propuestos.

La simulación es una experimentación dirigido a objetivos con modelos dinámicos, por ejemplo con comportamiento dependiente del tiempo (Ören, 2002) [137]. Como tal, la simulación añade otra dimensión a la experimentación:

(1) En simulación, es posible realizar experimentos incluso cuando el sistema real no existe (como ocurre en la mayoría de los problemas de diseño) o no está accesible para la experimentación.

(2) En simulación, es posible explorar efectos de una gran variedad de condiciones experimentales que no son prácticas o viables para realizar experimentos en el mundo real, por ejemplo, una simulación de un terremoto o de un accidente aéreo en un simulador de vuelo. Estas dos posibilidades son definitivamente superiores respecto la experimentación en sistemas reales.

(3) Desde una perspectiva más amplia, la simulación es una técnica de generación de conocimiento experimental; de ahí que pueda ser combinada con otras técnicas de generación de conocimiento no experimentales, tales como optimización, inferencia estadística, razonamiento inductivo y deductivo, y razonamiento difuso (Ören, 1990) [131]. La simulación también puede ser combinada con otras técnicas de procesamiento y especialmente con técnicas de inteligencia artificial (Ören, 1994; 1995) [132][133] así como con agentes software (Ören, 2001) [135].

De acuerdo con la descripción de Ören en (Ören, 2002) [137], las áreas de desarrollo prometedoras en el campo del modelado y la simulación se muestran en la Tabla 1. También describe un paradigma de ingeniería de sistemas para la simulación y modelado en estudio detallado sobre Body of Knowledge of Modelling and Simulation (M&SBOK) (Ören, 2005) [138] y está recogido en la Tabla 2.

Tabla 1. Áreas de desarrollo prometedoras en el campo del modelado y la simulación según (Ören, 2002) [137]

Categorías		Desafíos (o desarrollos deseables)	
1	Ciencia, metodología, y tecnología de modelado y simulación		Metodologías y tecnologías para:
		1.1	Multi-modelos
		1.2	Modelado multicriterio, multiestado, multiperspectiva, multiresolución y multiparadigma
		1.3	Modelos de estructura variable
		1.4	Simulación de formalismo mixto
		1.5	Multisimulación
		1.6	Simulación concurrente
		1.7	Procesado de objetivos en modelado y simulación
		1.8	Automatización del diseño de experimentos
		1.9	Simulación dirigida a agente
		1.10	Simulación de agente holónico (para sistema cooperativos)
		1.11	Lenguajes de especificación y entornos para interoperabilidad
2	Formalidad, fiabilidad, calidad y eficiencia en modelado y simulación	2.1	Fiabilidad integrada antes de la validación y verificación
		2.2	Documentación apropiada de estudios de simulación (incluyendo suposiciones)
		2.3	Reutilización de librerías
		2.4	Doma y monitorización de agentes software –tener comportamiento casi autónomo auto-inhibido frente al comportamiento autónomo completo– para que los agentes se comporten de forma civilizada
3	Áreas de aplicación		Uso de la simulación para mejorar (entrenar):
		3.1	Cooperación (negocios, defensa, …)
		3.2	Gestión de conflictos (evitación, resolución, …)
		3.3	Soporte para la paz / aseguramiento de la paz
		3.4	Comportamiento humano
		3.5	Sistemas sociales
		3.6	Uso de la simulación para entrenar sistemas con capacidades de aprendizaje
4	Consolidación y diseminación del conocimiento del modelado y la simulación (como referencia / educación)	4.1	Sistematización del cuerpo de conocimiento en M&S
		4.2	Desarrollo de currículum para estudios de postgrado en M&S
		4.3	Diccionario de modelado y simulación
		4.4	Libro de texto electrónico
		4.5	Diseminación del conocimiento del modelado y la simulación (un centro de referencia electrónico)
5	Profesionalidad	5.1	Código de éticas profesionales para profesionales de la simulación
		5.2	Certificación

Tabla 2. M&SBOK – Paradigma de ingeniería de sistemas para M&S (Ören, 2005) [138].

Estado del problema	Garantiza consenso con el cliente en el ciclo de vida del proyecto, necesidades, objetivos y varias métricas de rendimiento: • Adecuación al propósito, utilidad, usabilidad, efectividad en costo, durabilidad, eficiencia, manteniabilidad a nivel de especificación y a nivel de código • Ámbito de usabilidad o aplicabilidad • Documentar cada nivel: estudio, sistema, asunciones (explícitas o implícitas) … (estándares de documentación)
Investigar las alternativas	
Modelar el sistema	
Integrar	Sistemas de sistemas Federaciones de federaciones
Simular el sistema	
Lanzar el sistema	
Evaluar el funcionamiento	• Considerar el éxito de S&M a partir del cumplimiento del objetivo original del sistema y no sólo desde un punto vista limitado como la eficiencia • Para aplicaciones militares • Análogamente, usar una evaluación del funcionamiento dirigido a objetivos, en todas las áreas de aplicación
Re-evaluar	

Existe actualmente un amplio espectro de herramientas de modelado y algunas de ellas están descritas en (Forsyth, 2004) [53] como se muestra en la Tabla 3.

Tabla 3. Ejemplos de herramientas de modelado (Forsyth, 2004) [53].

Tipo de herramienta	Ejemplo de aplicación
Hoja de cálculo	Modelos simples de equilibrado de masas Modelos financieros simples
Paquete de modelado estocástico (por ejemplo, Witness)	Simulación (no optimización) de sistemas interactivos de colas en manufactura y servicios
Paquete de modelado de procesos (p.e. Aspen Plus de Aspen Technologies)	Cálculos simples de hoja de flujos Operación de una unidad (p.e. diseño de una columna de destilación)
Paquetes de modelado simple de propósito general (p.e. MatLab/Simulink de The Math works)	Diseño de bucle de control Modelo dinámico simple
FORTRAN y otros lenguajes informáticos	Modelos sofisticados de operación de unidades Optimización
Paquetes de modelado avanzado (p.e. gPROMS de PSE)	Modelos sofisticados de operación de unidades Reconciliación de plantas Optimización

2.2. Tecnologías de modelado y simulación y conocimiento de diseño

La sinergia de la Inteligencia Artificial (IA) y la Simulación y Modelado (S&M) está madurando en simulaciones dirigidas por IA y basadas en agentes (Ören, 2005) [138]. En este último caso, la sinergia de agentes y simulación facilita:

1. La simulación para agentes, por ejemplo, simulación de agente o simulación de sistemas que representan o implican componentes modelados mediante agentes software; así como
2. Agentes para la simulación, por ejemplo, el uso de agentes para la simulación que ofrece dos grupos de posibilidades:
   a. Simulación soportada por agente, para las funciones de interfaz front-end y back-end, para procesar simbólicamente elementos para un sistema S&M, por ejemplo, para garantizar la calidad incorporada; o para proporcionar habilidades cognitivas avanzadas para algunos componentes, tales como aprendizaje, comprensión, etc.; y
   b. Simulación basada en agente donde los agentes pueden utilizarse para generar modelos de comportamiento. En simulación dirigida a IA, lo análogo es la simulación cualitativa y la simulación basada en conocimiento.

Esta evolución en S&M proporciona capacidades para la simulación de fenómenos más complejos incluyendo la simulación de personalidad humana así como simulaciones sociales y simulaciones de conflictos. Nuestro estudio trata de utilizar teoría social para aplicarla a las interacciones entre los diseñadores, de manera que pueda ayudarnos a modelar el proceso de diseño.

En los primeros tiempos, cuando se usaban las computadores analógicas e híbridas e incluso con el uso de las computadoras digitales, ya era utilizado el término “simulación. Sólo unos pocos hacían alusión al modelado y simulación (Wikipedia.org, 2006) [175] (Zeigler, 1979) [188]. Posteriormente, para hacer énfasis en el proceso de modelado y las actividades y entornos asociados, se ha venido utilizando el término “modelado y simulación” (S&M abreviadamente). Actualmente, se está adoptando un cambio encomiable de paradigma para cubrir todo los aspectos de los estudios de simulación. Esto es, concebir S&M –dentro de una perspectiva más amplia– como la Ingeniería de Sistemas de Simulación (Simulation Systems Engineering, SSE).

En (Ören, 2000) [134] se pueden ver cuáles son los diferentes usos de la S&M (Tabla 4) donde podemos observar el uso de la S&M para la comprensión. Incluye test de hipótesis sobre la estructura y funcionamiento de sistemas complejos, especialmente en ciencias naturales, ciencias sociales y el modelado del comportamiento humano.

De acuerdo con (Ören, 2000) [134], un sistema A puede comprender a una entidad B si se cumplen tres condiciones:

1. A tiene acceso a C, un meta-modelo de las entidades B (C es el conocimiento de A respecto a Bs).
2. A puede percibir y analizar B para generar D (D es una percepción que A tiene de B).
3. A puede hacer corresponder las relaciones entre C y D.

Tabla 4. Una taxonomía del entendimiento (*understanding*) (Ören, 2000) [134].

Criterio relacionado con las características de		Tipos de entendimiento (understanding)
el producto del proceso de entendimiento	Dominio	– Entendimiento interno – Entendimiento externo
	Naturaleza	– Entendimiento léxico – Entendimiento sintáctico – Entendimiento morfológico (entender la estructura) – Entendimiento semántico (entender el significado) – Entendimiento pragmático (entender la intención)
	Ámbito	– Entendimiento focalizado – Entendimiento extendido (entender varias o todas las características) – Entendimiento multidisciplinar
	Granularidad (nivel de detalle)	– Entendimiento basto – Entendimiento en profundidad (entendimiento detallado)
	Fiabilidad	– Entendimiento fiable ▪ entendimiento válido, ▪ entendimiento verificado – Entendimiento no fiable ▪ entendimiento no válido, ▪ entendimiento no verificado
el proceso de entendimiento	Dirección	– Entendimiento top-down – Entendimiento bottom-up
	Franqueza	– Aprensión (entendimiento directo) – Comprensión (entendimiento indirecto, entendimiento mediado, entendimiento lógico)
	Acumulación de conocimiento	– Entendimiento re-inicializado – Entendimiento acumulativo
el meta-modelo usado	Fijo	– Entendimiento de visión única ▪ entendimiento dogmático
	Evolutivo	– Entendimiento de aprendizaje
	Reemplazable	– Entendimiento multivisión (entendimiento cambiante)
el sistema de entendimiento	Iniciativa del sistema de entendimiento	– Entendimiento autónomo – Entendimiento delegado ▪ entendimiento remoto
	Número de sistemas de entendimiento	– Entendimiento individual – Entendimiento grupal ▪ entendimiento distribuido
	Compartición de conocimiento del sistema de entendimiento	– Entendimiento repetitivo – Entendimiento cooperativo
	Mecanismos para diseminar los resultados del proceso de entendimiento	– Entendimiento por comando, bajo demanda de suscriptores, por difusión, mediante pizarra, por herencia

Por tanto, un sistema que comprende necesita tener tres elementos básicos: un meta-modelo de las entidades que deben ser comprendidas, un elemento de percepción y un analizador y un comparador para hacer corresponder una percepción de una entidad con el meta-modelo adecuado (Figura 3).

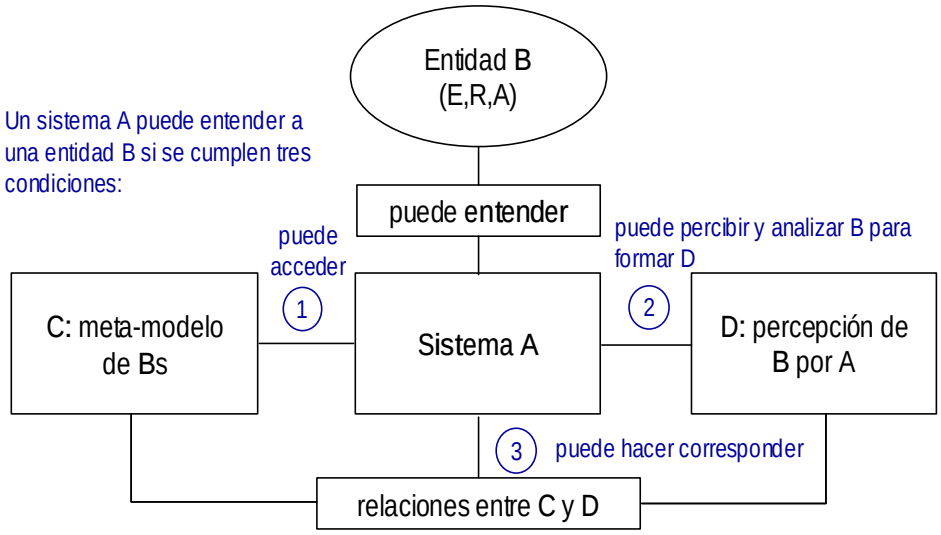

Figura 3. Elementos de un sistema comprensivo (Ören, 2000) [134].

La capacidad de un sistema A para comprender a una entidad B depende de las restricciones en las tres condiciones (Ören, 2000) [134]; por ejemplo, (1) de la existencia de un meta-modelo y de si está accesible, (2) de la percepción y el análisis de la entidad y (3) de las capacidades de mapeo de su comparador. Por tanto, las características de estas condiciones también pueden ser interpretadas como factores que afectan al rendimiento de los sistemas comprensivos.

De acuerdo con la capacidad de mapeo es necesario tener en cuenta las siguientes cuestiones:

- Para comprender a una entidad B, el sistema A necesita realizar un mapeo entre un meta-modelo C de Bs y D, una percepción de B o el resultado del análisis de B.
- Las características de las relaciones (por ejemplo, detectable, encontrada o no existen relaciones) afectan al límite de la comprensión.
- Los comentarios de implementación similares a las capacidades de percepción y análisis también influyen en las capacidades de mapeo.
- La velocidad de mapeo afecta a la velocidad de aprendizaje.

Por tanto, para comprender cómo trabajan los diseñadores durante el proceso de diseño, podríamos considerar todo lo propuesto por Ören e incluso es importante cuánto conocimiento podemos extraer de los diseñadores y de la base de conocimiento de las empresas que están implicadas en el proyecto, con idea de alcanzar una comprensión detallada y, en consecuencia, un buen modelo.

2.2.1. Metodologías de diseño

La arquitectura de compartición de información en las empresas ha estado dirigida principalmente hacia dos caminos (Dong et al., 1995) [38]: basada en agente o basada en tema (*issue*).

En el modelo basado en agente, son agentes inteligentes los que facilitan la coordinación entre diseñadores, enviando, aceptando y encontrando información relevante para las peticiones de los diseñadores.

El método IBIS (Issue Based Information System) basado en tema trata a los diseñadores como un proceso de negociación entre diferentes grupos de trabajo. Utilizando hilos de conversación (*posting*) diferentes para cada tema, los grupos de trabajo pueden navegar y discutir decisiones de diseño y su justificación. Una extensión del IBIS de Ullman (1995) [171] añade datos sobre restricciones de diseño, funciones y la estructura del producto y sus componentes.

En la Tabla 5 se muestran varios ejemplos del sistema racional de diseños de prototipos.

2.2.1.1. Metodologías basadas en IBIS

La representación IBIS de la toma de decisiones se ha mostrado como un buen modelo para la resolución de problemas de diseño (Blessing, 1994; Nagy, 1990; Nagy y Ullman, 1992) [14] [113] [114]. IBIS permite la representación de la naturaleza interdependiente de temas, alternativas y argumentos independientemente del tipo de datos, su completitud o su consistencia. En un proyecto de desarrollo de software de la NCR Corporation, los investigadores utilizaron herramientas basadas en IBIS para representar la organización de temas, alternativas y decisiones en un computador (Yakemovic y Conklin, 1989; 1990) [185] [186]. El proyecto en su totalidad dio lugar a 2260 temas y es el estudio más grande realizado de este tipo. Algunos resultados del estudio fueron:

- ✓ IBIS proporciona una memoria compartida por todo el equipo de diseño. El histórico de las decisiones tomadas y registradas en las herramientas resultaba fácil de revisar y utilizar por los miembros del equipo de diseño y de gestión.

- ✓ IBIS ayudó al equipo a detectar cuestiones de diseño que no se hubieran detectado sin estas herramientas. Se estimó que el ahorro era de 3 a 6 veces mayor que el coste de usar las herramientas.

- ✓ IBIS ayudó al equipo a comprender más rápidamente el problema que se trataba de resolver.

- ✓ IBIS ayudó a estructurar la información (temas, posiciones, y argumentos) y a establecer y centrar una agenda de trabajo. Las reuniones del equipo parecían más productivas.

Tabla 5. Sistemas racionales de diseño de prototipos (Regli et al, 2000) [143].

Acrónimo del sistema	Representación del conocimiento	Captura del conocimiento	Recuperación del conocimiento	Enfoque	Dominio del diseño	Año
IBIS [90]	Basado en caso	IU	Navegación	OP	Genérico	1970
PHI [108]	IBIS extendido	IU	Navegación	OP	Genérico	1987
QOC [101]	Análisis del espacio de diseño	IU	Navegación	OP	Genérico	1990
DRL [92]	Representando elementos de toma de decisión	IU	Navegación	OP	Genérico	1991
CRACK [51]	N/A	Auto	Disparo	OC	Cocina	1989
VIEWPOINTS [50]	IBIS	N/A	Navegación	OC	Cocina	1989
JANUS [49]	PHI	Auto	Híbrido	OC	Cocina	1989
IBIS-style browser [98]	IBIS	Auto	Navegación	OP	Genérico	1991
COMET [104]	LOOM	IU	Navegación	OC	Software rastreador basado en sensor	1992
ADD [58]	Argumentación & Basado en modelo	IU	Disparo	OC	HVAC	1992
REMAP [141]	IBIS	IU	Solicitud	OP	Genérico	1992
REMAP/MM [142]	IBIS	Auto	Solicitud	OP	Genérico	1995
ADD+ [57]	Estructura retórica	IU	Solicitud	OP	HVAC	1997
HOS [156]	PHI	Auto	Disparo	OP	Genérico	1997
PHIDIAS [156]	PHI	Auto	Disparo	OC OP	2D, 3D	1997
KBDS-IBIS [9][86]	IBIS	IU	Solicitud & Navegación	OC OP	Planta química	1997
DRIVE [33]	PDN	IU	Solicitud	OC	Construcción	1997
DRARS [34]	QOC	N/A	N/A	OC	Construcción	1995
KRITIK [20] [124]	SBF	IU	Solicitud	OC	Mecánica	1993
IDIS [23]	IBIS	IU	Navegación	OC	Planta química	1998
RCF [112]	N/A	Auto	N/A	OP	N/A	1999

Método de Captura: Intervención del usuario (IU) o Automático (Auto)
Método de Representación: Orientado a característica (OC) u Orientado a proceso (OP)
Método de recuperación: Navegación, Solicitud, Disparo o Híbrido.

- ✓ IBIS daba soporte de comunicación con otros niveles de la organización. Los que no pertenecían al equipo de diseño encontraban fácil discernir de qué se estaba discutiendo, y no sólo cuál había sido la decisión final.

El modelo IBIS permite la representación de información incompleta y dinámica sobre temas independientes asociados al producto o al proceso. La información se expresa informalmente, de modo que el espacio de diseño puede incluir datos cuantitativos o cualitativos, así como datos determinísticos o distribuidos como soporte al producto o al proceso. No hay una estructura formal para dar soporte a creencias o preferencias, sin embargo, se pueden expresar informalmente como parte de un argumento. Aunque IBIS puede modelar información sobre toma de decisiones complejas, no ofrece soporte automatizado más allá de la representación. Esta clasificación se muestra en la Tabla 6.

Tabla 6. Clasificación de IBIS (Ullman, 1995) [171].

Espacio de decisión	1. Completitud del problema	Incompleto
	2. Nivel de abstracción	Cuantitativo o cualitativo
	3. Determinismo	Determinístico o distribuido
Modelo de preferencia	4. Función objetivo	Ninguna
	5. Consistencia	N/A
	6. Base de comparación	N/A
Modelo de creencia	7. Dimensión	Ninguna
	8. Completitud de la creencia	N/A
	9. Foco del problema	Producto o proceso
	10. Rango de independencia	Tipo III: Interdependiente
	11. Nivel de soporte	Nivel 1: representación

Hay gran número de herramientas que usan la metodología IBIS, tales como:

- gIBIS (graphic IBIS) de Conklin y Berman (1988) [28]

- DRAMA (Design RAtionale MAnagement) de Enviros Software Solutions (Brice y Johns, 1999) [18]. Se trata de una herramienta para registrar la base lógica para el diseño de ingeniería. Es una herramienta software que maximiza la reutilización del conocimiento y la memoria corporativos. Así los diseños nuevos extraen el máximo valor de los diseños existentes, manteniendo la seguridad del producto y minimizando los fallos operacionales. Es un sistema basado en computadora multiusuario que da soporte al diseño y operación del proceso.

- R-Objects Pepper de R-Object Inc. (Ernst, 2002) [42]. Se trata de un sistema software que, entre otras cosas, soporta individuos y grupos (potencialmente distribuidos) para la toma de decisiones. Tiene en cuenta los requisitos específicos y se construye usando tecnología de red punto a punto y un modelo de información semántica extensible bastante conocido. Consideran Pepper como una aplicación de la web semántica, y describe su arquitectura y uso en el contexto de la web semántica.

- DRED de Bracewell et al. (2004) [16]. Es una herramienta software que permite a los diseñadores ingenieros registrar su lógica de diseño al mismo tiempo que la generación de ideas y la deliberación de las mismas. También permite realizar gráficos donde se representan los temas tratados, las opciones consideradas y los argumentos a favor y en contra. Se muestran algunos ejemplos en Figura 4, Figura 5, Figura 6 y Figura 7.

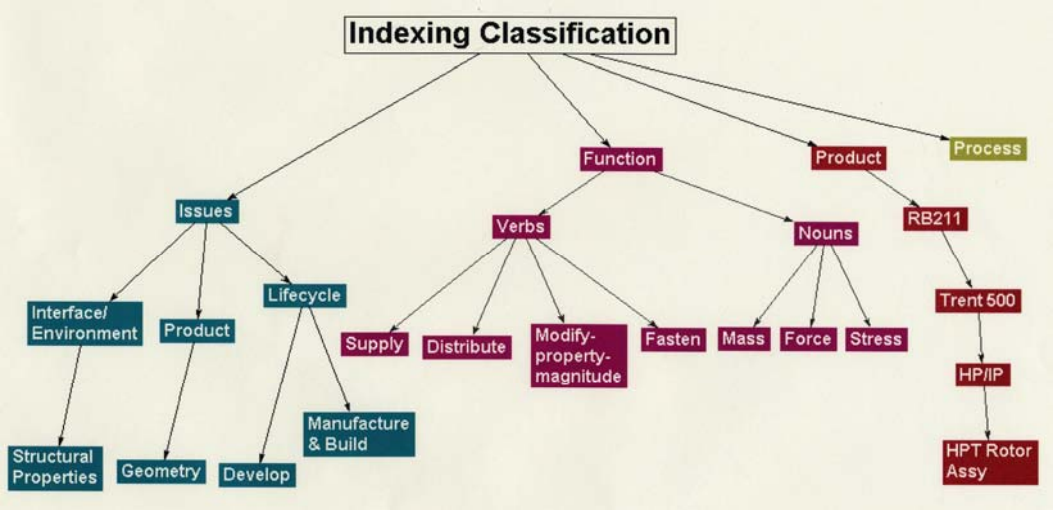

Figura 4. Indexación de la lógica de diseño (UTP Spring Conference, Marzo 2004).

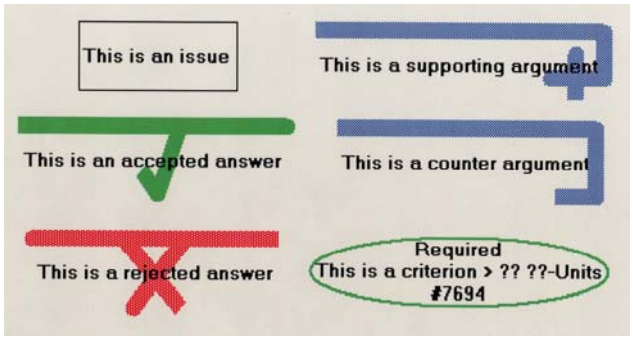

Figura 5. Conjunto de elementos para la representación (UTP Spring Conference, Marzo 2004).

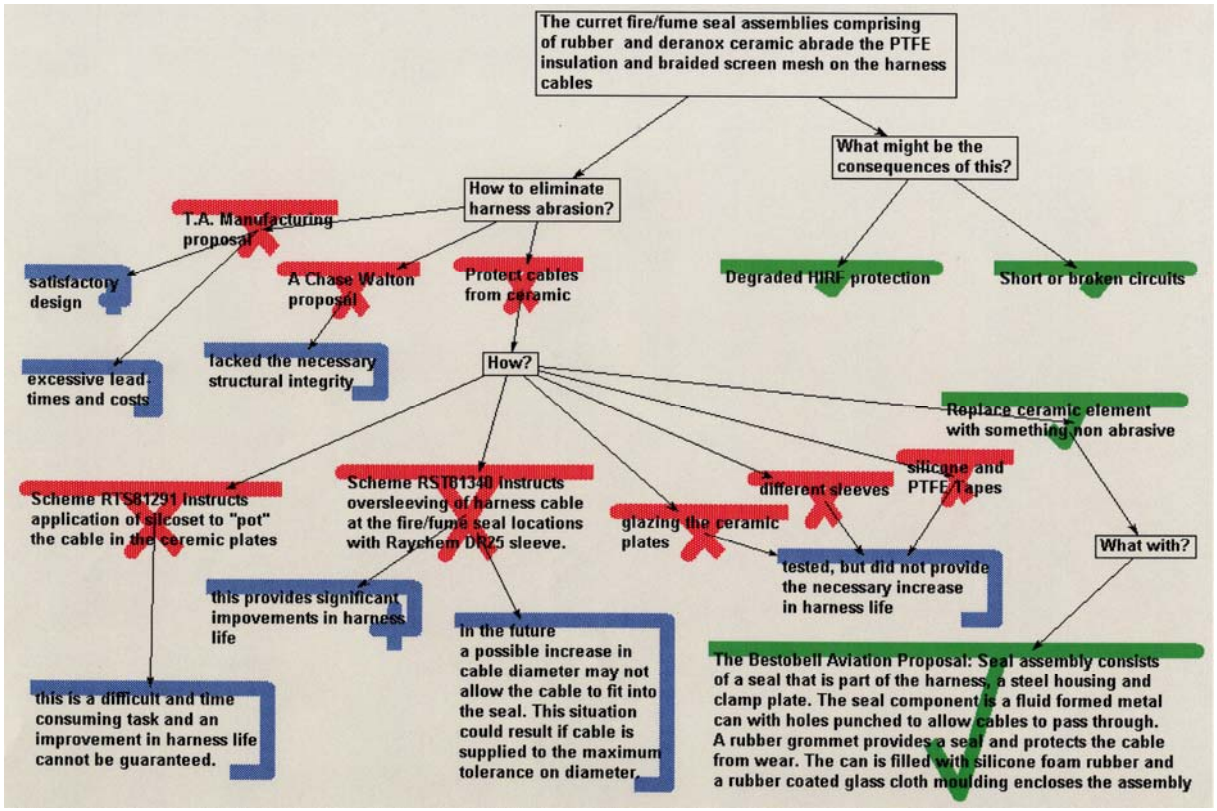

Figura 6. Un ejemplo (UTP Spring Conference, Marzo 2004).

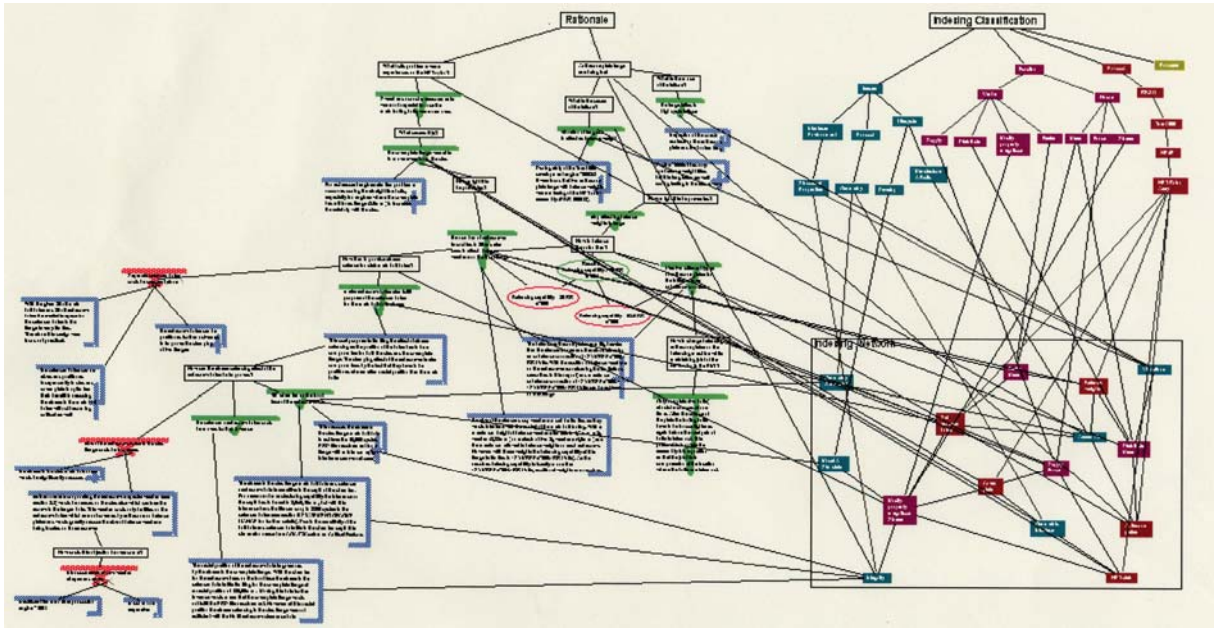

Figura 7. Indexación aplicada al ejemplo anterior (UTP Spring Conference, Marzo 2004).

2.2.1.2. KADS

La estructuración de la adquisición del conocimiento y la documentación (Knowledge Acquisition and Documentation Structuring, KADS) es un método estructurado para el desarrollo de sistemas basados en conocimiento (sistemas expertos). Fue desarrollado en la Universidad de Ámsterdam y actualmente es aceptado como el estándar europeo de facto para los sistemas basados en conocimiento (Wikipedia.org, 2006) [175].

Sus componentes son:

- Una metodología para gestionar proyectos de ingeniería del conocimiento.
- Un banco de trabajo (workbench) de la ingeniería del conocimiento.
- Una metodología para realizar licitaciones de conocimiento.

Posteriormente, KADS fue desarrollado dentro de CommonKADS.

CommonKADS es una metodología que da soporte a la ingeniería del conocimiento estructurada. Se ha desarrollado gradualmente y ha sido validado por muchas compañías y universidades en el contexto del programa europeo ESPRIT[2] IT. Ahora se trata de un estándar de facto para el análisis de conocimiento y el desarrollo de sistemas intensivos en conocimiento. Ha sido adoptado como un todo o parcialmente incorporado en métodos existentes por muchas grandes compañías en Europa, así como en Estados Unidos y Japón.

CommonKADS permite reconocer las oportunidades y cuellos de botella en la forma en que las organizaciones desarrollan, distribuyen y aplican sus recursos de conocimiento, y proporciona herramientas para la gestión del conocimiento corporativo (Schreiber et al, 2000) [152]. CommonKADS también proporciona métodos para llevar a cabo un análisis detallado de las tareas y procesos de conocimiento intensivo. Además, da soporte al desarrollo de sistemas de conocimiento que soportan partes seleccionadas del proceso de negocio.

En la Figura 8 se muestra la pirámide metodológica en CommonKADS.

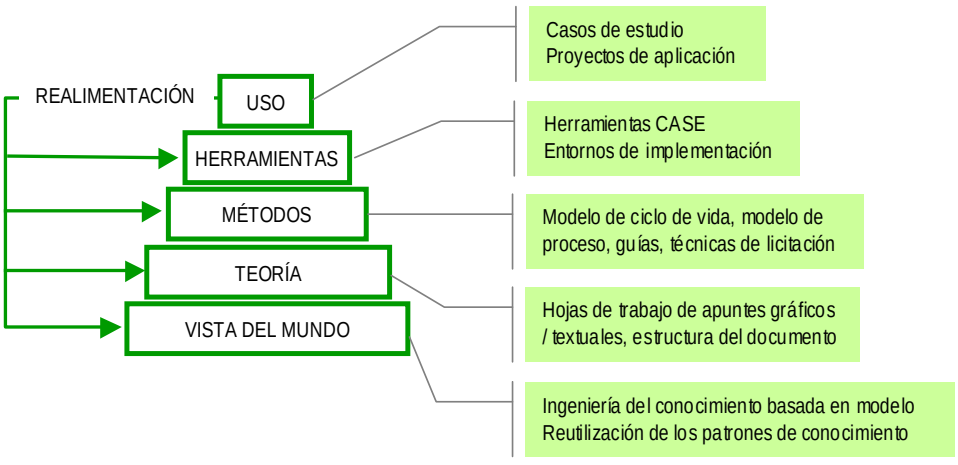

Figura 8. Pirámide metodológica en CommonKADS (Schreiber et al, 2000) [152].

[2] European Strategic Program on Research in Information Technology

Esta metodología está basada en unos cuantos principios básicos (Schreiber et al, 2000) [152]:

- La ingeniería del conocimiento no es una especie de "extracción de la cabeza del experto", sino que consiste en la construcción de modelos de aspectos diferentes del conocimiento humano.

- El principio del nivel de conocimiento: en el modelado del conocimiento, primero hay que concentrarse en la estructura conceptual del conocimiento, y dejar los detalles de programación para más adelante.

- El conocimiento tiene una estructura estable interna que es analizable distinguiendo los tipos de conocimiento y roles específicos.

- Un proyecto de conocimiento debe ser gestionado aprendiendo de las experiencias previas en forma de espiral controlada.

Hoy día, la ingeniería del conocimiento se enfoca como una actividad de modelado.

CommonKADS tiene un conjunto de modelos predefinidos, cada uno de los cuales se centran en un aspecto limitado, pero juntos proporcionan una visión comprensiva (Schreiber et al, 2000) [152], descrita en la Figura 9:

- Modelo de la organización
- Modelo de tarea
- Modelo de agente[3]
- Modelo de conocimiento
- Modelo de comunicación
- Modelo de diseño

Juntos, los modelos de organización, tarea y agente analizan el entorno organizacional y los factores de éxito críticos correspondientes para un sistema de conocimiento determinado. Los modelos de conocimiento y comunicación abarcan la descripción conceptual de las funciones que resuelven el problema y los datos que el sistema de conocimiento maneja y es capaz de entregar. El modelo de diseño convierte esto a una especificación técnica que es la base para la implementación del sistema software.

Esta metodología utiliza estos modelos e incluso Schreiber et al. (2000) la describen por completo en [152].

[3] En este contexto se supone que un agente es cualquier entidad capaz de llevar a cabo una tarea.

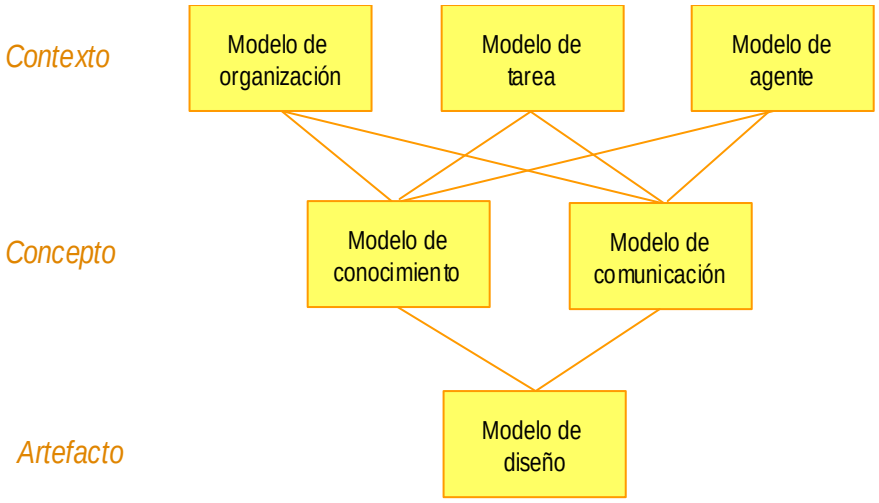

Figura 9. Suite del modelo CommonKADS (Schreiber et al, 2000) [152].

2.2.1.3. Otras metodologías

Whitfield et al. (2000) [176] presentan un sistema genérico que permite la gestión y coordinación de herramientas de análisis de diseño. Este sistema permite que la actividad de diseño sea coordinada de una forma temporal y apropiada. El punto fuerte de este sistema es que proporciona mecanismos para facilitar el proceso de toma de decisiones para el diseñador y gestiona de forma efectiva el proceso de diseño.

En la literatura se pueden encontrar muchas otras propuestas diferentes de herramientas y metodologías para dar soporte a los diseñadores en el proceso de diseño, tal como (Andreasen et al, 1996) [7], (Coates et al, 1999) [25], (Jennings, 1996) [77] y (Shakeri y Brown, 1998) [155].

3. Sistemas MultiAgente

En Informática, un agente de software actúa "de parte de" un usuario o de un programa intermediando con autoridad para decidir cuándo (y si) es apropiado emprender una acción (Moslehi y Kumar, 2006) [111]. La idea es que los agentes no sean requeridos estrictamente para una tarea, sino que puedan activarse dependiendo del contexto percibido.

Los agentes pueden ser inteligentes, es decir, poseer facultades de aprendizaje y razonamiento, y autónomos, con capacidad para adaptarse sin intervención humana a la forma en que consiguen sus objetivos. Pueden estar distribuidos en máquinas físicamente distintas, de acuerdo con las necesidades, y podrían ser móviles, de modo que su ejecución podría transferirse a diferentes procesadores. Los sistemas de múltiples agentes constan de agentes distribuidos que alcanzan un objetivo actuando en cooperación. Pueden ejecutar sus tareas de forma síncrona o asíncrona y, si es necesario, acceder a bases de datos descentralizadas.

El diseño de sistemas basados en agentes debe considerar el medio de proporcionar la capacidad para a) priorizar, programar y/o sincronizar tareas, b) facilitar la comunicación y colaboración, teniendo una naturaleza apropiada para representar conocimientos y metadatos organizados jerárquicamente y c) detectar todos los posibles cambios en el entorno y responder a ellos.

Un sistema multiagente (Wooldridge, 2002) [180] es un sistema constituido por un número de agentes, cada uno de ellos con capacidades autónomas de decisión y acción, que interactúan entre sí. Para interactuar satisfactoriamente, los agentes necesitan las habilidades de cooperación, coordinación y negociación.

Wooldridge y Jennings presentaron esta definición en 1995 [181]: "Una agente es un sistema informático situado en algún entorno, y que es capaz de realizar de forma autónoma acciones sobre ese entorno para alcanzar los objetivos de diseño". Véase Figura 10.

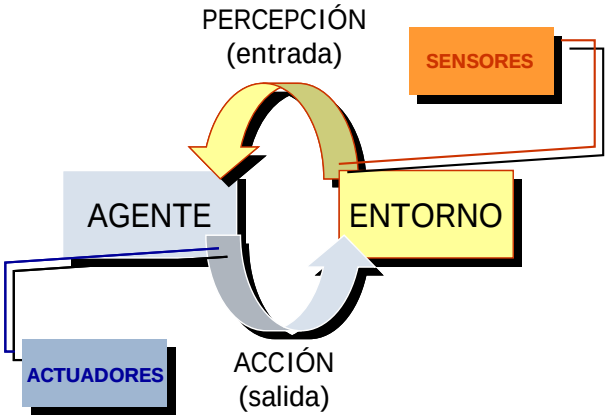

Figura 10. El agente y su entorno.

¿Cómo influye el entorno en el agente (Figura 10)? En la mayoría de los casos el agente sólo será capaz de controlar parte del entorno y una misma acción realizada por el mismo agente en ocasiones diferentes puede tener efectos muy distintos (entornos no deterministas). Por ello, un agente debe estar preparado para fallar o para la incertidumbre de no saber si ha tenido éxito o no y dispone de un repertorio de acciones disponibles con sus correspondientes precondiciones.

El principal problema al que se enfrenta un agente es decidir qué acción realizar para alcanzar sus objetivos de diseño.

Tal como Sycara afirma en (1998) [166], la capacidad de un agente inteligente está limitada por su conocimiento, sus recursos de computación y su perspectiva. Para que los agentes que conforman un sistema multiagente puedan interactuar, es necesaria la aplicación de técnicas de negociación y cooperación, que veremos más adelante.

Según Sycara las características de los sistemas multiagente en cuanto a la interacción entre agentes son:

- Cada agente tiene información parcial e incompleta y capacidades para resolver el problema y, por tanto, una visión limitada del entorno. En la Figura 11 se muestra un esquema propuesto por Wooldridge en el que se plasma este hecho mediante lo que él denominó la esfera de influencia del agente.
- No existe un control global del sistema.
- Los datos están descentralizados.
- La computación es asíncrona.

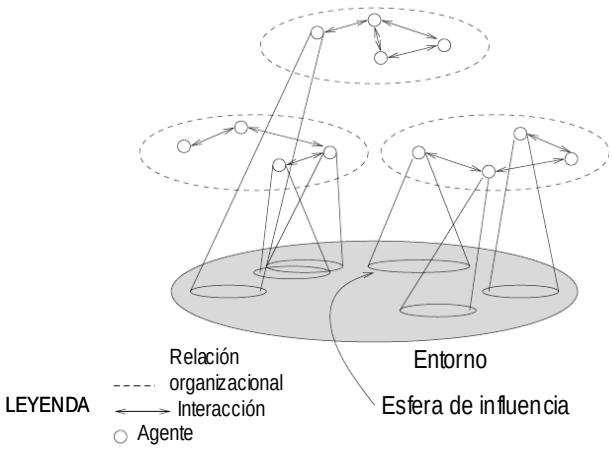

Figura 11. Visión parcial de los agentes en un entorno (Wooldridge, 2002) [180].

Es decir, que un sistema multiagente carece de perspectiva global, control global o dato global, pero son capaces de:

- Resolver problemas que son demasiado grandes como para que un agente centralizado pueda resolverlos.

- Permiten la interconexión e interoperación de múltiples *legacy systems*[4].
- Proporcionar soluciones a problemas que pueden ser considerados como una sociedad de agentes autónomos que interactúan.
- Proporcionar soluciones que usan de forma eficiente fuentes de información que se encuentran espacialmente distribuidas.
- Proporcionar soluciones en situaciones donde las habilidades están distribuidas.
- Mejorar el rendimiento en cuanto a:
  - Eficiencia computacional
  - Fiabilidad
  - Escalabilidad
  - Robustez
  - Mantenimiento
  - Receptividad
  - Flexibilidad
  - Reutilización

Para finalizar esta introducción a los MAS, haremos una breve comparación entre con los sistemas expertos:

- Los agentes que componen un MAS interactúan con el entorno, mientras que los sistemas expertos son tradicionalmente sistemas cerrados.
- En los MAS, la toma de decisiones es distribuida, ya que cada agente toma sus propias decisiones, mientras que los sistemas expertos son sistemas de decisión centralizados.
- En los MAS el grado de interacción con el usuario es mayor que en los sistemas expertos, en los que la interacción con el usuario suele ser bajo petición. Además en los MAS, también existe interacción con otros agentes.

3.1. Características de un agente

Las características básicas de una agente son:

- ⇨ Autonomía: Los agentes actúan sin intervención humana directa o de otros agentes y tienen alguna clase de control sobre sus acciones y estado interno. El software tradicional se ejecuta en entornos interactivos, donde responde a órdenes directas del usuario.
- ⇨ Reactividad: Percibe el entorno en el que está inmerso y responde de manera oportuna a cambios que tienen lugar en él (para actuar adecuadamente un agente debe poder conocer en todo momento el "mundo" que le rodea).
- ⇨ Iniciativa (pro-actividad): Tiene que tener un carácter emprendedor y tomar la iniciativa para actuar guiado por los objetivos que debe satisfacer. En cada momento el agente decide qué acción llevar a cabo. No sólo actúa en función de los estímulos que percibe sino que realiza acciones como resultado de sus decisiones.
- ⇨ Sociabilidad: Capacidad de interaccionar con otros agentes (incluso humanos) utilizando alguna clase de lenguaje de comunicación de agentes. Los agentes colaboran entre sí para la ejecución de tareas (MAS).

[4] Existen dos acepciones para la expresión "legacy systems": 1) son sistemas obsoletos que siguen utilizándose en una empresa porque no se quiere hacer el esfuerzo de reemplazarlo o rediseñarlo; 2) también hacen alusión a grandes sistemas corporativos en los que una compañía ha invertido gran esfuerzo humano y económico.

Los agentes pueden tener otras características adicionales, y que, en ocasiones, se implementan de forma habitual:

- ⇨ Movilidad: habilidad para trasladarse en una red electrónica (agentes móviles).
- ⇨ Veracidad: se supone que un agente no comunica información falsa de forma intencionada.
- ⇨ Benevolencia: se supone que un agente no tiene objetivos contradictorios y siempre intenta realizar la tarea que se le solicita.
- ⇨ Inteligencia: racional, coherente y adaptable.
- ⇨ Racional: el agente tiene unos conocimientos de su entorno, unos objetivos y unas reglas que determinan cómo alcanzar los objetivos a partir del conocimiento que maneja.
- ⇨ Coherente: el conocimiento que maneja el agente (base de conocimiento) tiene un alto grado de cohesión, para que el comportamiento del agente sea el adecuado.
- ⇨ Adaptable: el agente es capaz de actualizar su base de conocimiento y su comportamiento (base de reglas) a partir de las percepciones que recibe del entorno y de sus comportamientos anteriores (aprender). Es una de las características más complejas y difíciles de llevar a cabo.

3.2. Clasificación de agentes

En la literatura podemos encontrar multitud de clasificaciones de agentes, pero aquí nos quedaremos con la clasificación clásica que en su momento propuso Wooldridge en (Wooldridge, 2002) [180], ya que recoge una amplia visión según tres características importantes en la implementación de agentes: el grado de inteligencia del agente, su grado de movilidad y el número de agentes que conforma el sistema completo.

En la Figura 12 se muestra una representación tridimensional en la que dependiendo del número de agentes que conformen el sistemas podremos hablar de agente único o sistema multiagente; dependiendo del grado de inteligencia del agente podremos hablar de agente simple o complejo; y dependiendo del grado de movilidad podremos hablar de agentes inmóviles y agentes móviles.

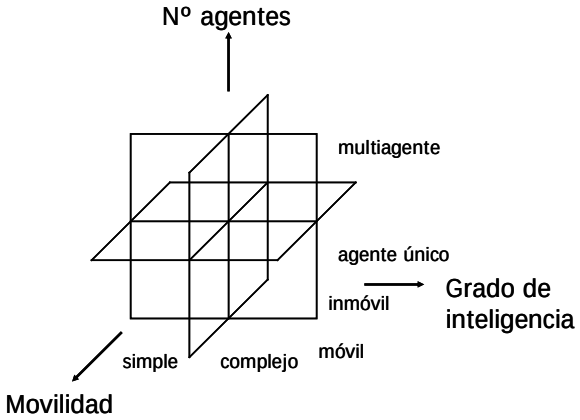

Figura 12. Clasificación de los agentes según Wooldridge (2002) [180].

En la Figura 13 se ilustra esa misma representación tridimensional en la que se indican características de distintos tipos de agente de acuerdo con la clasificación dada por Wooldridge.

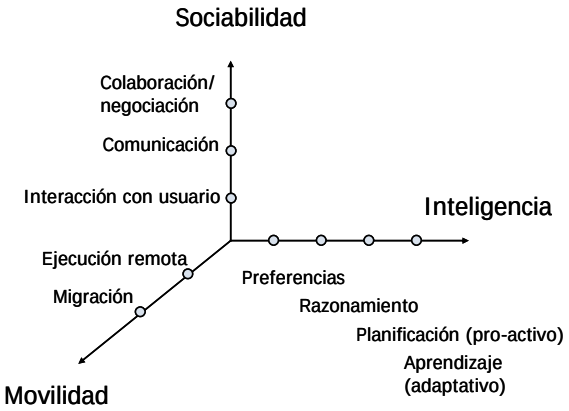

Figura 13. Tipos de agentes de acuerdo con la clasificación de Wooldridge.

En la Tabla 7 se muestra otra clasificación de agentes, esta vez propuesta por Woods y Barbacci (1999) [178].

Tabla 7. Clasificación de agentes según Woods y Barbacci (1999) [178].

<table>
<tr><th>Criterio</th><th colspan="2">Tipología de agentes</th></tr>
<tr><td rowspan="6">Basado en muchas apariencias físicas y muchos roles como los ejes en un espacio de agentes</td><td rowspan="2">Movilidad</td><td>Estático</td></tr>
<tr><td>Móvil</td></tr>
<tr><td rowspan="2">Comportamiento</td><td>Deliberativo: tiene un modelo interno.</td></tr>
<tr><td>Reactivo: tiene un comportamiento estímulo/respuesta.</td></tr>
<tr><td rowspan="3">Atributos deseados</td><td>Autonomía</td></tr>
<tr><td>Aprendizaje</td></tr>
<tr><td>Cooperación</td></tr>
<tr><td rowspan="7">Considerar atributos adicionales, tales como la versatilidad, benevolencia/poco servicial, antagonista/altruista, actitudes (emoción, creencia/deseo/intención)</td><td colspan="2">Colaborativo: muestra comportamiento de aprendizaje y cooperación o autonomía y cooperación</td></tr>
<tr><td colspan="2">Interfaz: muestra los atributos de autonomía y aprendizaje (actuando en nombre de sus propietarios)</td></tr>
<tr><td colspan="2">Móvil: se mueve en nombre de sus propietarios</td></tr>
<tr><td colspan="2">Información/Internet: gestionar información a partir de fuentes distribuidas</td></tr>
<tr><td colspan="2">Reactivo: muestra comportamiento emergente (estímulo / respuesta)</td></tr>
<tr><td colspan="2">Híbrido: una mezcla de los tipos de agentes anteriores</td></tr>
<tr><td colspan="2">Inteligente: autónomo, capaz de aprender y cooperar</td></tr>
<tr><td rowspan="3">Definiendo estas propiedades:
• Actúa en nombre de otras entidades de forma autónoma
• Realiza acciones son algún grado de pro-actividad y/o reactividad
• Muestra algún nivel de los atributos clave (aprendizaje, cooperación, movilidad)</td><td colspan="2">Heterogéneo: similar al híbrido pero integrado, incluyendo sistemas heredados</td></tr>
<tr><td colspan="2">Agentes interactivos: Actúan en nombre de un usuario, trata de adquirir conocimiento sobre el comportamiento del usuario y anticipa/optimiza la visualización de información. Las arquitecturas de agentes interactivos incluyen agentes de filtrado de información, agentes de recuperación de información y asistentes digitales personales.</td></tr>
<tr><td colspan="2">Agentes distribuidos: Su principal atributo es la cooperación; los agentes se comunican, coordinan y negocian entre sí. Usan modelos de coordinación alternativos: organizacional (central), por contrato/puja, y por planificación previa (central, distribuida). Los agentes usan modelos de negociación alternativos: competitivo (agentes independientes con objetivos independientes) y cooperativo (todos los agentes tienen un único objetivo común).</td></tr>
<tr><td colspan="2">Agentes móviles: Agentes que combinan múltiples modelos: modelo del ciclo de vida, modelo computacional, modelo de seguridad, modelo de comunicación y modelo de navegación. La mayoría de los modelos son implementados por el entorno, que también proporcionan servicios de distintos tipos.</td></tr>
</table>

3.3. Arquitecturas

En la extensa literatura sobre el tema, podemos encontrar diversidad de modelos en los que los autores presentan distintas arquitecturas. Por ejemplo, en 1997, Bradshaw [17] propuso una arquitectura enfocada al uso de agentes (Figura 14).

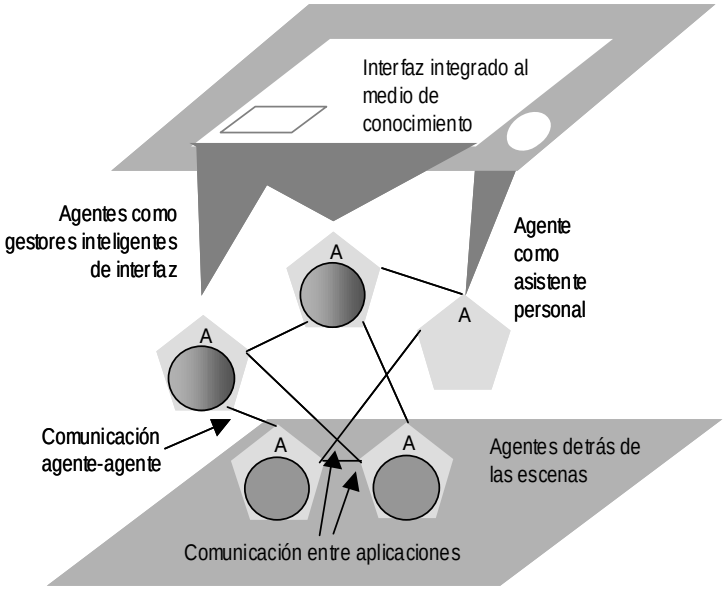

Figura 14. Arquitectura para el uso de agentes (Bradshaw, 1997) [17].

Esta arquitectura propuesta por Brawshaw ilustraba que, en una arquitectura preparada para el uso de agentes, éstos podrían tomar varios. Algunos podían actuar en el papel de gestores de interfaces de usuario inteligentes, dibujando los recursos de otros agentes que se encontraran trabajando detrás de las escenas. Estos agentes trabajarían de acuerdo para ayudar a coordinar la selección de los modos de visualización y representaciones para los datos importantes, incorporando representaciones semánticas del conocimiento en los documentos para mejorar la navegación y la recuperación de información. Ya que la distribución y el contenido de las vistas podría venir dado por el contexto y los modelos de configuración, más que por código del interfaz de usuario introducido manualmente, el ahorro económico era significativo en cuanto que los componentes de datos y software podían ser reutilizados y reconfigurados semi-automáticamente para diferentes parámetros y propósitos. Algunos agentes podrían ser representados por el usuario como varios tipos de asistentes personales. Desde el punto de vista ideal, cada componente software podría estar preparado para ser un agente, sin embargo, por razones prácticas se mantuvieron los mecanismos de comunicación entre aplicaciones tradicionales, en vez de usar los protocolos basados en agentes.

Aunque hemos querido ilustrar un ejemplo de los muchos existentes, se puede decir que actualmente, para construir un sistema multiagente, se suelen considerar tres posibles arquitecturas básicas:

- **Arquitecturas puramente deliberativas**: en este tipo de arquitectura cada agente individual puede razonar acerca de efectos no locales de acciones locales, formar expectativas sobre el comportamiento de otros agentes, o explicar y posiblemente reparar

conflictos e interacciones perjudiciales. Se define un axioma para formalizar un modelo para el comportamiento de los agentes en términos de creencias, deseos e intenciones, también conocidas como las arquitecturas BDI (Belief-Desire-Intention) (véase Figura 15).

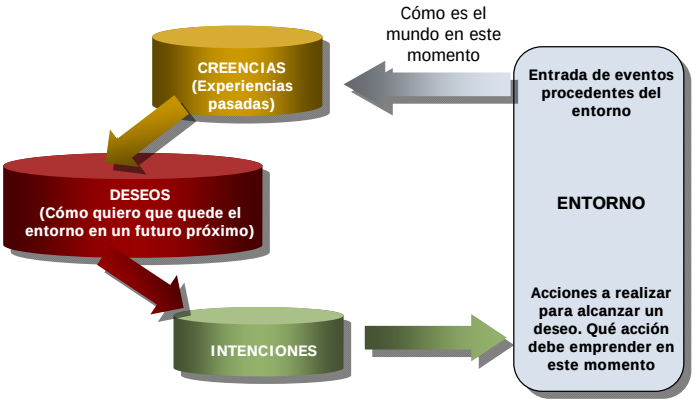

Figura 15. Arquitectura BDI (Beliefs-Desires-Intentions).

- **Arquitecturas puramente reactivas**: en esta arquitectura, la inteligencia es el producto de la interacción entre una gente y su entorno y el comportamiento inteligente emerge de la interacción entre varios comportamientos simples organizados de una forma ordenada mediante una relación de inhibición maestro-esclavo. Los agentes reactivos no tienen representaciones de su entorno y actúan usando un tipo de comportamiento estímulo-respuesta y toma decisiones basadas en información local.

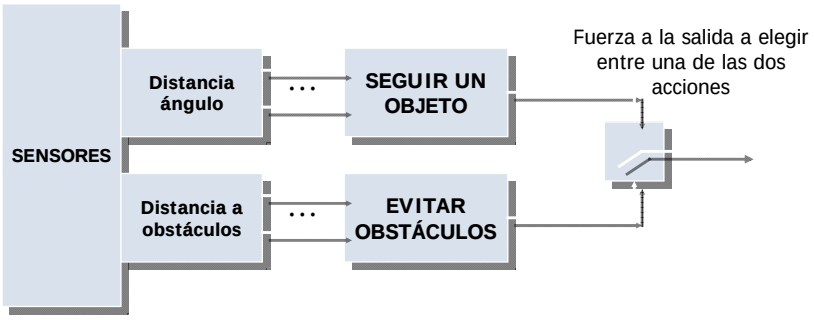

Figura 16. Arquitectura reactiva. Ejemplo.

- **Arquitecturas híbridas**: estas arquitecturas combinan aspectos de las dos anteriores. Generalmente se construye como una serie de capas software, cada una de las cuales comprende un nivel abstracción diferente. La mayoría de las arquitecturas de este tipo tienen suficiente con tres capas: una capa reactiva, haciendo de la capa más baja, una capa intermedia abstrayendo la visión del nivel de conocimiento del entorno del agente, y una capa más alta donde se cubren los aspectos sociales del entorno (incluyendo la coordinación).

Una organización proporciona un marco de trabajo para las interacciones entre los agentes mediante la definición de roles, las expectativas de comportamiento y las relaciones de autoridad. Consiste en un grupo de agentes, un conjunto de actividades a realizar por el

agente, un conjunto de conexiones entre los agentes y un conjunto de objetivos o criterio de evaluación para que los agentes sean evaluados. Algunos ejemplos más significativos de organización son:

- **Jerarquía**: los agentes interactúan siguiendo una estructura jerárquica.
- **Comunidad de expertos**: los agentes interactúan siguiendo una estructura plana.
- **Mercado**: los agentes usan mecanismos contractuales y de pujas.
- **Comunidad científica**: los problemas se resuelven de forma local y después se comunica la solución a otros miembros de la comunidad para que puedan testearla, cuestionarla y refinarla.

Los canales a través de los cuales se mueve la información de un agente a otro se distinguen en función del medio, el direccionamiento, la persistencia y la localidad (Parunak, 1998) [121]:

✓ El **medio**: Los agentes sólo pueden comunicarse mediante cambios en su entorno físico compartido (véase Figura 17) o, más comúnmente, mediante el intercambio de mensajes digitales sobre una red de comunicaciones. En este último caso, también es importante tener en cuenta que la información que fluye a través del mundo no digital, ya que esta información es parte del protocolo de comunicaciones completo.

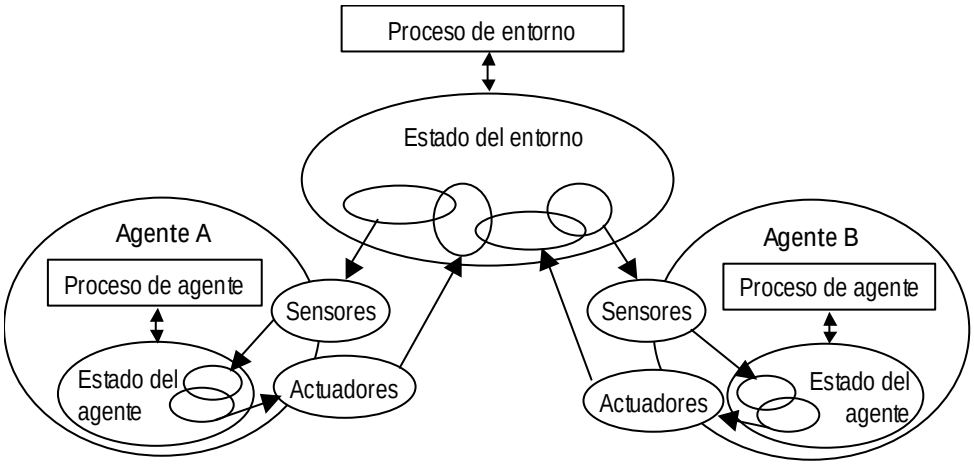

Figura 17. Interacción de los agentes a través de un entorno compartido (Parunak, 1998) [121].

✓ El **direccionamiento**: Los mensajes pueden ser difundidos a toda la comunidad de agentes o a algún subconjunto de la población. Las dos formas importantes de direccionamiento restringido son el direccionamiento directo (en el que un agente especifica los destinatarios particulares del mensaje) o un direccionamiento basado en tema (en el que se etiquetan los mensajes por contenido en vez de por destinatario, y son reenviados a todos los agentes que estén suscritos al área del contenido especificado).

✓ La **persistencia**: Los mensajes pueden persistir durante un periodo de tiempo después de ser enviados (como en un sistema de pizarra), o pueden existir sólo de forma transitoria mientras viajan por la red.

✓ La **localidad**: Algunos agentes pueden moverse en el espacio de comunicaciones, así como cambiar su proximidad comunicativa a otro. La clave en esta categoría es

precisamente eso, la proximidad comunicativa. Los robots móviles que se comunican con otros sobre un enlace de radio de difusión no viajan en este sentido, ya que sus relaciones de comunicación no se ven afectadas por su movimiento. Pero un agente que representa una parte de una fábrica, en la que hay un procesador escolta que se comunica sólo con los lectores a los que se aproxima físicamente, es realmente un tipo de "mensaje inteligente" y cae en esta categoría.

Un protocolo de comunicaciones determina cómo se estructuran las conversaciones entre los agentes:

- ✓ *Directiva*: Los agentes dan orden a algún otro agente.

- ✓ *Votación*: Los agentes expresan una cantidad escalar, y su comportamiento viene determinado por alguna medida de agregación (tal como una suma, producto o media) sobre escalares de diferentes agentes.

- ✓ *Negociación*: Como en una votación, los agentes negocian el intercambio de una serie de mensajes antes de tomar una decisión. El ejemplo canónico es la red de contratos (contract net) de Davis y Smith (1983) [32]. La votación y la negociación difieren en tres aspectos fundamentalmente. El primero es que mientras que tanto una puja como una votación están pensadas para influir en el comportamiento del receptor, la puja tiene el objetivo adicional de adquirir más tareas para el vendedor. En segundo lugar, los intercambios en una votación tienden a ser más cortos que en una negociación, ya que el protocolo debe incluir pasos adicionales para acordar el contrato y gestionar la ejecución de la tarea. En tercer lugar, aunque las pujas más simples son sólo escalares, en general las subastas pueden llegar a ser mucho más complicadas, incluyendo la información simbólica extensa que representa el tiempo y las especificaciones de las tareas a ser realizadas.

- ✓ *Actos del habla*: Los protocolos de negociación determinan un conjunto definido de opciones que una conversación puede seguir, y esta estructura es definida cuando los agentes están implementados. Recientemente, la teoría de los actos del habla ha sido utilizada para construir gramáticas generales independientemente de qué protocolos arbitrarios puedan construirse, y los agentes que comprenden estas gramáticas pueden desarrollar nuevos protocolos para sus conversaciones cuando estén funcionando.

Dependiendo de dónde esté centrado nuestro interés, tenemos que diseñar diferentes aspectos, tal como se muestra en la Tabla 8.

Tabla 8. ¿Qué es necesario diseñar? (Parunak, 1998)[121]

La comunidad del agente (Nivel social)	Protocolos (dinámicos de comunicación y coordinación)
	Organización (roles o servicios de cada agente con respecto a los otros)
El agente individual (Nivel de conocimiento)	Planificación local (capacidades y planes)
	Comportamiento local (reactividad, tareas rutinarias)
	Conocimiento local (las creencias del agente)

3.4. Aplicaciones de los sistemas multiagente

A lo largo de las dos últimas décadas son abundantes las aplicaciones que se han desarrollado basándose en el enfoque de los sistemas multiagente (véase Figura 18), así como el número de disciplinas cubiertas, que no son nada desdeñables (véase Figura 19).

En este apartado, aportaremos una visión general de los tipos de aplicaciones y su repercusión en el ámbito científico-tecnológico.

Figura 18. Dominios de aplicación de los sistemas multiagente

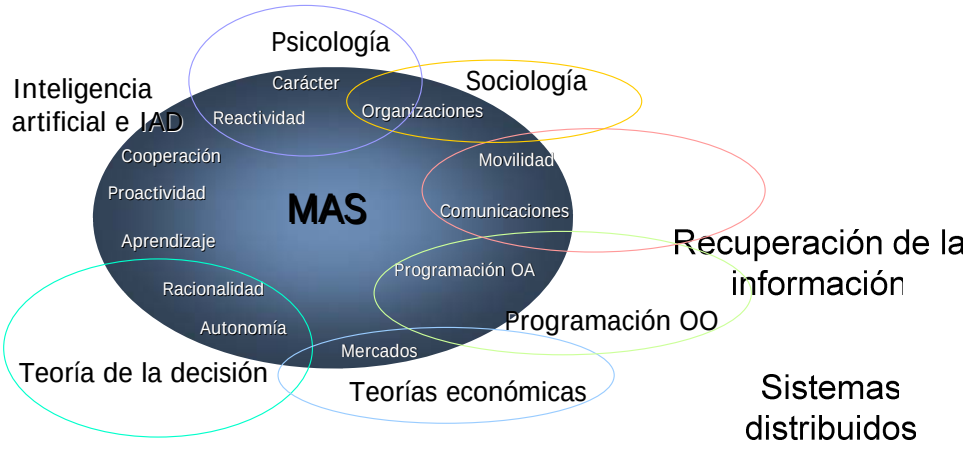

Figura 19. Disciplinas que cubren los sistemas multiagente.

En líneas generales, los agentes pueden ser utilizados para (Wooldridge, 2002) [180]:

- la gestión de los flujos de trabajo y de los procesos de negocio,
- la recolección de información mediante sensores distribuidos,
- la recuperación de información y su gestión,
- el comercio electrónico,
- el desarrollo de interfaces hombre-máquina,
- el desarrollo de entornos virtuales,

– la realización de simulaciones sociales,
– otras aplicaciones.

Por un lado, Jennings et al. (1998) [82] han descrito en la Tabla 9 las aplicaciones más importantes de los sistemas multiagentes.

Tabla 9. Aplicaciones de los MAS (Jennings et al, 1998) [82].

Campo	Especialidades	Ejemplos de sistemas
Aplicaciones industriales	Fabricación	YAMS de Parunak (1987) [120]
		PACT de Cutosky et al (1993) [29]
		MACIV de Oliveira et al (1997) [130]
		Amacoia de Sprumont y Müller (1997) [159]
	Control de procesos	ARCHON de Jennings et al (1996) [79]
		Wang y Wang (1997) [173]
		Schwuttke y Quan (1993) [153]
	Telecomunicaciones	Griffeth y Velthuijsen (1994) [62]
		Adler et al (1989) [1]
		Esfandiari et al (1996) [43]
	Control de tráfico aéreo	OASIS de Ljunberg y Lucas (1992) [97]
	Sistema de transporte	Burmeister et al (1997) [19]
		Fischer et al (1996) [51]
Aplicaciones comerciales	Gestión de información	Maxims de Maes (1994) [102]
		WARREN de Sycara et al (1996) [167]
		WEBMATE de Chen y Sycara (1998) [22]
	Comercio electrónico	Kasbah de Chavez y Maes (1996) [21]
		BargainFinder de Krulwich (1996) [89]
		Jango de Doorembos et al (1997) [39]
		MAGMA de Tsvetovatyy et al (1997) [170]
	Gestión de proceso de negocio	ADEPT de Jennings et al (1996) [80]
		Fox et al (1993) [56]
		Huhns y Singh (1998) [73]
		Merz et al (1997)[109]
Aplicaciones de ocio	Juegos	Grand y Cliff (1998) [61]
		Wavish et al (1996) [174]
	Teatro interactivo y cine	Bates (1994) [10]
		Trappl y Petta (1997) [169]
		Hayes-Roth (1995) [70]
		Lester y Stone (1997) [95]
		Foner (1997) [52]
Aplicaciones médicas	Monitorización de pacientes	GUARDIAN de Hayes-Roth et al (1989) [71]
	Cuidado de la salud	Huang et al (1995) [72]

Por otro lado, AgentLink [100] ha elaborado un documento [100] donde, entre muchas otras cuestiones relacionadas con los agentes, se incluyen algunos ejemplos de aplicaciones de sistemas multiagente en producción en el año 2005 (Tabla 10). También señalan (Figura 20) cuáles son las tendencias emergentes en el dominio de los sistemas multiagente y cómo la fabricación, el transporte, las telecomunicaciones y la medicina promoverán el desarrollo de agentes (Figura 21).

Tabla 10. Ejemplos de aplicación de agentes (AgentLink, 2006) [99].

Astronomía	**sSTAR** (eScience Telescopes for Astronomical Research): agentes desarrollados en el telescopio infrarrojo británico (UKIRT, United Kingdom Infrared Telescope) en Hawaii.
Aeroespacial	**Aerogility**: sistema multiagente para ayudar a los gerentes a comprender mejor la complejidades del post-mercado aeroespacial.
	RAX (Remote Agent eXperiment): un agente software para dirigir una nave espacial.
Industria	**Ocean i-Scheduler**: un optimizador basado en agente desarrollado por Magenta Technology para el uso de la planificación en tiempo real de la asignación de carga a naves en una flota de transporte de crudo.
E-comercio	El interfaz del programa de aplicación de **eBay** permite a los usuarios definir sus propios agentes de pujas automatizadas.

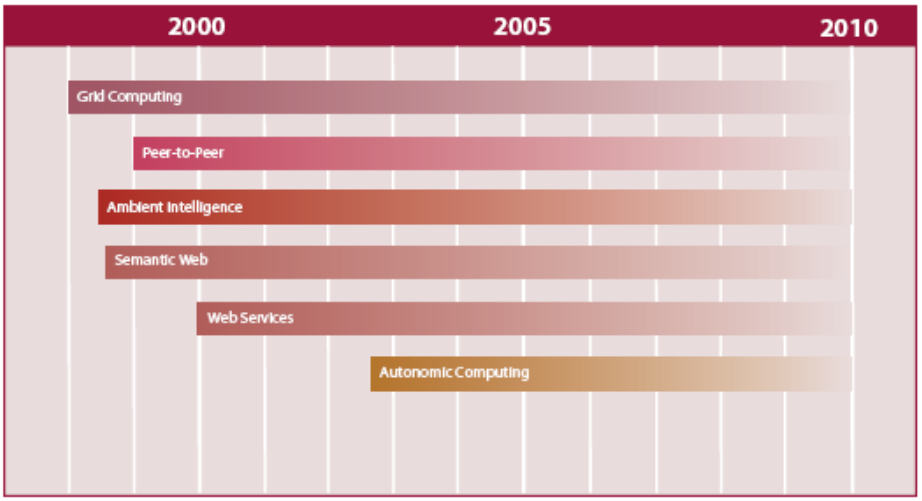

Figura 20. Tendencias emergentes en el dominio de los agentes (AgentLink, 2005) [100].

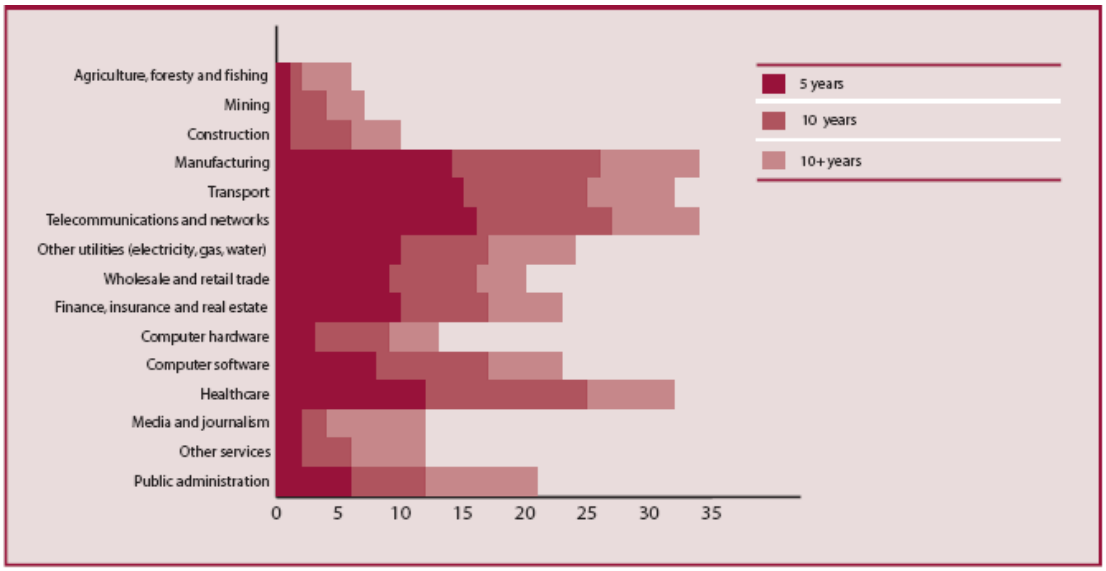

Figura 21. Áreas donde se prevé una promoción del desarrollo de agentes (AgentLink, 2005) [100].

En la Tabla 11 se muestran las tecnologías de agente que comprenden las áreas que serán cubiertas en diferentes escalas de tiempo y la muestra cuál es la actividad reciente en Europa en el ámbito de la computación basada en agente, ambos presentados por AgentLink (2005) en su informe anual [100].

Tabla 11. Tecnología de agentes que comprenden áreas que serán cubiertas en diferentes escalas temporales (AgentLink, 2005) [100].

	Corto plazo	Medio plazo	Largo plazo
Software de robustez industrial	Peer to peer Herramientas de desarrollo mejores UML para agentes Computación orientada al servicio	Diseños genéricos para coordinaciones Librerías para el desarrollo orientado a agentes	Mejores prácticas en el diseño de sistemas de agentes
Estándares de acuerdo	FIPA ACL Descripción semántica	Lenguajes de negocio flexibles Librerías de protocolos de interacción	Herramientas para la evolución de los lenguajes y protocolos de comunicaciones
Infraestructura para comunidades abiertas	Minería web Integración de datos y web semántica Metadato	Interacción semántica Servicios web semánticos capacitados para agentes Instituciones electrónicas Normas, roles, leyes y organizaciones dinámicas	Ontologías mejoradas y compartidas
Razonamiento en entornos abiertos	Vistas organizacionales de sistemas de agente	Entendimiento mejorado de sociedades de agentes dinámicas Teoría y práctica de estrategias de argumentación Normas y estructuras sociales Teoría y práctica de estrategias de negociación	Sistemas de eCiencia automatizados y otros dominios de aplicación
Tecnologías de aprendizaje	Adaptación Personalización Tecnologías híbridas	Agentes evolutivos Autoorganización Aprendizaje distribuido	Configuración en tiempo de ejecución y rediseño
Confianza y reputación	Seguridad y verificación para agentes Test de fiabilidad para agentes Protocolos de autoimposición	Normas y estructuras sociales Mecanismos de reputación Métodos formales para sistemas de agentes abiertos Contratos electrónicos	Técnicas de confianza para copiar con agentes maliciosos

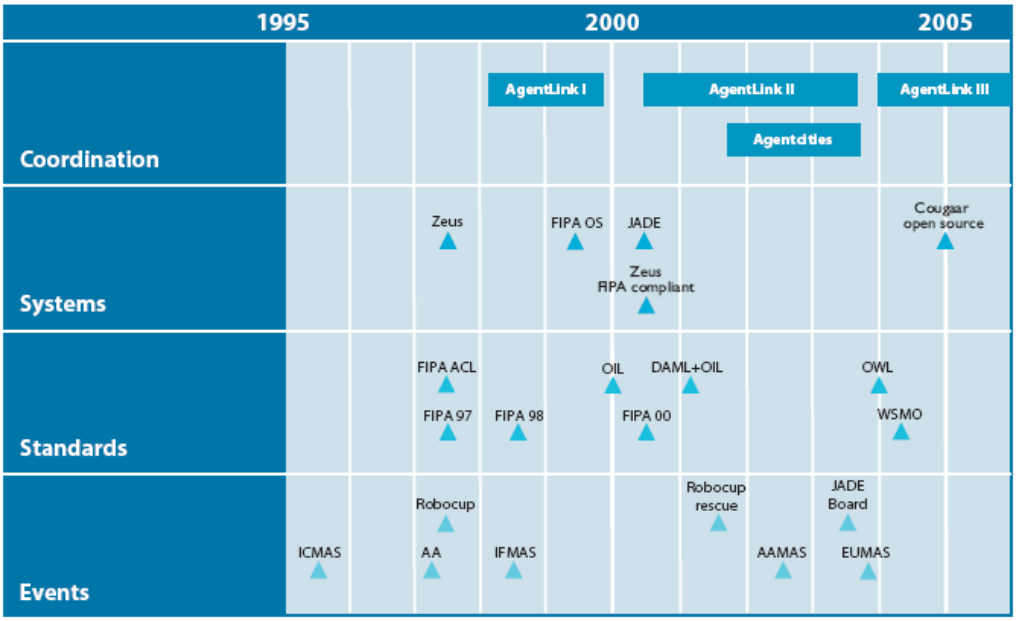

Figura 22. Actividad europea en computación basada en agentes (AgentLink, 2005) [100].

3.5. Lenguajes de comunicación entre agentes

Como ya hemos comentado, las distintas formas de interaccionar que tienen los agentes son:

- la comunicación mediante el entorno,
- los sistemas de pizarra,
- mediante el nivel de conocimiento de los agentes, y
- sin comunicación explícita (en este caso los agentes usan inferencia basándose en teoría de juegos).

Pero para que esa interacción sea posible, es necesario contar con un lenguaje de comunicación entre agentes, destacando los dos estándares más conocidos:

- En el marco de FIPA (Foundation for Intelligence Physical Agents, 1996), se definió el lenguaje **FIPA ACL (**Agent Communication Language). Se trata de un lenguaje basado en la teoría de actos del habla, en el que se definen unas performativas, se emplea comunicación en el nivel de conocimiento y se define SL (Semantic Language) como lenguaje semántico.

  FIPA (2006) [54] es actualmente una organización de estándares del IEEE Computer Society que promueve la tecnología basada en agente y la interoperabilidad de sus estándares con otras tecnologías. Esta organización fue oficialmente aceptada por el IEEE como su decimoprimero comité de estándares el 8 de junio de 2005. Originalmente, FIPA se fundó en Suiza en 1996 para producir especificaciones estándares de software para agentes heterogéneos y que interactúan, así como para sistemas basados en agente. Desde su fundación, FIPA ha jugado un papel crucial en el desarrollo de los estándares para agentes y ha promovido gran número de iniciativas y eventos que han contribuido al desarrollo y evolución de la tecnología agente. Además, muchas de las ideas originadas y desarrolladas en FIPA constituyen hoy día un punto importante en las nuevas generaciones de la tecnología web/Internet y las especificaciones relacionadas.

  Las especificaciones de comunicaciones entre agentes de FIPA incluyen los mensajes ACL, los protocolos de interacción de intercambio de mensajes, los actos comunicativos basados en la teoría de actos del habla y las representaciones del lenguaje de contenido. En la Figura 23 se ilustra la arquitectura y la encapsulación de los mensajes a través de la red y en la Figura 24 el formato de los mensajes FIPA.

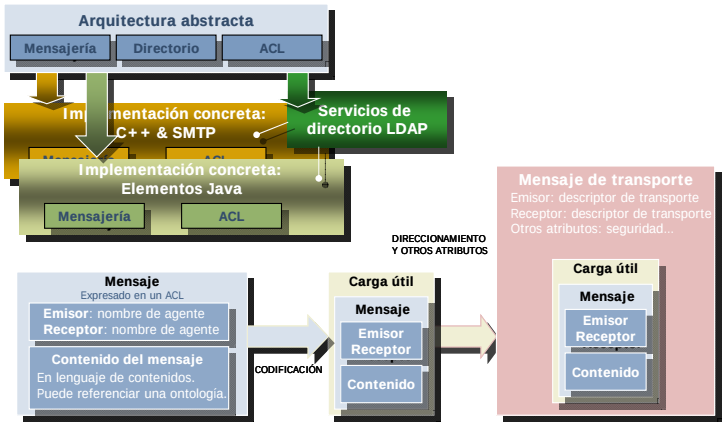

Figura 23. Arquitectura y mensajes para FIPA ACL.

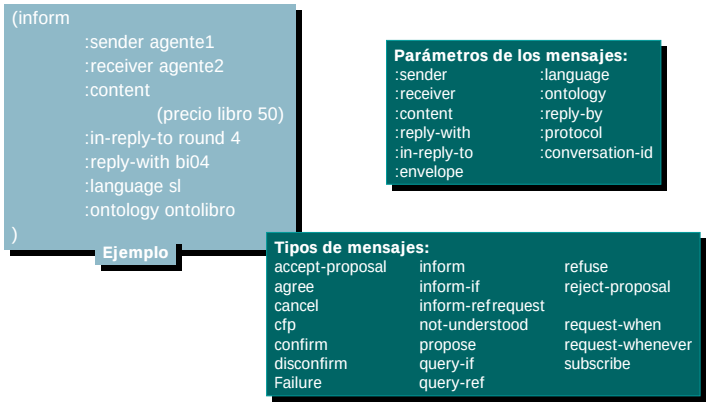

Figura 24. Formato de mensajes FIPA.

– Como parte del KSE (Knowledge Sharing Effort), que fue desarrollado por ARPA en 1990, se definió el **KQML** (Knowledge Query Manipulation Language) (Finin, 1997) [46] como lenguaje de comunicación entre agentes que compartían una sintaxis, denominada KIF (Knowledge Interchange Format) (Genesereth, 1996) [59] y una semántica, denominada Ontolingua.

KIF ofrece semánticas declarativas, equivalentes a la lógica de primer orden, y la habilidad de expresar el conocimiento sobre el conocimiento, y actualmente está siendo desarrollado como un estándar ANSI (American National Standard).

KQML es un lenguaje que los agentes usan para comunicar sus actitudes frente a la información. Estas actitudes (tales como petición, aserción u ofrecimiento) pertenecen al dominio de la teoría del acto del habla, una rama de la lingüística que explora las unidades mínimas del habla (utterances) como acciones que pueden tener éxito o fallar, no sólo como proposiciones que pueden ser ciertas o falsas. KQML es el esfuerzo más extendido para aplicar teoría del habla a la comunicación entre agentes. KQML y KIF son complementarios: KIF expresa el contenido de una proposición, mientras KQML expresa la actitud del agente hacia la proposición. KQML es, por tanto, un lenguaje de comunicación y protocolo, orientado a mensajes, para el intercambio de información. Es independiente de los protocolos de transporte (TCP/IP, HTTP, ...), de la sintaxis de contexto, de las ontologías y de los protocolos de alto nivel (contract net, subasta, ...). Véase Figura 25.

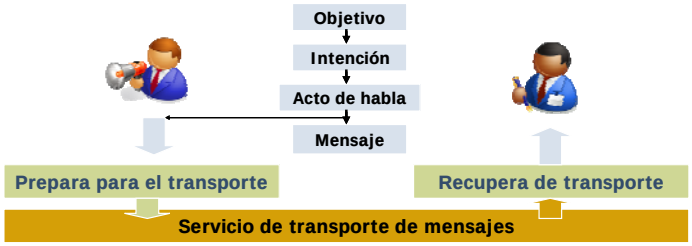

Figura 25. Esquema de comunicación con KQML.

Los mensajes KQML representan un acto de habla o performativas y están formados por una lista de pares atributo-valor como la de la Figura 26.

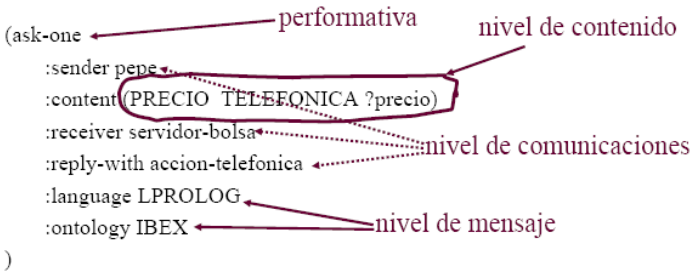

Figura 26. Ejemplo de mensaje KQML (Pavón, 2003) [122].

En la Figura 27 se resumen las distintas performativas disponibles para KQML.

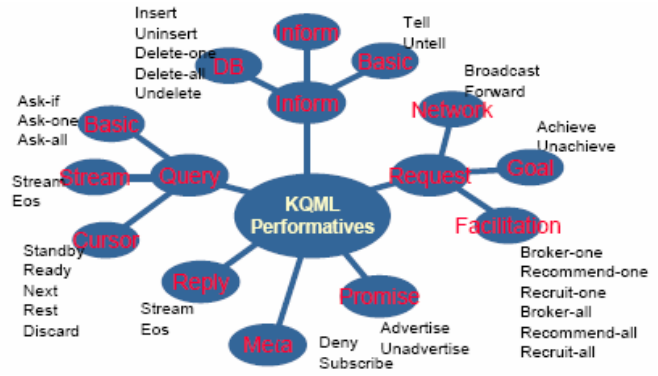

Figura 27. Performativas de KQML (Pavón, 2002) [122].

Ni que decir tiene que no sólo existen estos dos lenguajes de comunicación para agentes, de hecho podemos encontrar amplia literatura sobre diferentes lenguajes y herramientas para agentes, tales como el AGENT0 de Shoham (1993)[157], el lenguaje CONGOLOC (Lésperance et al., 1996) [94], el lenguaje de programación concurrente METATEM (Fisher, 1995) [48], APRIL (McCabe y Clark, 1995) [107], que proporciona al desarrollador un conjunto de herramientas software para la implementación de MAS y una versión modificada de CLISP (C Language Integrated Production System) en la implementación de agente ADEPT de Norman et al. (1997) [118].

Smith (1980) [158] propuso un mecanismo de coordinación muy utilizado en sistemas multiagente y que denominó el protocolo de red de contratos (contract net protocol, CNP). Este protocolo permite una negociación completamente automatizada, resolución de problemas y un mercado electrónico para comprar y vender bienes. El protocolo tiene dos tipos de agentes: iniciador y participante. En cualquier momento, cualquier agente puede ser iniciador, participante o ambos y el protocolo permite contratación y subcontratación.

En el año 2001, Xu y Weigand [184] publicaron una revisión sobre la evolución del CNP donde proponían una arquitectura integrada. Dicha arquitectura toma las ventajas de TRACONET –que extendió a CNP con un proceso de decisión de subasta y acuerdo basado en cálculos de costes marginales–, el marco de trabajo de CIA (Cooperative Information Agent) –que introdujo la noción de obligaciones, que fueron extendidas por CAS (Contractual Agent Societies) para soportar organizaciones flexibles de sociedades de agentes– y el marco de trabajo del TPA (Trading Partner Agreement) –que introducido por IBM, aunque no basado en agentes, usa también contratos y usa un contrato ejecutable en particular–. Al año

siguiente, FIPA (2002) [47] obtuvo una versión del CNP denominada CNIP (Contract Net Interaction Protocol).

En la Figura 28 se presentan temporalmente los distintos estándares que han ido surgiendo y que están relacionados con la computación basada en agentes.

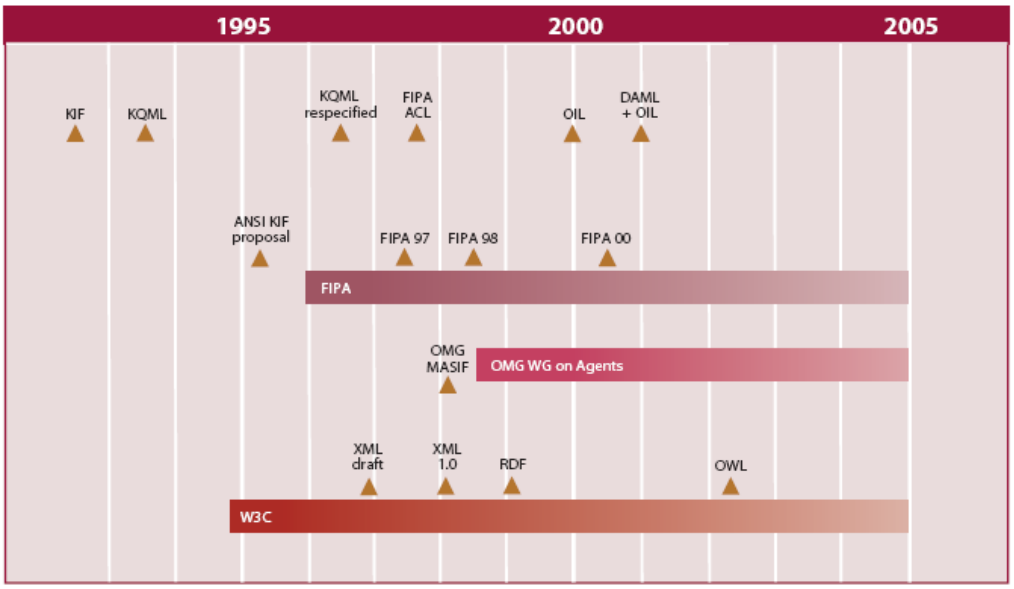

Figura 28. Estándares en la computación basada en agente (AgentLink, 2005) [100].

3.5.1. Semántica de los agentes: ontologías

Las ontologías surgen para resolver los siguientes problemas:

- tener diferentes términos que describen un mismo concepto,
- usar el mismo término para indicar diferentes conceptos, y
- tener definidos diferentes sistemas de clases.

Una ontología común a un conjunto de agentes permite representar el conocimiento de distintos universos de discurso. De modo que los agentes son capaces de entenderse entre sí, desde el punto de vista semántico.

Según (Pérez, 2002) [123], cuando hablamos de ontologías como "sistemas de representación de conocimiento" debemos especificar a qué tipo de sistemas nos referimos. En realidad, las ontologías se están empleando en todo tipo de aplicaciones informáticas en las que sea necesario definir concretamente el conjunto de entidades relevantes en el campo de aplicación determinado, así como las interacciones entre las mismas. Algunas ontologías se crean con el mero objetivo de alcanzar una comprensión del universo de dominio pertinente, ya que su creación impone una especificación muy detallada. Otras ontologías han sido creadas con un propósito general, como por ejemplo el proyecto CyC (Guha y Lenat, 1990) [67], que está orientado a la construcción de una base de conocimiento que contenga el conocimiento humano necesario para hacer inferencias.

Steve et al. (1998) [160] distinguen tres tipos fundamentales de ontologías:

- Ontologías de un dominio, en las que se representa el conocimiento especializado pertinente de un dominio o subdominio, como la medicina y las aplicaciones militares.
- Ontologías genéricas, en las que se representan conceptos generales y fundacionales del conocimiento como las estructuras parte/todo, la cuantificación, los procesos o los tipos de objetos.
- Ontologías representacionales, en las que se especifican las conceptualizaciones que subyacen a los formalismos de representación del conocimiento, por lo que también se denominan meta-ontologías (meta-level o top-level ontologies).

A estos tres tipos, Guarino (1998) [66] añade las ontologías que han sido creadas para una actividad o tarea específica (denominadas task ontologies), como por ejemplo la venta de productos o el diagnóstico de una enfermedad y las ontologías creadas para una aplicación específica.

De entre todas las implementaciones de ontología, podemos destacar las más conocidas:
- OIL
- Ontolingua / KIF
- RDF / Esquemas XML / DTD
- OWL

Protégé 2000 es uno de los sistemas gestores de ontologías más extendidos, así como las plataformas Zeus y JADE, que también dan soporte para el desarrollo de nuevas ontologías.

3.6. Modelos de interacción en sistemas multiagente

La Figura 29 ilustra la estructura típica de un sistema multiagente. El sistema contiene una serie de agentes, que interactúan con otros agentes mediante algún mecanismo de comunicación. Los agentes son capaces de actuar sobre el entorno, de modo que diferentes agentes tienen diferentes esferas de influencia, en el sentido de que los agentes sólo controlan parte del entorno. En ocasiones, estas esferas de influencia pueden coincidir. El hecho de que estas esferas puedan coincidir puede dar lugar a relaciones de dependencia entre distintos agentes.

Cuando nos encontramos frente a lo que parece ser un dominio multiagente, resulta crítico comprender el tipo de interacción que tiene lugar entre los distintos agentes (Wooldridge 2002) [180].

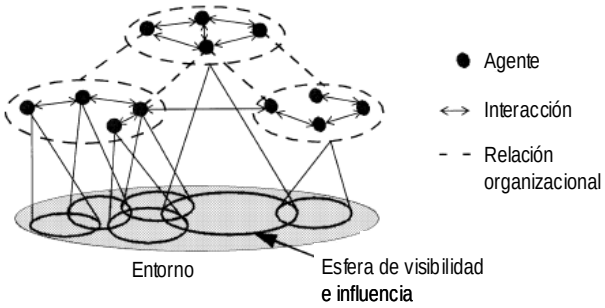

Figura 29. Vista canónica de un sistema basado en agente (Jennings, 2000) [78].

De ese modo, algunos modelos desarrollados recientemente, tal como el de Whitfield et al. (2000; 2002) [176] [26] que proponen una perspectiva estratégica y operativa para sistemas de agentes en el diseño de ingeniería, o Amchurn et al. (2004) [140] que han concebido un modelo de confianza para interacciones multiagente usando confianza y reputación. Otro ejemplo es el trabajo desarrollado por Huynh et al. (2004) [74], denominado FIRE, que describe un modelo de confianza y reputación para sistemas multiagente abiertos.

Para aclarar cómo puede ser la interacción entre agentes, Jennings et al. (1998) [82] distinguen entre modelos cooperativos y modelos de interés personal (self-interested). En el primer tipo de modelo, los agentes cooperan para alcanzar un objetivo común, mientras que en el segundo los agentes negocian para alcanzar su propio objetivo tan bien como sea posible.

Algunos modelos cooperativos son:

- **FA/C (Functionally-Accurate, Cooperative)**: este paradigma aporta un modelo para la descomposición de tareas y la interacción de agentes en un sistema de resolución de problemas distribuido (Lesser, 1991) [93]. En este modelo, los agentes no necesitan tener toda la información localmente para resolver sus sub-problemas, y los agentes interactúan a través del intercambio de co-rutinas asíncrono de los resultados parciales. Permite la posibilidad de que los agentes puedan comportarse de forma no coordinada.

- **PGP (Partial Global Planning)**: este modelo fue propuesto por Durfee y Lesser (1987) [41] y si principio fundamental es que los agentes cooperantes intercambian información para alcanzar conclusiones comunes sobre el proceso de resolución de problemas. La planificación es parcial porque el sistema no genera un plan para el problema completo. Es global porque los agentes forman planes no locales para intercambiar planes locales y cooperar para alcanzar una vista no local de la resolución del problema (Wooldridge, 2002) [180].

- **Joint intentions framework**: este modelo propuesto por Levesque et al. (1990) [96] se centra en la caracterización de un estado mental del equipo, denominado intención colectiva (joint intentions). Un equipo colectivamente tiene la intención de una acción de equipo si los miembros del equipo se comprometen colectivamente a completar la acción de equipo, mientras creen mutuamente lo que estaban haciendo. Un compromiso colectivo es definido como un objetivo persistente colectivo. Para entrar en un compromiso colectivo, todos los miembros del equipo deben establecer las creencias y compromisos mutuos apropiados. Para hacer esto se utiliza un protocolo de compromiso para sincronizar al equipo.

- **SharedPlan**: Grosz y Kraus (1996) [63] elaboraron este modelo basado en la intención de que una acción se ha realizado. Eso concierne a una actividad colectiva de grupo o acciones de un colaborador. Definen un conjunto de axiomas para guiar a un compañero de equipo a tomar una acción, o entrar en una comunicación que permite o facilita a sus compañeros de equipo realizar las tareas asignadas. Hay dos ejemplos de prototipos que usan SharedPlan, que son COLLAGEN (Rich y Sidner, 1997) [144], que se aplica a un agente de interfaz colaborativo que ayuda con la planificación de tráfico aéreo, y GRATE (Jennings, 1993) [76], que se aplica al dominio de la gestión del transporte eléctrico.

- **Arquitectura SOAR**: este modelo fue propuesto primero por Laird, Newell y Rosenbloom (1987) [91] y posteriormente extendido por Newell (1990) [116]. Se trata de una arquitectura cognitiva general para sistemas de desarrollo que exhiben comportamiento inteligente. Newell describió la mente huma como una solución a un conjunto de restricciones funcionales –por ejemplo, exhibe comportamiento adaptativo (orientado a objetivos), usa lenguaje y opera con un cuerpo de varios grados de libertad– y un conjunto de restricciones en construcción –un sistema neural, crecido por procesos embrionarios, surgiendo mediante evolución–. La estructura de SOAR está formada fundamentalmente por tres restricciones funcionales: (a) exhibiendo comportamiento flexible, dirigido a objetivos, (b) aprendiendo continuamente de la experiencia, y (c) exhibiendo conocimiento en tiempo real (el comportamiento cognitivo elemental debe ser evidente en casi un segundo).

La negociación es vista como un método de coordinación y de resolución de conflictos (por ejemplo, resolver objetivos dispares en planificación, resolver restricciones en reserva de recursos, resolver inconsistencias de tareas en la determinación de la estructura organizacional). La negociación también se ha utilizado como una metáfora para la comunicación de cambios de planes, reserva de tareas o resoluciones de violaciones de restricciones centralizadas. De ahí que el término negociación sea casi tan confuso como el de agente. Jennings et al. (1998) [82] dieron lo que se considera actualmente las principales características de la negociación, que son necesarias para el desarrollo de aplicaciones en el mundo real. Estas características son: (a) la presencia de alguna forma de conflicto que debe ser resuelto de forma descentralizada, por (b) agentes con intereses personales, bajo condiciones de (c) racionalidad limitada, e (d) información incompleta. Además, los agentes comunican e intercambian iterativamente propuestas y contrapropuestas.

Algunos modelos de interés personal basado en negociación son:

- **PERSUADER**: fue propuesto por Sycara (1988, 1990) [163] [165]. Se trata de un marco de trabajo para la resolución inteligente de conflictos asistido por computador mediante negociación/mediación. El modelo integra inteligencia artificial y técnicas teóricas de decisión para proporcionar soporte mejorado a la resolución de conflictos y negociación en los parámetros de resolución del problema de grupo. Este modelo ha sido implementado en el PERSUADER (Sycara, 1990) [164], un programa informático que opera en el dominio de las disputas de gestión de trabajo. PERSUADER, actuando como mediador, facilita la resolución del problema de los que disputan de modo que pueda ser alcanzado un consenso sobre un acuerdo. PERSUADER incluye un modelo de negociación general que maneja sistemas multiagente basándose en la integración de razonamiento basado en casos y la teoría de la utilidad multiatributo (Multi-Attribute Utility Theory).

- **Modelo de Rosenschein**: su nombre se debe a su autor, Rosenscehin (1985) [148], y está basado en teoría de juegos. La utilidad es el único aspecto que los agentes consideran, y se suponen omniscientes. Los valores de utilidad para resultados alternativos se representan en un matriz de recompensas que es conocimiento común para ambas partes en la negociación. Cada parte razona y elije la alternativa que maximizará su utilidad. A pesar de la elegancia matemática de la teoría de juegos, los modelos teóricos de juego adolecen de suposiciones restrictivas que limitan su aplicación a problemas reales. Las negociaciones en el mundo real ocurren bajo incertidumbre e implican varios criterios, no sólo la dimensión de utilidad. Según Jennings et al. (1998) [82], las utilidades de los agentes no son conocimiento común sino privado, y los agentes no son omniscientes.

- **Modelo de Kraus**: este modelo fue presentado por Kraus et al. (1995) [88]. Propusieron un modelo de negociación que tiene en cuenta el paso del tiempo durante el mismo proceso de negociación. Introducen un mecanismo de negociación distribuida simple, eficiente, estable y flexible en varias situaciones. El modelo considera situaciones caracterizadas por completo, así como información incompleta y otras en las que algunos agentes pierden tiempo y otros lo ganan. Usando este mecanismo de negociación los agentes autónomos tienen estrategias de negociación simples y estables que resultan en acuerdos eficientes sin retrasos, incluso cuando hay cambios dinámicos en el entorno.

- **Sistema ADEPT (Advanced Decision Environment for Process Tasks):** En [118] Norman, Jennings, Faratin y Mamdani (1997) propusieron una arquitectura multi-agente general para la gestión de procesos de negocio, y un diseño específico de agente implementado dentro de dicho sistema. La arquitectura multi-agente ADEPT está formada por una serie de agencias autónomas. Una agencia está formada por un conjunto, posiblemente vacío, de agencias subsidiarias representadas por un único agente responsable. Así, la arquitectura puede modelar tanto una estructura jerárquica como una plana, o una mezcla de ambas. El agente responsable puede ser abordado por otros agentes autónomos para la provisión de un "servicio" (por ejemplo, alguna unidad de una actividad del proceso de resolución de un problema). Un servicio se corresponde con una tarea atómica dentro del proceso de negocio, o una composición de otros servicios provenientes de varios agentes. Cada agente que actúa autónomamente evaluará constantemente la situación y decidirá cómo asignar los recursos de su agencia, ya sea llamando a servicios de otros agentes basados en un acuerdo anterior, o negociando para alcanzar nuevos acuerdos de nivel de servicio.

- **Modelo de negociación Bazaar**: Zeng y Sycara (1997) presentaron en [190] un modelo de negociación tomando decisiones secuenciales que era capaz de aprender. El objetivo era desarrollar un modelo computacional de negociación que puede manejar aprendizaje multiagente y otros aspectos complicados (como la negociación multicriterio multipropósito) que no tienen modelos analíticos sencillos y eficientes computacionalmente. Zeng y Sycara creyeron que un modelo computacional útil debía tener las siguientes características:
  (1) El modelo debería soportar una manera concisa pero efectiva de representar el contexto de la negociación.
  (2) El modelo debería ser preceptivo por naturaleza.

(3) Los recursos computacionales requeridos para encontrar soluciones o sugerencias razonables deberían ser moderados, a veces a costa de comprometer el rigor del modelo y la optimalidad de las soluciones.
(4) El modelo debería proporcionar medios para modelar la parte dinámica de la negociación.
(5) El modelo debería soportar la capacidad de aprendizaje de los agentes participantes.

Paruchi et al. (2006) [119] consideraron que actualmente, dentro de la comunidad multiagente, es posible distinguir entre al menos cuatro métodos competidores para la construcción de sistemas multiagente actuando en entornos complejos y dinámicos: Creencias-Deseos-Intenciones (Belief-Desire-Intention, BDI), optimización de restricciones distribuidas (Distributed COnstraint OPtimization, DCOP), problemas de decisión de Harkov parcialmente observable distribuidos (Distributed Partially Observable Markov Decision Problems, DPOMDP), y subastas o métodos de teoría de juegos.

En primer lugar, los métodos DCOP se aprovechan localmente de la interacción para buscar un óptimo local o global (Merz et al., 2005 [110]; Alli et al., 2005 [5]; Mailler, 2005 [105]; Scerri et al., 2005 [149]). En Segundo lugar, POMDPs se centra en la coordinación de equipos con la presencia de incertidumbre en acciones y observaciones en el dominio del mundo real (Scerri et al., 2002 [150]; Nair et al., 2003 [115]; Bernstein et al., 2000 [12]). En tercer lugar, las técnicas basadas en subastas y teoría de juegos se concentran en la coordinación entre agentes de interés personal usando mecanismos orientados al mercado (Maheswaran y Basar, 2003) [103], que también pueden ser aplicados en la configuración de equipos. En cuarto lugar, el enfoque BDI, inspirado por el campo de la lógica y la psicología, es un enfoque simbólico, que permite una mejor comprensión humana de la metodología empleada.

3.7. Modelos de programación

La programación basada en agentes supone un paso más en la abstracción de los paradigmas de programación, tal como se esquematiza en la Figura 30.

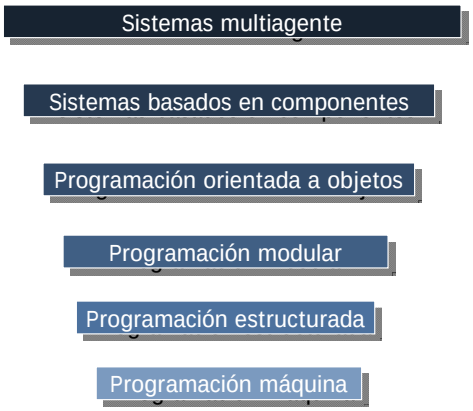

Figura 30. Niveles de abstracción en paradigmas de programación.

Si comparamos el paradigma orientado a agentes con el orientado a objetos, podemos observar varias diferencias fundamentales entre un agente y un objeto (véase Tabla 12), de las cuales destacamos las tres siguientes:

- Un objeto que publica un método no puede controlar o decidir si ese método es ejecutado o no por otro objeto, mientras que un agente sí decide si una solicitud recibida es ejecutada o no.
- Un objeto no integra un comportamiento autónomo flexible con respecto a las características de reactividad, pro-actividad y de habilidades sociales.
- Un sistema multiagente es multi-hilo por definición, en el sentido de que cada agente toma un hilo de control independiente, mientras que en la programación orientada a objetos hay un único hilo de control y un único punto de entrada en la ejecución.

Agente	Objeto
Autonomía de decisión	Ejecuta métodos invocados
Flujo de control propio	Flujo de control del llamante
Encapsula activación del comportamiento	Encapsula estado y comportamiento
Estado mental: objetivos, creencias...	Estado: valor de variables
Comportamiento: cómo decidir lo que hacer	Comportamiento: salida a partir de una entrada
Interacciones: actos de habla (intencionalidad)	Mensajes invocan procedimiento
Organización: relaciones sociales entre agentes	Asociaciones entre objetos

Tabla 12. Agente vs. objeto.

Asimismo, existen otras diferencias destacables entre ambos paradigmas de programación, que se recogen en la Tabla 13 a modo de resumen.

Aspecto	POA	POO
Unidad básica	Agente	Objeto
Parámetros que definen el estado de la unidad básica	Creencias, obligaciones, capacidades, selecciones	Sin restricciones
Proceso de cómputo	Métodos de paso de mensajes y de respuestas	
Tipos de mensajes	Informes, requerimientos, ofertas, promesas...	Sin restricciones
Restricciones sobre los métodos	Honestidad, consistencia...	Ninguno

Tabla 13. Programación orientada a objeto vs. programación orientada a agentes.

En 1999 Wooldridge afirmó lo siguiente: "*Objects do it for free, agents do it for money*." Ese mismo año Parunak definió a un agente como un objeto con las siguientes características: tiene iniciativa, posee actitud u orientación, puede decir "no" (o "adelante") y es proactivo.

Tal como ilustra la Figura 31, inspirada en una propuesta de ANA MAS (2005) [6], podemos decir que existe una evolución desde el concepto de objeto hasta el concepto de sistema multiagente, y que además esa evolución ha sido paralela a la evolución tecnológica que se ha vivido a lo largo de las dos últimas décadas.

Figura 31. Evolución Objeto – Agente (ANA MAS, 2005) [6].

Sin entrar en detalles, podemos decir que la ingeniería del software orientada a agentes distingue cinco etapas fundamentales de desarrollo (véase Figura 32):

1. La definición del problema
2. La representación del conocimiento
3. El diseño de los agentes (véase Figura 33)
    a. Diseño de la base de conocimiento
    b. Diseño de la máquina de inferencia
    c. Diseño del interfaz
    d. Diseño del lenguaje de comunicación
4. La implantación de los agentes (véase Figura 34)
    a. Análisis y selección de herramientas
    b. Implantación de la base de conocimiento
    c. Implantación de la máquina de inferencia
    d. Implantación de la interfaz
    e. Implantación de la comunicación
5. Las pruebas de los agentes

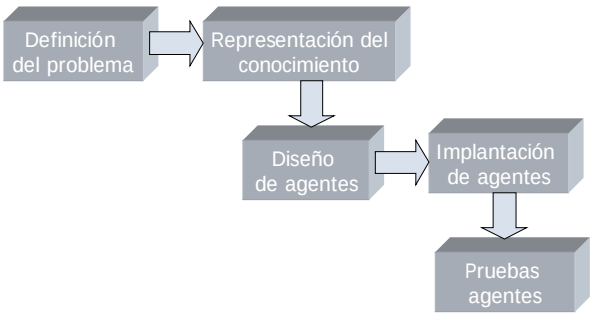

Figura 32. Fases de la ingeniería del software orientada a agentes.

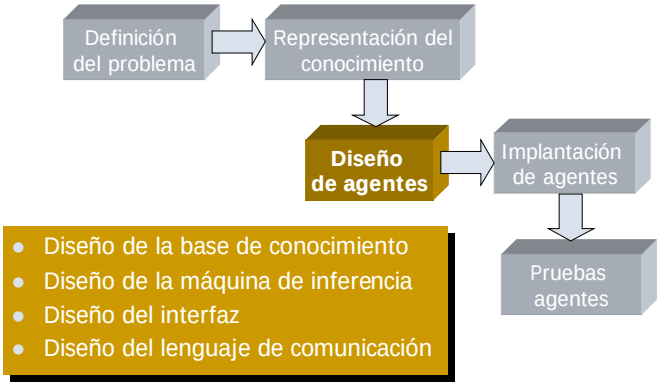

Figura 33. Diseño de agentes.

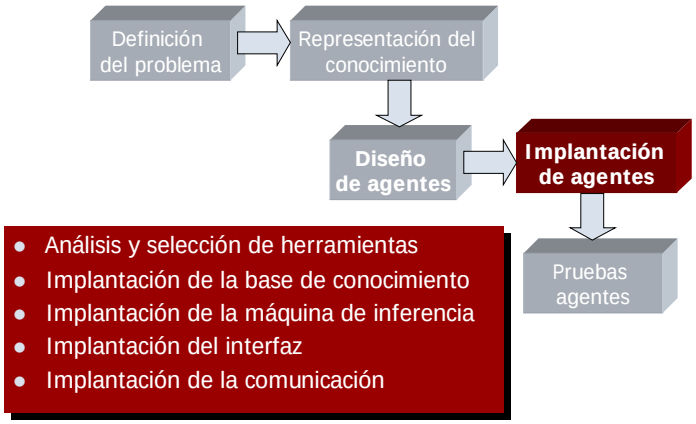

Figura 34. Implantación de agentes.

En la Figura 35 se ilustra cuál es el esquema general de un agente software, en el que se considera la percepción del entorno (entrada del agente) y la acción que el agente realiza sobre el entorno (salida del agente) que vimos en la Figura 10 (página 25).

```
Programa
  Función AGENTE-ESQUELETO ()
    estática: memoria;
    memoria ← ACTUALIZA (memoria, percepción)
    acción ← ESCOGE-ACCIÓN (memoria)
    memoria ← ACTUALIZA (memoria, acción)
    devuelve (acción)
  Fin AGENTE-ESQUELETO()
```

Figura 35. Estructura general de un agente software.

3.8. Herramientas de programación orientada a agentes

El objetivo de este apartado es indicar, de forma esquemática, cuáles son las herramientas de programación orientada a agentes más conocidas:

- **Lenguajes de programación**:
  - Java
    - Javalog; basado en programación lógica.
    - Jess, ILOG Jrules; basados en reglas de producción.
  - Prolog
    - Ciao-Prolog, Prolog-Café
- Plataformas de desarrollo:
  - JADE (Java Agent DEvelopment Framework) de Telecom Italia Lab (TILab).
  - FIPA-OS (FIPA Open Source) de Nortel Networks.
  - ABLE (Agent Building and Learning Environment) de IBM.
  - Jackal del Consortium for Intelligent Integrated Manufacturing Planning-Execution(CIIMPLEX).
  - OAA (Open Agent Architecture) de SRI International.
  - JatLite (Java Agent Template Lite) de la Universidad de Stanford.
  - Aglets de IBM; para agentes móviles.
  - Grasshopper de IKV++; para agentes móviles.
  - AgentBuilder de Reticular Systems.
- **Entornos de desarrollo**:
  - Zeus de British Telecom (BT).
  - AgentBuilder de Acronymics Incorporation.
- **Lenguaje de descripción de agentes**:
  - AUML de FIPA.
- **Metodologías de desarrollo**:
  - AAII (1997) [87], basada en la arquitectura BDI.
  - MAS-CommonKADS (1998) [151], que considera 6+1 modelos: Organización, Tareas, Agente, Comunicaciones, Experiencia, Diseño, Coordinación. Se trata del CommonKADS (1994) extendido con orientación a objetos, con SDL (Specification and Description Language) y MSC (Message Sequence Chart).
  - Gaia (2000) [179], que considera un MAS como conjunto de entidades que interactúan.
  - MaSE (2001) [35], que se basa en orientación a objetos con conversaciones entre objetos.
  - MESSAGE (2001) [44], INGENIAS (2003) [60] [139], que son meta-modelos.

3.8.1. JADE

La plataforma JADE es la implementación del estándar FIPA más extendida hoy día para el desarrollo de agentes, y es el resultado de la combinación de dos productos:

– Una plataforma de agentes FIPA-Compliant.
– Una herramienta de desarrollo de agentes Java.

Esta plataforma proporciona una arquitectura (véase Figura 36) para ejecutar agentes que soporta:

– Un modelo de programación de agentes asíncrono.
– La comunicación entre agentes: en la misma y en diferentes plataformas.
– La movilidad de agentes, la seguridad en la comunicación y los accesos a los recursos y otras utilidades.

En la Figura 36, vemos que sobre la plataforma JADE se dispone de una capa homogénea, a través de la cual cualquier agente del MAS puede comunicarse con la plataforma para acabar ejecutándose sobre una máquina virtual Java, que dependiendo de la aplicación, será J2SE (Java 2 Platform, Standard Edition), J2EE (Java 2 Platform, Enterprise Edition) o J2ME(Java 2 Platform, Micro Edition).

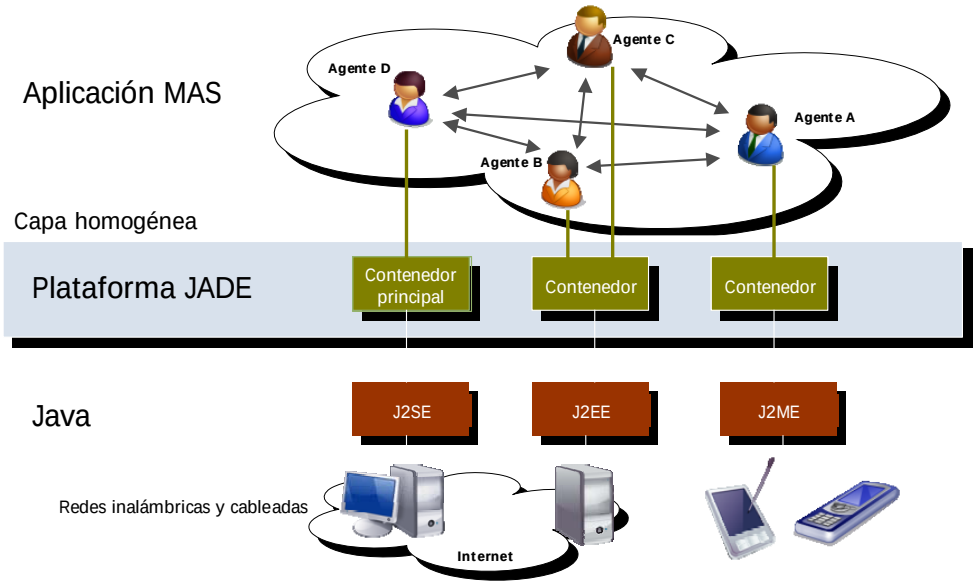

Figura 36. Arquitectura JADE.

Para implementar un agente en JADE primero tiene que heredar de la clase `jade.core.Agent`. Esta clase proporciona métodos para iniciar el agente en la plataforma con un nombre predeterminado, así como enviar y recibir mensajes de otros agentes, desde la misma u otras plataformas conformes al estándar de FIPA.

El contenido de los mensajes se describe con el lenguaje SL-0. Además, se pueden definir ontologías, para enviar mensajes de acuerdo con un lenguaje y ontología particulares. A la hora de crear una ontología, se crea automáticamente un codificador/decodificador del

contenido de los mensajes según esa ontología específica. De este modo, el implementador no tiene que preocuparse de cómo convertir a texto una determinada información o cómo analizar el mensaje. JADE se encarga de crear las clases Java con la información, y al definir la ontología se indican las clases que contienen la información que se desea enviar y recibir, para que la plataforma cree los codificadores/decodificadores apropiados.

JADE también proporciona implementaciones de los protocolos estándares identificados en FIPA. Estos protocolos se definen en función de mensajes asíncronos. Se espera que los agentes utilicen estos protocolos para preguntar a otros y obtener información, para negociar, para pedirles que hagan algo, es decir, para cualquier interacción que requiera comunicación entre los agentes. El uso de estos protocolos permite la reutilización de gran cantidad de código en el desarrollo de agentes.

Para integrar todo esto con los distintos comportamientos que puede requerir un agente, la plataforma JADE proporciona una arquitectura de agente básica. El acceso a los recursos y servicios de la plataforma se hace a través de código encapsulado en clases predefinidas de JADE, `jade.core.Behaviour`, que proporcionan el método `action()` para describir la tarea asignada a cada comportamiento. Estos comportamientos permiten enviar o recibir mensajes y organizar las actividades del agente en bloques de código que se ejecutarían de forma secuencial o no determinista, entre otros. La idea es que todo el comportamiento del agente se codifique con estas clases. Sin embargo, muchos desarrolladores optan por construir sobre estos comportamientos otras capas, como una capa Jess que permita expresar más claramente la implementación de la conducta del agente.

3.9. Tecnologías para interacción en entornos abiertos

De forma resumida, podemos introducir que las principales tecnologías que se utilizan para la interacción con entornos abiertos son las siguientes:

- XML: aporta estructuración de la información.
- Java y RMI: permiten el desarrollo de MAS y la interoperabilidad entre agentes heterogéneos siempre y cuando se encuentren implementados en Java.
- CORBA: permite la independencia de las plataformas, posibilitando el intercambio de mensajes XML entre agentes desarrollados en cualquier lenguaje de programación para el que se haya definido una correspondencia con el IDL (Interface description language).

CORBA (OMG, 2004) [128], la Common Object Request Broker Architecture, define un mecanismo estándar por el que los objetos escritos en diferentes lenguajes y ejecutándose en un entorno distribuido puede hacer solicitudes a otros objetos y, a su vez, responder solicitudes de otros.

Generalmente, los MAS están caracterizados como extensiones de sistemas orientados a objetos. Con frecuencia, esta visión tan simplificada ha traído problemas a los diseñadores de sistemas cuando tratan de capturar características que son exclusivas de los MAS usando herramientas orientadas a objeto. Como respuesta a este problema, FIPA (2003) está desarrollando un lenguaje de modelado unificado basado en agente (Agent-based Unified Modeling Language, AUML), que ya fue descrito por Bauer et al. (2001) [11].

Java [161] de Sun es un mecanismo estándar para permitir que el comportamiento del agente viaje desde un procesador a otro usando Aglets. Aglets [2] es una plataforma de agentes móviles basados en Java y librerías para construir aplicaciones basadas en agentes móviles. Un aglet es un agente Java que puede moverse de forma autónoma y espontáneamente desde una máquina a otra llevando un fragmento de código consigo. Un aglet puede ser programado para ejecutarse en una máquina remota y muestra comportamientos diferentes en diferentes máquinas. Las implementaciones de seguridad basadas en Java cuidan los accesos autorizados a recursos locales en la máquina remota. Aglets está completamente escrito en Java, permitiendo así una alta portabilidad tanto de los agentes como de la plataforma. Incluye una plataforma agente móvil Java completa, con un servidor autónomo denominado Tahiti, y una librería que permite a los desarrolladores construir agentes móviles y embeber la tecnología Aglets en sus aplicaciones.

Por otro lado, tenemos Jini [83]. Se trata de una arquitectura de red para la construcción de sistemas distribuidos donde la escalabilidad, la velocidad de cambio y la complejidad de las interacciones inter y entre redes son extremadamente importantes y no pueden resolverse satisfactoriamente con tecnologías precedentes. La tecnología Jini proporciona una infraestructura flexible para la entrega de servicios en una red y para la creación de interacciones espontáneas entre clientes que usan estos servicios independientemente de sus implementaciones hardware o software.

La tecnología de red Jini es una arquitectura abierta que permite a los desarrolladores crear servicios de red centralizados –ya sea implementados en hardware o en software–, que son muy adaptables a cambios. Esta tecnología puede usarse para construir redes adaptativas

que son escalables, evolutivas y flexibles como suele ser necesario en entornos de computación dinámicos.

Esta arquitectura especifica un método para que clientes y servicios puedan encontrarse unos a otros en la red y trabajar juntos para la realización de una tarea. Los proveedores de servicio suministran clientes con objetos basados en tecnología Java portable que dan a los clientes acceso al servicio. Esta interacción de red puede utilizar cualquier tipo de tecnología de red, tal como RMI [75], CORBA [128], o SOAP [182], porque el cliente sólo ve el objeto Java proporcionado por el servicio y, en consecuencia, toda la comunicación de red es confinada en ese objeto Java y el servicio de donde vino.

Cuando un servicio se une a una red de servicios y dispositivos con tecnología Jini, se anuncia a sí mismo publicando un objeto Java que implementa la API del servicio. Esta implementación del objeto puede funcionar en cualquier forma que el servicio elija. El cliente encuentra los servicios buscando un objeto que soporta el API. Cuando consiga el objeto publicado del servicio, descargará cualquier código que necesite para poder hablar con el servicio, aprendiendo, por tanto, cómo hablar con la implementación particular de ese servicio mediante el API. El programador que implemente el servicio, elige cómo traducir una solicitud del API en bits usando RMI, CORBA, XML [183], o un protocolo privado.

La existencia de la plataforma Java hace posible definir la tecnología de red Jini que, a su vez, mejora el valor de la plataforma Java haciendo disponibles los servicios a través de la red. La plataforma Java especifica qué hay disponible en una máquina concreta que esté ejecutando la plataforma. Define un conjunto de servicios (clases y la máquina virtual de Java) que existe en una máquina concreta que puede ser utilizada por los programas que están ejecutándose en esa máquina. La tecnología Jini extiende esta noción de plataforma desde una máquina particular en la red hasta la propia red que conecta las máquinas que están ejecutando la plataforma Java. Los servicios habilitados con tecnología Jini no tienen por qué ser necesariamente residentes en una máquina particular de la red, de hecho deben estar disponibles para todas las máquinas a través de la red.

Por otro lado, algunos ingenieros industriales han desarrollado sus propias convenciones para la especificación e implementación de sistemas, y en su ámbito aceptarán con más facilidad tecnologías de agente si un sistema de agente soporta estas convenciones. Por ejemplo, Grafcet (Bierel y Roussel, 1995) [13] es un estándar internacional (IEC 848) para un lenguaje de control gráfico basado en redes de Petri. Las redes de Petri suponen un potente mecanismo para representar la lógica interna de un agente (Ferber, 1995) [45], y Grafcet disfruta de un uso industrial extendido como representación la lógica de control, haciéndolo buen candidato para la implementación de agentes industriales.

También podemos mencionar otra tecnología relacionada con las comunicaciones entre agentes. Esta tecnología, denominada JXTA [84][162], es una plataforma punto-a-punto de código abierto creada por Sun Microsystems en el año 2001. Esta plataforma se define como un conjunto de protocolos basados en XML que permiten que cualquier dispositivo conectado a una red pueda intercambiar mensajes y colaborar independientemente de la topología de red. Actualmente, JXTA es el marco de trabajo P2P más maduro disponible y fue diseñado para permitir la comunicación de gran variedad de dispositivos –PCs, mainframes, teléfonos móviles, PDAs– de forma centralizada.

Dado que JXTA está basado en un conjunto de protocolos abiertos, puede ser portado virtualmente a cualquier lenguaje de programación moderno y estas implementaciones son denominadas "bindings". El *binding* Java, JXTA2SE, es la implementación más madura hasta ahora, pero también se está desarrollando una versión en C, JXTA-C.

Los pares JXTA crean una red *overlay*[5] virtual que permite a un extremo interactuar con otro directamente, incluso cuando alguno de los extremos y recursos están tras algún cortafuegos y NATs o usando diferentes transportes de redes.

En la Figura 37 se muestran las diferentes tecnologías que se han ido desarrollando a largo de los últimos quince años y que están relacionadas en mayor o menor medida con agentes para el soporte de infraestructuras que interoperan con sistemas abiertos.

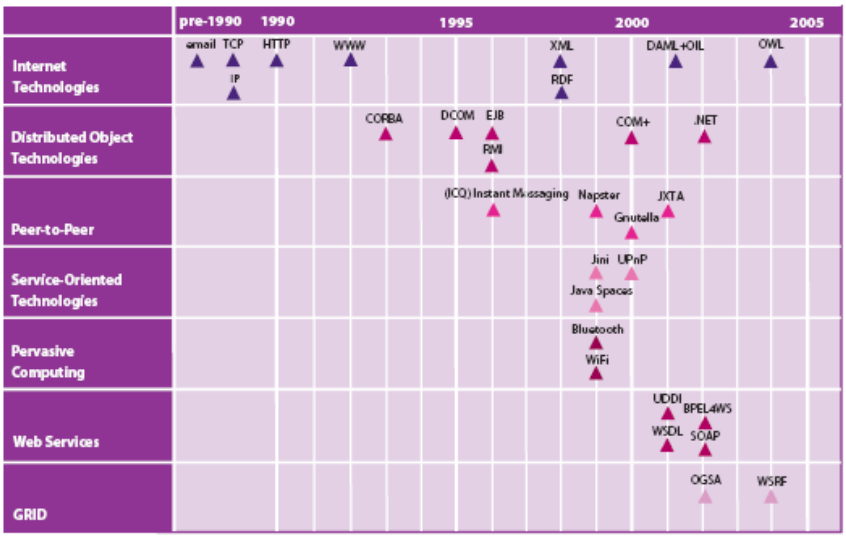

Figura 37. Tecnologías relacionadas con agentes para el soporte de infraestructuras (AgentLink, 2005) [100].

[5] Una red *overlay* es aquella que es construida sobre otra red.

4. Aplicación de MAS en el proyecto HIPARSYS

Como hemos comentado en las primeras secciones de este documento, el principal objetivo del proyecto HIPARSYS es el modelado del comportamiento de los diseñadores para obtener un diseñador automático aplicando el conocimiento adquirido por los diseñadores en entornos industriales. Este proyecto está especialmente centrado en los trabajos de diseño realizados por Jaguar y Rolls Royce en la industria automovilística.

Hemos estudiado que existen una serie de tecnologías en el sector del diseño industrial que se emplean en proyectos de diseño de diferente complejidad. Aunque estas metodologías no suelen utilizarse para modelar el comportamiento del diseñador en un diseño específico, sino para modelar los diseños en sí mismos, pueden ayudar a perfilar cómo es el procedimiento de trabajo de los diseñadores desde un punto de vista general. Para conocer este comportamiento desde el punto de vista de Jaguar y Rolls Royce, es necesario adquirir información directamente de los diseñadores que están trabajando en ambas empresas.

La obtención de esa información es trabajo de la Universidad de Sheffield, que también participa en el proyecto y es responsable de entrevistar a los diseñadores y extraer toda la información necesaria para modelar su comportamiento durante el proceso de diseño.

La información necesaria no sólo incluye los pasos que los diseñadores realizan durante el proceso de diseño, sino también las interacciones entre los diferentes agentes. Es en este punto donde las ciencias sociales entran en juego. Las ciencias sociales pueden ser aplicadas en el modelado y simulación de sistemas multiagente. Davidsson (2002) [31] describe un punto de vista informático de la simulación social basada en agente (Agent Based Social Simulation, ABSS) cuyas interacciones se muestran en la Figura 38.

Figura 38. Las intersecciones entre las tres áreas que definen ABSS (Davidsson, 2002) [31].

Dignum et al. (2002) [36] proponen un marco de trabajo y una metodología para el diseño de sociedades de agentes[6]. Distinguen tres tipos de modelos: mercado, red y jerárquico. Las características de cada uno de ellos se describen en la Tabla 14.

[6] Con idea de resaltar los aspectos sociales de la interacción de agentes, Digmun et al. usan el término sociedad se agente en vez de sistema multiagente..

Tabla 14. Características de diferentes marcos de trabajo (Dignum et al., 2002) [36].

	Mercado	**Red**	**Jerarquía**
Coordinación	Mecanismo de precios	Colaboración	Supervisión
Forma de relación	Competición	Interés mutuo	Autoridad
Medios primarios de comunicación	Precios	Relaciones	Rutinas
Tono o clima	Precisión/ Sospecha	Abierta/ Beneficios mutuos	Formal/ Burocracia
Resolución de conflictos	Regateo (recurso ante tribunal)	Reciprocidad (Reputación)	Supervisión
Tipo de sociedad	Abierto	Confianza	Cerrada
"Valores" de agentes	Autointerés	Interés mutuo/ Colaboración	Dependencia
Roles de facilitación	*Matchmaking* Banca	Con guardianes Registro Matchmaking	Control de interfaz
Propósito de la sociedad	Intercambio	Colaboración	Producción
Objetivos	Objetivos individuales (determinados por el agente)	Ambos son posibles	Determinado por los objetivos globales de la sociedad
Forma de relaciones	Negotiación (p.e. Contract Net Protocol)	Negociar dentro de las normas y reglas sociales	Fijado (p.e. bucle de acción/flujo de trabajo)
Capacidades de comunicación de los agentes	Interacción basada en estándares; comunicación sólo concierne al intercambio	Tanto los procedimientos de de interacción como los intercambios pueden ser negociados	Especificado en diseño
Interfaz para el mundo exterior	Generalmente abierto para agentes (después de la identificación)	Procedimiento de admisión para agentes	Cerrado para agentes; abierto para datos (entrada y salida)

De acuerdo con Dignum et al (2002) [36], el marco de trabajo jerárquico es el más adecuado en entornos industriales y de fabricación, que es precisamente el caso que nos ocupa.

En el modelo jerárquico de sistemas de información, cada agente de información es responsable de proporcionar información sobre un dominio específico. Los agentes de información más bajos de la jerarquía proporcionan información más especializada sobre un determinado dominio. En respuesta a una petición, un agente de información puede cooperar con agentes de información de otros dominios o sub-dominios para obtener una respuesta. Las

soluciones de la red de comunicaciones están basadas en una jerarquía de agentes inteligentes autónomos, que tienen capacidades locales para tomar decisiones, pero cooperan para resolver conflictos. Los agentes de más alto nivel arbitran las disputas entre agentes pares.

La metodología para el desarrollo de sociedades de agente basada en este marco de trabajo está formada por varios niveles (Figura 39):

- Diseñar el modelo de coordinación.
- Definir el entorno en términos de los requisitos globales y la ontología del dominio en cuestión.
- Describir el comportamiento en término de los roles de los agentes y los patrones de interacciones.
- Definir la estructura interna de agentes en términos de cuáles son los requisitos para la comunicación, acción, interfaz y el comportamiento del razonamiento.

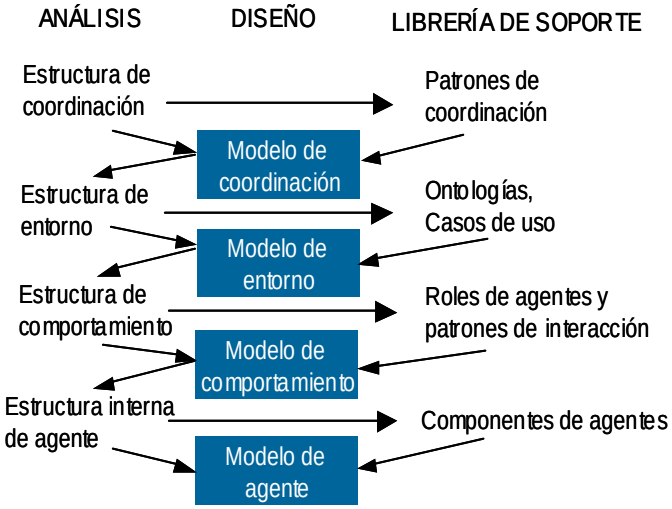

Figura 39. Metodología de desarrollo para sociedades de agentes (Dignum et al, 2002) [36].

Por otro lado, Norman et al. (1997) [118] proponen una arquitectura para la gestión del proceso de negocio que puede ser aplicada a cualquier arquitectura social jerárquica en la es necesario que las interacciones entre los distintos agentes esté organizada de alguna forma específica. El esquema general se ilustra en la Figura 40.

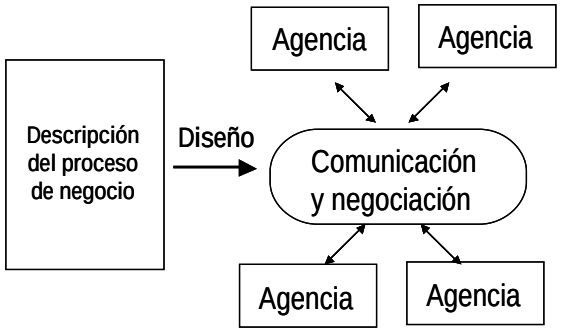

Figura 40. Diseño de un sistema de gestión de procesos de negocio basado en agentes (Norman et al, 1997) [118].

Norman et al. proponen una jerarquía lógica de agencias para representar las interacciones jerárquicas entre humanos en un entorno de negocio. Dependiendo de la estructura y de la interacción entre los diseñadores en Jaguar y Rolls Royce, es posible modelar el comportamiento de los diseñadores y posteriormente simularlo.

La arquitectura que proponen está formada por una serie de elementos:
- Agencia
- Agentes responsables
- Tareas
- Sub-agencias

Una agencia se define recursivamente: una agencia está formada por un único agente responsable (o controlador), un conjunto de tareas –que puede ser vacío– que el agente responsable puede realizar, y un conjunto de sub-agencias –que también puede ser vacío– (ver Figura 41). El agente responsable representa los intereses de la agencia de cara a sus homólogos[7]. Cualquier comunicación con una agencia debe ir a través del agente responsable. Una sub-agencia (por ejemplo, la agencia G es una sub-agencia de la agencia Y, Figura 41) se comporta típicamente de modo cooperativo hacia su agente responsable, este agente es el responsable de representar los intereses de la agencia en la comunidad más amplia. Esta relación entre la sub-agencia y el agente responsable puede verse como un tipo de compromiso social, y proporciona un mecanismo de encapsulación y abstracción de servicios.

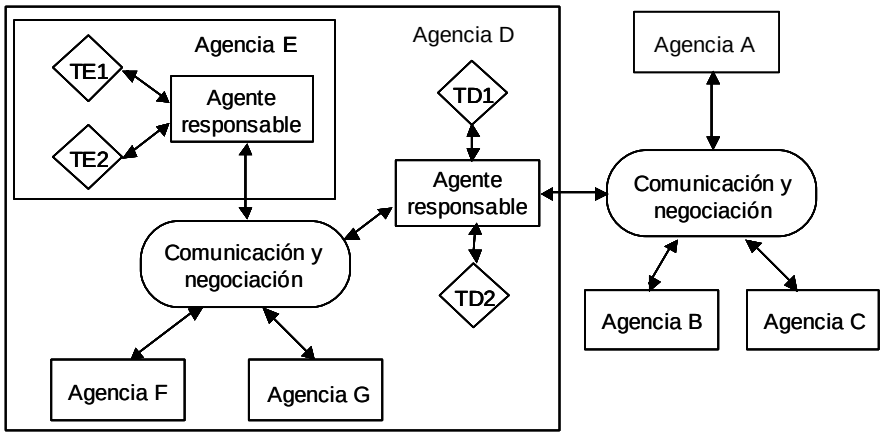

Figura 41. La jerarquía lógica de agentes (Norman et al., 1997) [118].

En nuestro caso, podemos adoptar esta arquitectura para modelar la posible jerarquía entre los diseñadores de Jaguar y Rolls Royce.

Tal como comentamos en el apartado de tecnologías de modelado, también existen otros modelos para el modelado de equipos de trabajo y las interacciones entre sus miembros. Estos modelos pueden ser aplicados al trabajo de los diseñadores y, por tanto, al comportamiento de los mismos. Algunos de estos modelos son:

[7] Las agencias homólogas son aquellas que tienen agentes responsables capaces de comunicar sin cruzar el límite de la agencia.

✓ *Basado en intenciones colectivas (joint intentions) de Cohen y Levesque (1990, 1991) [96] [27]:*

Presentaron un modelo creencia-objetivo-compromiso de los estados mentales de los individuos en los que las intenciones se especifican no como características mentales primitivas, sino como compromisos internos de realizar una acción mientras se mantiene un cierto estado mental. Su noción de compromiso, por el contrario, fue especificada como un objetivo que persiste en el tiempo. Una cuestión primaria de este estudio fue investigar de qué modo un equipo es similar a un agente agregado, y extender su trabajo previo en intenciones individuales que pueden ser transportados a la intención colectiva. A partir de ahí, continúan con su anterior desarrollo y dan argumentos acerca de qué consideran una intención colectiva, que fue formulada como un compromiso colectivo para realizar una acción colectiva mientras se comparta un cierto estado mental, como el punto de unión de los miembros de un mismo equipo.

Para alcanzar un cierto grado de realismo requerido para el comportamiento autónomo satisfactorio, modelaron los agentes individuales como si estuvieran situados en un mundo dinámico, multi-agente, sin creencias completas ni correctas, con objetivos cambiantes, acciones que fallaban, y sujeto a interrupción por eventos externos. Además, asumen que las creencias y objetivos de los agentes no necesitan ser conocidos por otros agentes, y que incluso si los agentes empiezan con un estado donde ciertas creencias y objetivos son compartidos, esta situación puede cambiar con el paso del tiempo.

✓ *Basado en SharedPlans de la teoría de Grosz (1996, 1999) [63][64]:*

El modelo propuesto proporciona una especificación de las capacidades para actuar y actitudes mentales que los agentes individuales deben tener para participar en actividades colaborativas con cualquier otro agente. Además, esta teoría proporciona especificaciones de planes para acciones individuales que se modifican a partir de acciones previas para adaptarlas a una actividad colaborativa. Las especificaciones son normativas y se entiende que proporcionan la base para construir agentes que actúan de manera racional. Aunque este trabajo estuvo basado en un análisis de colaboraciones humanas, no consideraron necesario considerar una descripción completa del comportamiento humano colaborativo. Sin embargo, el modelo ha sido utilizado para explicar una serie de diálogos en lenguaje natural.

Adoptaron una visión del estado mental de los planes, es decir, los agentes tienen planes cuanto tienen un conjunto particular de intenciones y creencias. Distinguieron planes individuales, que están formados por agentes individuales, de los planes compartidos (SharedPlans), que se construyen con grupos de agentes que están colaborando. Cuando los agentes tienen un plan compartido para llevar a cabo una acción grupal, poseen ciertas creencias individuales y mutuas acerca de cómo ha de realizarse la acción y sus subacciones constituyentes. Cada agente puede tener intenciones y planes individuales para realizar algunas de las subacciones. Los agentes también tienen intenciones individuales para la realización satisfactoria de sus acciones, tanto individuales como grupales.

Usaron lógica de primer orden extendida con varios operadores modales, metapredicados y expresiones de acciones.

✓ *Basado en TEAMCORE de Tambe et al. (1999, 2000) [126] [168]:*

Tambe et al. presentaron un nuevo enfoque de dos niveles para solucionar este problema de coordinación. Proporcionaron primero la arquitectura de integración con capacidades de coordinación de equipos de trabajo de propósito general, pero permitieron la adaptación de

tales capacidades a las necesidades o requisitos de los individuos específicos. Un aspecto nuevo clave de esta adaptación es que tiene lugar en el contexto de otros miembros de equipo heterogéneos. Desarrollaron este enfoque en una arquitectura de integración de agentes distribuida denominada Teamcore. Presentaron resultados experimentales de dos dominios diferentes (Tambe et al, 2000) [168].

Su enfoque para solucionar este problema fue proporcionar la arquitectura de integración con capacidades de coordinación en equipos de trabajo de propósito general y, después, adaptar tales capacidades (vía aprendizaje máquina) para las necesidades y rendimiento de individuos específicos. El conocimiento de equipo de trabajo general evita la necesidad de escribir grandes cantidades de planes de coordinación para cada nuevo dominio y agente. La adaptación basada en este fundamento permite que la arquitectura de integración satisfaga las necesidades de coordinación y rendimiento individuales. Aquí la institución del conocimiento del grupo de trabajo es crítico para la adaptación, ya que aprender todo el conocimiento de la coordinación a partir de cero para cada caso resultaría muy costoso.

- ✓ *Basado en el problema de decisión de equipos multiagente comunicativos (COM-municative Multiagent Team Decision Problem, COM-MTDP) de Pynadath et al (2002) [125]:*

Estos autores proponían un marco de trabajo unificado donde se suministraba una herramienta para el uso de investigadores multi-agente en la evaluación del seguimiento de la optimalidad-complejidad de equipos de trabajo, el COM-municative Multiagent Team Decision Problem (COM-MTDP). El modelo COM-MTDP combina y extiende teorías multi-agente existentes, tales como los procesos de decisión de Markov observables parcialmente y descentralizados y la teoría económica de equipos. Además de su generalidad de representación, este modelo da soporte al análisis tanto de la optimalidad del rendimiento de equipos como de la complejidad de los problemas de decisión de los agentes. Analizando la complejidad, presentaron un desglose de la complejidad computacional construyendo equipos óptimos bajo varias clases de dominios de problemas, mediante las dimensiones de observabilidad y coste de la comunicación. Analizando la optimalidad, explotaron la habilidad de COM-MTDP para codificar teorías de grupos de trabajo existentes y modelos para codificar las dos instancias de la teoría de intenciones colectivas tomadas de la literatura. Además, el modelo COM-MTDP proporciona la base para el desarrollo de nuevos algoritmos de coordinación de equipos. Obtuvieron un criterio independiente del dominio para comunicaciones óptimas y proporcionaron un análisis comparativo de las dos instancias de intenciones colaborativas con respecto a esta política óptima. Implementaron una paquete software reutilizable e independiente del dominio basado en COM-MTDP para analizar las estrategias de coordinación de trabajo en equipo, y demostraron su uso codificando y evaluando las dos estrategias de intenciones colaborativas dentro de un dominio ejemplo.

- ✓ *Basado en los enfoques híbridos de Paruchi et al. (2006) [119]:*

Paruchi et al. presentaron una revisión de las investigaciones recientes que incluyen optimización de restricciones distribuidas (Distributed Constraint Optimization, DCOP) y problemas de decisión de Markov observables parcialmente distribuidos (Partially Observable Markov Decision Problems, POMDP) en la construcción de equipos de agentes. Mientras que los algoritmos DCOP y POMDP distribuido proporcionan resultados prometedores, los enfoques híbridos permiten usar longitudes complementarias de diferentes técnicas para crear algoritmos que funcionen mejor que cualquiera de sus algoritmos constituyentes de forma independiente. Por ejemplo, en el enfoque híbrido DBI-POMDP, los planes de los equipos

DBI se aprovechan para mejorar la adaptabilidad del POMDP, y los algoritmos POMDP mejoran el rendimiento de los planes de equipos BDI.

Por otro lado, para el modelo de procesos que tiene que realizar el equipo de trabajo de la Universidad de Southampton, es posible elegir entre una serie de metodologías de diseño tradicionales[8]. Por ejemplo, es posible utilizar el CommonKADS (Schreiber, 2000) [152] o la metodología DRED (Bracewell et al., 2004) [16], que han sido descritas brevemente en apartados anteriores, para diseñar la base de conocimiento que la Universidad de Sheffield tiene que extraer de los diseñadores de Jaguar y Rolls Royce.

Cuando esté disponible la información extraída de los diseñadores, tendremos que encontrar qué métricas se pueden usar para modelar el comportamiento de los diseñadores (tiempo, coste, complejidad, motivación, etc.).

Cuando se haya obtenido el modelo de comportamiento de los diseñadores, ya sea usando una metodología del ámbito industrial, o mediante el entorno de sistemas basados en agentes, sería apropiado llevar a cabo una simulación del modelo en un proceso de diseño específico y observar cómo funciona y cómo de fiable resulta, introduciendo métricas que dependerán de la información extraída de los diseñadores.

Una de las posibilidades barajadas para llevar a cabo las simulaciones en la simulación orientada a agentes (Ören, 2002) [137]. Los agentes software se encuentra en un cierto grado de madurez y la simulación orientada a agentes va ganando adeptos. En recientes años, podemos encontrar en la literatura varios trabajos de investigación sobre simulación basada en agentes, entre los cuales destacamos:

- ✓ Davidsson (2000) [30]:

De acuerdo con Davidsson, la simulación basada en multi agente (Multi Agent Based Simulation, MABS) ha sido utilizada en la mayoría de los casos en contextos puramente sociales. Sin embargo, comparado con otros enfoques, por ejemplo la simulación de eventos discreta tradicional, la simulación orientada a objetos y la simulación micro dinámica, MABS tiene una serie de propiedades interesantes que la hacen muy útil incluso para otros dominios de aplicación. Por ejemplo, soporta el modelado preservando la estructura de la realidad simulada, simulación de comportamiento pro-activo, computaciones paralelas y muchos escenarios de simulación dinámica. MABS es una técnica útil para simular escenarios, incluso en más dominios técnicos; en particular, para la simulación de sistemas técnicos que están distribuidos e implican interacciones complejas entre humanos y máquinas. Específicamente, Davisson (2000) [30] propone, para ilustrar las ventajas de MABS, una aplicación para monitorizar y controlar edificios inteligentes.

- ✓ Drogoul et al. (2002) [40]:

En este estudio, sus autores exploran las relaciones entre agentes computacionales, dentro de un sistema multiagente (MAS) o una aplicación de inteligencia distribuida (DAI), y las diferentes técnicas reagrupadas bajo el apelativo genérico simulación basada en multi-agente (MABS). Su principal objetivo es mostrar que este tipo de simulación, a pesar de su nombre,

[8] No basada en agentes.

en la práctica rara vez está basada en agentes computacionales. Estos autores basan su demostración en una presentación innovadora del proceso metodológico usado en el desarrollo de sistemas MABS actuales. Esta presentación confía en la definición de diferentes roles implicados en el proceso de diseño, y son capaces de mostrar que la noción de "agente", aunque compartida a un nivel conceptual por diferentes participantes, no implicaba un uso sistemático de agentes computacionales en los sistemas desarrollados. Concluyen también discutiendo cuál podría ser el uso de los agentes computacionales en MABS, en base a la mayoría de las tendencias de investigación interesantes en el campo de los MAS o DAI.

- ✓ Hare y Deadmanb (2004) [69]:

Proporcionan una revisión de la simulación basada en agente en el modelado medioambiental, de modo que los modeladores pueden enlazar sus requisitos al estado del arte actual en las técnicas que se utilizan actualmente para satisfacerlos. Clarifican la terminología y simplifican dos términos clave, modelado y simulación basados en agente, que representan sutilmente diferentes enfoques a la ABS, reflejado en su respectiva vida artificial (A-life) y en sus raíces de inteligencia artificial distribuida. Un conjunto representativo de casos de estudio fue revisado, desde el cual se desarrolla un esquema de clasificación que supone un peldaño para el desarrollo de una taxonomía. La taxonomía puede ser utilizada por los modeladores para corresponder técnicas ABS a sus requisitos.

4.1. Un primer modelo

Para el caso que nos ocupa, se ha considerando el punto de vista de Davidsson (2002) [31] sobre la simulación y modelado de los sistemas basados en agentes, y la arquitectura jerárquica propuesta por Norman et al. (1997) [118].

Teniendo en cuenta esto, se ha definido un modelo en el que se han considerado las siguientes variables para caracterizar un entorno de diseño de ingeniería:

- ✓ complejidad de la tarea,
- ✓ habilidad del diseñador,
- ✓ frecuencia y contenido de la comunicación hacia y desde los diseñadores,
- ✓ y variables psicológicas, incluyendo la comprensión compartida, la confianza y la motivación.

En función de la importancia relativa de estas variables utilizadas en el modelo, el desarrollo de una simulación realista de los equipos de diseño es fundamental . Dada la dificultad para cuantificar estas variables, nuestro enfoque inicial asume un conjunto finito de descriptores cualitativos (por ejemplo, alto...medio...bajo). Los pesos de cada una de las variables se ajustan en base a la información recogida durante las entrevistas con los equipos de diseño reales de Jaguar y Rolls Royce.

En la Figura 42 se ilustra un esquema del modelo de la actividad de diseño. Este modelo para la actividad de diseño individual está basado en el enfoque IDED0 de O'Donnell y Duffy (2002) [129], donde se utilizan varias fuentes de conocimiento para proporcionar el conocimiento necesario para satisfacer los requisitos de diseño, que en este modelo es definido por el objetivo, K_O.

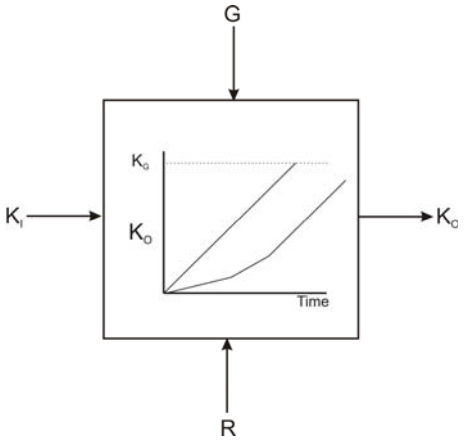

Figura 42. El modelo de la actividad de diseño (Romero et al., 2008) [147].

Dado que los equipos son multidisciplinares, se propone que la tarea, el recurso y el conocimiento de salida sean representados como un vector de competencias esenciales, también como un conjunto cualitativo finito de descriptores. Para realizar la actividad individual, opera bajo un conjunto de restricciones de conocimiento, a saber:

- K_I es el conocimiento interno del diseñador y puede ser explícito o tácito, manteniéndose constante durante toda la tarea.

- K_O, es el conocimiento generado cuando el diseñador realiza la tarea, que coincide con K_G al acabar la misma.
- $G = \{G_T, G_S\}$, es el conjunto de restricciones tanto técnicas (G_T) como sociales (G_S) que dirige y determina la actividad de diseño.
- R, representa las fuentes de conocimiento externo que se encuentran disponibles para el diseñador.

Dentro del modelo se han utilizado un conjunto de algoritmos con objeto de relacionar la entrada de conocimiento con la salida de conocimiento como una función del tiempo. Para simular la interacción diseñador-diseñador o diseñador-recursos, las variables de los caminos de comunicación dependen de ciertos parámetros, incluyendo el medio de comunicación y el contexto, que en definitiva determina la confianza que el diseñador pone en el conocimiento recibido. La implementación actual asume que todas las solicitudes son respondidas de forma inmediata.

Por ahora, la función de la actividad de diseño es lineal, pero se prevén modificaciones cuando finalice el proceso de recogida de información mediante las entrevistas a los diseñadores.

Como mínimo, hemos considerado que un equipo, por definición, debe estar formado por dos personas (con un máximo de 20). La ilustra un equipo de producto integrado simplificado con dos miembros trabajando en una única tarea.

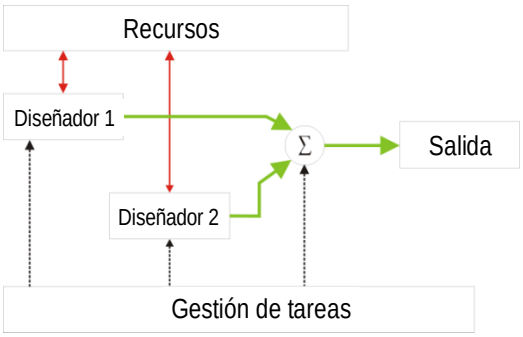

Figura 43. Dos diseñadores trabajando en una única tarea (Romero et al., 2008) [147].

En cuanto a la implementación, se ha utilizado la plataforma JADE para la simulación del MAS. Tenemos seis agentes diseñadores, dos agentes recursos y un gestor de tarea viviendo en el MAS y los estados y comportamientos para cada uno se definen en la Tabla 15.

Cada agente diseñador y recurso tiene dos variables (habilidad y motivación), cuyos valores son asignados al comienzo de la simulación. El agente gestor de tareas (task manager) es el encargado de asignar la tarea a los agentes diseñadores y de monitorizar el progreso de su trabajo. El proceso de diseño se supone secuencial en esta primera fase de desarrollo, de modo que, por ejemplo, el diseñador 2 no empezará a trabajar hasta que el diseñador 1 no haya concluido su tarea (véase Tabla 16).

Tabla 15. Estado y comportamiento de los agentes definidos (Romero et al., 2008) [147].

Agentes	Estado	Comportamiento
Diseñador	Habilidad Motivación	Realiza la tarea asignada por el Gestor de tareas Si la habilidad es menor que la complejidad de la tarea, obtiene información del recurso
Recurso	Capacidad Motivación	Capacidad alta Responde con información
Gestor de tareas	Progreso de tarea Complejidad de tarea	Asigna las tareas a los agentes Mantiene una traza del progreso de la tarea

Tabla 16. Resultados para un grupo de seis personas (Romero et al., 2008) [147].

Habilidad del diseñador	Motivación del diseñador	Tiempo para completar la tarea
Alta	Alta	18
Media	Media	28
Baja	Baja	52
Baja	Media	36
Baja	Alta	28

La Tabla 16 muestra los resultados para un grupo integrado de seis personas, donde el tiempo de realización está en unidades arbitrarias. La complejidad de la tarea se ha considerado alta y como regla de comunicación se asume que cada solicitud es tratada en recepción.

Con este modelado inicial, se ha implementado una relación conocimiento-tiempo que actualmente es lineal, pero que tendrá que ser revisado más adelante para optimizar los algoritmos dentro del modelo de la actividad de diseño. La realimentación inicial procedente de Jaguar y Rolls Royce indica que se trata de un enfoque aceptable para el modelado del comportamiento de los diseñadores.

5. Referencias

[1] Adler, M. R., Davis, A.B., Weihmayer, R. y Worrest, R.W. (1989): *Conflict resolution strategies for non-hierarchical distributed agents*, In L. Gasser y M. Huhns, editors, Distributed Artificial Intelligence Volume II2. Pitman Publishing: London y Morgan Kaufmann: San Mateo, CA, pp 139–16.

[2] Aglets, http://aglets.sourceforge.net/

[3] Ahmed, S., Blessing, L. y Wallace, K. (1999): *The relationships between data, information y knowledge based on a preliminary study of engineering designers*, Proceedings of DETC'99, ASME 1999 Design Theory y Methodology, Las Vegas, Nevada, September 12th-15th.

[4] Ahmed, S. y Wallace, K.M. (2004): *Identifying and supporting the knowledge needs of novice of designers within the aerospace industry*, Journal of Engineering Design, ISSN 0954-4828, Taylor & Francis Ltd.

[5] Ali, S., Koenig, S. y Tambe, M. (2005): *Preprocessing techniques for accelerating the dcop algorithm adopt*, AAMAS, http://teamcore.usc.edu/papers/2005/aamas-paper.pdf

[6] ANA MAS (2005): *Agentes software y sistemas multiagente: Conceptos, arquitecturas y aplicaciones*, Pearson – Prentice Hall, ISBN 84-205-4367-5

[7] Andreasen, M. M. et al (1996): *The design coordination framework: key elements for effective product development*, Proceedings First International Engineering Design Debate (EDD'96), Glasgow, pp 151–174, 23–24 September.

[8] ARC Advisory Group (2003): *Process Simulation & Optimization Worldwide Outlook – Five year market analysis and technology forecast through 2007*, September 2003, http://www.arcweb.com/research/pdfs/Study_pso.pdf

[9] Banares-Alcantara, R. y King, J. M. P. (1997): *Design support systems for process engineering III – design rationale as a requirement for effective support*, Computing and Chemical Engineering, 21(3), 263–276.

[10] Bates, J. (1994): *The role of emotion in believable agents*, Communications of the ACM, vol. 37(7) pp. 122–125, July, http://delivery.acm.org/10.1145/180000/176803/p122-bates.pdf?key1=176803&key2=1902476411&coll=GUIDE&dl=GUIDE&CFID=70702362&CFTOKEN=42747716

[11] Bauer B., Mueller, J., Odell, J. (2001): *Agent UML: A Formalism for Specifying Multiagent Software Systems*, In: Ciancarini, P., Wooldridge, M. (eds.): Agent-Oriented Software Engineering, LNCS 1957. Springer-Verlag (2001) 91-104.

[12] Bernstein, D., Zilberstein, S. y Immerman, N. (2000): *The complexity of decentralized control of markov decision processes*, In UAI, http://anytime.cs.umass.edu/~shlomo/papers/uai00.pdf

[13] Bierel, E. y Roussel, J.M. (1995): *Welcome to the GRAFCET Home Page"*, http://www.lurpa.ens-cachan.fr/grafcet.html

[14] Blessing, L. (1994): *A Process- Based Approach to Computer- Supported Engineering Design*, Thesis, University of Twente, Enschede, Netherlands.

[15] Bonabeau, E. y Meyer, C. (2001): *Swarm intelligence: A whole new way to think about business*, Harvard Business Review (May): 107-114.

[16] Bracewell, R.H., Ahmed, S. y Wallace K. (2004): *DRED and design folders, a way of capturing, storing and passing on, knowledge generated during design projects*, Proceedings of DETC'04, ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Salt Lake City, Utah, USA, September 28 – October 2.

[17] Bradshaw, J.M. (1997), *Software Agents*, AAAI Press, ISBN-13: 978-0262522342

[18] Brice, A. y Johns, B. (1999): *Improving design by improving the design process: A DRAMA white paper*, http://www.enviros.com/index/softw/drama/DownloadPage/DRAMA white paper.pdf

[19] Burmeister, B., Haddadi, A. y Matylis, G. (1997): *Applications of multi-agent systems in traffic and transportation*, IEE Transactions on Software Engineering, vol. 144(1) pp. 51–60, February, http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=580353 (only IEEE subscribers)

[20] Chandrasekaran, B. y Iwasaki, Y. (1993): *Functional representation as design rationale*, IEEE Computer, 26(1), 48–56.
[21] Chavez, A. y Maes, P. (1996): *Kasbah: An agent marketplace for buying and selling goods*, In Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96), London, UK, pp 75–90, http://scholar.google.com/url?sa=U&q=http://agents.media.mit.edu/publications/kasbah-paam96.ps
[22] Chen, L. y Sycara, Katia (1998): *Webmate : A personal agent for browsing and searching*, In Proceedings of the Second International Conference on Autonomous Agents (Agents 98), Minneapolis/St Paul, MN, May, http://delivery.acm.org/10.1145/290000/280789/p132-chen.pdf?key1=280789&key2=4758376411&coll=GUIDE&dl=GUIDE&CFID=75141207&CFTOKEN=8574850
[23] Chung, P. W. H. y Goodwin, R. (1998): *An integrated approach to representing and accessing design rationale*, Engineering Application Artificial Intelligent, 11(1), 149–159.
[24] Clarke, T.L. y McBride, D.K. (2002): *The Simulation of Consciousness: A Grand Challenge for Modelling and Simulation*, Proceedings of First International Conference on Grand Challenges for Modelling and Simulation, Sheraton Four Points Hotel, San Antonio, Texas, 27-31 January, http://www.thesimguy.com/GC/papers/WMC02/G094_CLARKE.pdf
[25] Coates, G. et al (1999): *A methodology for design coordination in a distributed computing environment*, Proceedings 12th International Conference on Engineering Design (ICED'99), Munich, 24–26 August.
[26] G. Coates, R.I. Whitfield, A.H.B. Duffy y W. Hills (2000): *Coordination Approaches & Systems - Part II: An Operational Perspective*, Research in Engineering Design, October 2000, 12:73-89, Springer-Verlag.
[27] Cohen, P.R. y Levesque, H.J. (1991): *Teamwork*, Noûs 25(4), Special Issue on Cognitive Science and Artifical Intelligence, pp 487-512, http://www.stanford.edu/class/cs224m/papers/teamw10.pdf
[28] Conklin, J. y Begeman, M. L. (1989): *gIBIS: a tool for all reasons*, Journal of the American Society for Information Science, 40(3).
[29] Cutosky, M. R., Fikes, R. E., Engelmore, R. S., Genesereth, M. R., Mark, W. S., Gruber, T., Tenenbaum, J. M. y Weber, J. C. (1993): *PACT: An experiment in integrating concurrent engineering systems*, IEEE Transactions on Computers, vol. 26(1) pp. 28–37, ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-93-21.ps.gz
[30] Davidsson, P. (2000): *Multi Agent Based Simulation: Beyond Social Simulation*, In Multi Agent Based Simulation (LNCS Vol. 1979), Springer Verlag, http://www.ide.bth.se/%7Epdv/Papers/MABS2000.pdf
[31] Davidsson, P. (2002): *Agent Based Social Simulation: A Computer Science View*, Journal of Artificial Societies and Social Simulation vol. 5, no. 1, http://jasss.soc.surrey.ac.uk/5/1/7.html
[32] Davis, R. y Smith, R. G. (1983): *Negotiation as a metaphor for distributed problem solving*, Artificial Intelligence, 20, pp 63-109.
[33] de la Garza, J. M. y Alcantara Jr, P. T. (1997): *Using parameter dependency network to represent design rationale*, Journal Computing in Civil Engineering, 11(2), 102–112.
[34] de la Garza, J. M. y Ramakrishnan, S. (1995): *A tool for designers to record design rationale of a constructed project*, 10th International Conference on Applications of Artificial Intelligence in Engineering, Southampton, U.K. Computational Mechanics Publications, pp 533–540.
[35] DeLoach, S. (2001): Analysis and design using MaSE and agentTool, Actas del 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS).
[36] Dignum, V., Weigand, H. y Xu L. (2002): *Agent Societies: Towards Frameworks-Based Design*, In: The Second International Workshop on Agent-Oriented Software Engineering (AOSE-2001), Lecture Notes of Computer Science Volume 2222/2002, Springer-Verlag, pp 33 – 49, http://www.cs.uu.nl/people/xu/papers/AOSE01.pdf
[37] Documentation of DTI Funding for HIPARSYS Project, 2005.
[38] Dong, A., Moore, F., Woods, C. y Agogino, A.M. (1995): *Managing Design Knowledge in Enterprise-Wide CAD, Advances in Formal Design Methods for CAD*, (Eds. J.S. Gero and F. Sudweeks; Preprints of the IFIP WG 5.2 Workshop on Formal Design Methods for CAD, Key

Centre of Design Computer, University of Sydney), pp 329-347, http://best.me.berkeley.edu/~adong/cae.doc

[39] Doorenbos, R., Etzioni, O. y Weld, D. (1997): *A scaleable comparison-shopping agent for the world wide web*, In Proceedings of the First International Conference on Autonomous Agents (Agents 97), Marina del Rey, CA, 1997, pp. 39–48.

[40] Drogoul, A., Vanbergue, D. y Meurisse, T. (2002): *Multi-Agent Based Simulation: Where are the Agents?*, Proceedings of MABS'02 (Multi-Agent Based Simulation, Bologna, Italy, July 2002), LNCS, Springer-Verlag, http://www-poleia.lip6.fr/~drogoul/papers/drogoul.mabs02.pdf

[41] Durfee, E. H. y Lesser, V. R. (1987): *Using partial global plans to coordinate distributed problem solvers*, Proceedings of the10th International Joint Conference on Artificial Intelligence, Milan, Italy, 875-883.

[42] Ernst, J. (2002): *Collaborative Decision making and Personal Knowledge Management with R-Objects Pepper™*, International Workshop Real World Rdf And Semantic Web Applications 2002, 7 May, http://www.cs.rutgers.edu/~shklar/www11/final_submissions/paper9.pdf

[43] Esfandiari, B. Deflandre, G. y Quinqueton, J. (1996): *An interface agent for network supervision*, In Proceedings of the ECAI-96 Workshop on Intelligent Agents for Telecom Applications, Budapest, Hungary, http://scholar.google.com/url?sa=U&q=http://www.sce.carleton.ca/faculty/esfandiari/papers/ecai96b.ps

[44] Evans, R., Kearny et al. (2001): *MESSAGE: Methodology for Engineering Systems of Software Agents*, European Institute for Research and Strategic Studies in Telecommunications (Eurescom), http://www.eurescom.de/~pub-deliverables/P900-series/P907/TI1/p907ti1.pdf

[45] Ferber, J. (1995): *Les systèmes multi-agents: vers une intelligence collective*. Paris, France, InterEditions.

[46] Finin, T. (1997): *UMBC KQML Web*, http://www.cs.umbc.edu/kqml/

[47] Foundation For Intelligent Physical Agents (2002): *FIPA Contract Net Interaction Protocol (SC00029H)*, http://www.fipa.org/specs/fipa00029/SC00029H.pdf

[48] Fisher, M. (1995): *Representing and executing agent-based systems*, In M.Wooldridge and N. R. Jennings, editors, Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890). Springer-Verlag: Berlin, Germany, pp 307–323, http://citeseer.ist.psu.edu/context/156229/547769

[49] Fischer, G. y McCall, R. (1989): *JANUS: Integrating hypertext with a knowledge-based design environment*, In: Halasz, F., Meyrowitz, N. (eds.), Hypertext, Addison-Wesley, pp pages 105–117.

[50] Fischer, G., McCall, R. y Morch, A. (1989): *Design environments for constructive and argumentative design*, In: Bice, K., Lewis, C. (eds.), Conference on Wings for the Mind, Addison-Wesley, pp 269–275.

[51] Fischer, K., Müller, J. P. y Pischel, M. (1996): *Cooperative transportation scheduling: An application domain for dai*, Applied Artificial Intelligence, vol. 10(1) pp. 1–34, http://docserver.ingentaconnect.com/deliver/connect/tandf/08839514/v10n1/s1.pdf?expires=1146739216&id=28854614&titleid=37&accname=National+Oceanographic+Library&checksum=CE94E91BCAF186D2C3936E31B30DAAF2

[52] Foner, L.N. (1997): *Entertaining agents: A sociological case study*, In Proceedings of the First International Conference on Autonomous Agents (Agents 97), Marina del Rey, CA, pp 122–129, http://foner.www.media.mit.edu/people/foner/Reports/Agents-97/Julia.pdf

[53] Forsyth, C. (2004): *Roadmap for Modelling and Simulation of Continuous and Hybrid Processes – T.6. Exploitation of Modelling*, WG#7 Roadmap of Simulation in Process Industries, Sim-Serv, May 2004, http://www.sim-serv.com/wg_doc/WG7_Exploitation.pdf

[54] The Foundation for Intelligent Physical Agents, http://www.fippa.org

[55] Fowler, D.W. et al (2004): *The Designers' Workbench: Using Ontologies and Constraints for Configuration*, Proceedings of The Twenty-fourth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Queens' College, Cambridge, UK.

[56] Fox, M. S., Chionglo, J. F. y Barbuceanu, M. (1993): *The integrated supply chain management system*, Technical report, Department of Industrial Engineering, University of Toronto, http://www.eil.utoronto.ca/public/iscm-intro.ps

[57] Garcia, A. C. B. y de Souza C. S. (1997): *Add+: Including rhetorical structures in active documents*, Artificial Intelligent for Engineering Design, Analysis and Manufacturing, 11(2), 109–124.
[58] Garcia, A. C. B. y Howard, C. H. (1992): *Acquiring design knowledge through design decision justification*, Artificial Intelligent for Engineering Design, Analysis and Manufacturing, 6(1), 59–71.
[59] Genesereth, M. R. (1996): Knowledge Interchange Format (KIF), http://logic.stanford.edu/kif/
[60] Gómez, C.J. y Pavón, J. (2003): *Curso de doctorado: Agentes Inteligentes - Desarrollo de Sistemas Multi-Agente. La metodología INGENIAS*, UCM, Departamento de Sistemas Informáticos y Programación, http://grasia.fdi.ucm.es
[61] Grand, S. y Cliff, D. (1998): *Creatures: Entertainment software agents with artificial life*, Autonomous Agents and Multi-Agent Systems, vol. 1(1), http://citeseer.ifi.unizh.ch/cache/papers/cs/26466/http:zSzzSzwww-uk.hpl.hp.comzSzpeoplezSzdave_cliffzSzgrand_cliff.pdf/grand97creatures.pdf
[62] Griffeth, N. D. y Velthuijsen, H. (1994): *The negotiating agents approach to run-time feature interaction resolution*, In L. G. Bouma and H. Velthuijsen, editors, Feature Interactions in Telecommunications Systems. IOS Press, 1994, pp. 217–235.
[63] Grosz, B. y Kraus, S. (1996): *Collaborative Plans for Complex Group Action*, In Artificial Intelligence, 86(2), pp. 269-357, http://www.eecs.harvard.edu/grosz/papers/CollaborativePlans.pdf
[64] Grosz, B. y Kraus, S. (1999): *The Evolution of SharedPlans*, In Foundations and Theories of Rational Agencies, A. Rao and M. Wooldridge, eds. pp. 227-262, http://www.eecs.harvard.edu/grosz/papers/Evolution_SharedPlans.pdf
[65] Groumpos, P.P. et al (2004): *Roadmap for modelling and simulation of continuous and hybrid processes*, WG#7 Roadmap of Simulation in Process Industries, Sim-Serv, http://www.sim-serv.com/WG7_Heterogeneous.pdf
[66] Guarino, N. (1998): *Formal Ontologies and Information Systems*, N. Guarino (ed.): 3-15.
[67] Guha, R. V. , D. B. Lenat (1990), *CyC: a Midterm Report*, Artificial Intelligence, 11: 32-59.
[68] Guo, L., Chen-Burger, Y. y Roberston D. (2004): *Mapping a Business Process Model to a Semantic Web Service Model*, Proceedings IEEE International Conference on Web Services, San Diego, California, USA.
[69] Hare, M. y Deadmanb, P. (2004): *Further towards a taxonomy of agent-based simulation models in environmental management*, Mathematics and Computers in Simulation 64 (2004) 25–40, http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V0T-49PYR5V-5-5&_cdi=5655&_user=126770&_orig=search&_coverDate=01%2F05%2F2004&_qd=1&_sk=999359998&view=c&wchp=dGLbVzb-zSkWz&md5=61ac9e3801e83b055d7bacce67b4b1a3&ie=/sdarticle.pdf
[70] Hayes-Roth, B. (1995): *Agents on stage: Advancing the state of the art in AI*, In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), Montréal, Québec, Canada, August, pp 967–971, http://scholar.google.com/url?sa=U&q=ftp://ksl.stanford.edu/pub/KSL_Reports/KSL-95-50.ps.gz
[71] Hayes-Roth, B., Hewett, M., Washington, R., Hewett, R. y Seiver, A. (1989): *Distributing intelligence within an individual*, In L. Gasser and M. Huhns, editors, Distributed Artificial Intelligence Volume II. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, pp 385–412.
[72] Huang, J., Jennings, N. R. y Fox, J. (1995): *An agent-based approach to health care management*, Applied Artificial Intelligence, vol. 9(4) pp. 401–420, http://www.ecs.soton.ac.uk/~nrj/download-files/Intelligent-Agents95.ps
[73] Huhns, M. N. y Singh, M. P. (1998): *Managing heterogeneous transaction workflows with cooperating agents*, In N. R. Jennings and M. Wooldridge, editors, Agent Technology: Foundations, Applications and Markets. Springer-Verlag: Berlin, Germany, pp 219-239.
[74] Huynh, T. D., Jennings, N. R. y Shadbolt, N. R. (2004): *FIRE: An Integrated Trust and Reputation Model for Open Multi-Agent Systems*, Proceedings of the 16th European Conference on Artificial Intelligence (ECAI), http://www.ecs.soton.ac.uk/~nrj/download-files/dong-ecai2004.pdf
[75] Java RMI (Remote Method Invocation), http://java.sun.com/products/jdk/rmi/

[76] Jennings, N. R. (1993): *Specification and implementation of a belief desire joint-intention architecture for collaborative problem solving*, Journal of Intelligent and Cooperative Information Systems, vol. 2(3) pp. 289–318, http://www.ecs.soton.ac.uk/~nrj/download-files/IJICIS-2-3.ps
[77] Jennings, N.R. (1996): *Coordination Techniques for Distributed Artificial Intelligence*, O'Hare GMP, Jennings NR (eds) Foundations of distributed artificial intelligence. Wiley, pp 187–210, http://www.ecs.soton.ac.uk/~nrj/download-files/FOUND-DAI-COORD.ps
[78] Jennings, N. R. (2000): *On Agent-Based Software Engineering*, Artificial Intelligence, 117 (2) 277-296, http://www.ecs.soton.ac.uk/~nrj/download-files/aij2000.ps
[79] Jennings, N. R., Corera, J., Laresgoiti, I., Mamdani, E. H., Perriolat, F., Skarek, P. y Varga, L. Z. (1996): *Using ARCHON to develop real-world DAI applications for electricity transportation management and particle acceleration control*, IEEE Expert, vol. 11(6) pp. 60–88, December, http://www.ecs.soton.ac.uk/~nrj/download-files/IEEE-Expert-part1.pdf
[80] Jennings, N. R., Faratin, P., Johnson, M. J., Norman, T. J., O'Brien, P. y Wiegand, M. E. (1996): *Agent-based business process management*, International Journal of Cooperative Information Systems, vol. 5(2-3) pp 105–130, http://www.ecs.soton.ac.uk/~nrj/download-files/IJCIS96.ps
[81] Jennings, N. R., Faratin, P., Norman, T. J., O'Brien, P. y Odgers, B. (2000): *Autonomous Agents for Business Process Management*, International Journal of Applied Artificial Intelligence 14 (2) 145-189, http://www.ecs.soton.ac.uk/%7Enrj/download-files/aaij991.ps
[82] Jennings, N. R., Sycara, K. y Wooldridge, M. (1998): *A Roadmap of Agent Research and Development*, International Journal of Autonomous Agents and Multi-Agent Systems 1 (1) 7-38, http://www.ecs.soton.ac.uk/~nrj/download-files/jaamas98.pdf
[83] Jini.org - The Community Resource for Jini Technology, http://www.jini.org
[84] JXTA.org, http://www.jxta.org/
[85] Karcanias, N. (2004): *Modelling and Simulation in Technological and Emerging Fields: Emerging Challenges*, WG#7 Roadmap of Simulation in Process Industries, Sim-Serv, http://www.sim-serv.com/wg_doc/WG7_General_roadmap.pdf
[86] King, J. M. P. y Banares-Alcantara, R. (1997): *Extending the scope and use of design rationale*, Artificial Intelligent for Engieering Design, Analysis and Manufacturing, 11(2), 155–167.
[87] Kini, D., Georgeff, M. y Rao, A. (1997): *A methodology and modelling technique for systems of BDI agents*, Australian Artificial Intelligence Institute.
[88] Kraus, S., Wilkenfeld, J. y Zlotkin, G. (1995): *Multiagent negotiation under time constraints*, Artificial Intelligence, vol. 75(2), pp 297–345, http://cs.biu.ac.il/~sarit/./Articles/16.ps
[89] Krulwich, B. (1996): *The BargainFinder agent: Comparison price shopping on the Internet*, In J. Williams, editor, Bots, and other Internet Beasties. Macmillan Computer Publishing: Indianapolis, pp 257–263.
[90] Kunz, W. y Rittel, W. (1970): *Issues as elements of information systems*, Working paper 131, Center for Planning and Development Research, University of California, Berkeley.
[91] Laird, J. E., Newell, A. y Rosenbloom, P. S. (1987): *SOAR: An architecture for general intelligence*, Artificial Intelligence, 33, 1-64.
[92] Lee, J. y Lai, K. (1991): *What's in design rationale*, Human-Comput. Interaction, 6(3–4), 251–280.
[93] Lesser, V. R. (1991): *A Retrospective View of FA/C Distributed Problem Solving*, IEEE Transactions on Systems, Man, and Cybernetics, http://citeseer.ist.psu.edu/cache/papers/cs/7249/ftp:zSzzSzdis.cs.umass.eduzSzpubzSzLesser-retro91.pdf/lesser91retrospective.pdf
[94] Lésperance, Y., Levesque, H. J., Lin, F., Marcu, D., Reiter, R. y Scherl, R. B. (1996): *Foundations of a logical approach to agent programming*, In M. Wooldridge, J. P. Müller, and M. Tambe, editors, Intelligent Agents II (LNAI Volume 1037), Springer-Verlag: Berlin, Germany, pp. 331–346, http://www.cs.toronto.edu/cogrobo/Papers/agentprog.pdf
[95] Lester, J. C. y Stone, B. A. (1997): *Increasing believability in animated pedagogical agents*, In Proceedings of the First International Conference on Autonomous Agents (Agents 97), Marina del Rey, CA, 1997, pp. 16–21, http://delivery.acm.org/10.1145/270000/269943/p16-lester.pdf?key1=269943&key2=5282476411&coll=portal&dl=ACM&CFID=70702954&CFTOKEN=98439118

[96] Levesque, H.J., Cohen, P.R. y Nunes, J.H.T. (1990): *On acting together*, Proceedings of the Annual Meeting of the American Association for Artificial Intelligence, AAAI-90, http://www.cse.ogi.edu/CHCC/Publications/on_acting_together_Levesque.pdf

[97] Ljunberg, M. y Lucas, A. (1992): *The OASIS air traffic management system*, In Proceedings of the Second Pacific Rim International Conference on AI (PRICAI-92), Seoul, Korea.

[98] Lubars, M. D. (1991): *Representing design dependencies in an issue-based style*, IEEE Software, 6(4), 81–89.

[99] Luck, M. (2006): *50 facts about Agent-Based Computing*, AgentLink III, School of Electronics and Computer Science University of Southampton, http://www.ecs.soton.ac.uk/news/pdfs/AgentLink_Fifty_Facts.pdf

[100] Luck, M., McBurney, P., Shehory, O., Willmott, S. y the AgentLink Community (2005): *Agent Technology Roadmap. A Roadmap for Agent Based Computing*, AgentLink III, ISBN 085432 845 9, http://www.agentlink.org/roadmap/al3rm.pdf

[101] MacLean, A., Young, R., Belloti, V., Moran, T. (1991): *Questions, options, and criteria: Elements of design space analysis*, Human-Comput. Interaction, 6(3–4), 201–250

[102] Maes, P. (1994): *Agents that reduce work and information overload*, Communications of the ACM, vol. 37(7), pp 31–40, July, http://delivery.acm.org/10.1145/180000/176792/p30-maes.pdf?key1=176792&key2=0318376411&coll=GUIDE&dl=GUIDE&CFID=75140653&CFTOKEN=17921961

[103] Maheswaran, R. y Basar, T. (2003): *Coalition formation in proportionality fair divisible auctions*, In AAMAS.

[104] Mark, W., Tyler, S., McGuire, J. y Schlossberg, J. (1992): *Commitment-based software development*, IEEE Trans. Softw. Eng., 18(10), 870–886.

[105] Mailler, R. (2005): *Comparing two approaches to dynamic, distributed constraint satisfaction*, In AAMAS, http://www.ai.sri.com/~mailler/publications/p845-mailler.pdf

[106] Mayfield, J., Labrou, Y. y Finin, T. (1996): *Evaluating KQML as an agent communication language*, In M. Wooldridge, J. P. M¨uller, and M. Tambe, editors, Intelligent Agents II (LNAI Volume 1037), Springer-Verlag: Berlin, Germany, pp 347–360, http://www.flacp.fujitsulabs.com/~yannis/publications/lnai95.pdf

[107] McCabe, F. G. y Clark, K. L. (1995): *April — agent process interaction language*, In M. Wooldridge and N. R. Jennings, editors, Intelligent Agents: Theories, Architectures, and Languages (LNAI Volume 890), Springer-Verlag: Berlin, Germany, pp 324–340.

[108] McCall, R. J. (1991): *PHI: A conceptual foundation for design hypermedia*, Design Studies, 12(1), 30–41

[109] Merz, M., Lieberman, B. y Lamersdorf, W. (1997): *Using mobile agents to support inter-organizational workflow management*, Applied Artificial Intelligence, vol. 11(6) pp. 551–572, http://www.agent.ai/doc/upload/200302/merz97_1.pdf

[110] Modi, P., Shen, W., Tambe, M. y Yokoo, M. (2005): *Adopt: Asynchronous distributed constraint optimization with quality guarantees*, AIJ, 161:149–180, http://cmu.edu/~pmodi/papers/modiaij03.ps

[111] Moslehi, K. y Kumar, R., *Una visión de red eléctrica autocorrectora*, Revista ABB, Abril 2006, http://dialnet.unirioja.es/servlet/articulo?codigo=2214204&orden=101657&info=link

[112] Myers, K. L., Zumel, N. B. y Garcia, P. (1999): *Automated capture of rationale for the detailed design process*, In: Uthurusamy, R: Hayes-Roth, B. (eds.), Eleventh Conference on Innovative Applications of Artificial Intelligence, AAAI Press, pp 876–883.

[113] Nagy, R. L. (1990): *A Knowledge Base Data Representation for Collaborative Mechanical Design*, Master Thesis, Department of Mechanical Engineering, Oregon State University, November.

[114] Nagy, R. L. y Ullman, D.G. (1992): *A Data Representation for Collaborative Mechanical Design*, Research in Engineering Design, Vol. 3, No. 4, 1992, pp. 233-242.

[115] R. Nair, M.Tambe, M.Yokoo, D.Pynadath, y S.Marsella (2003): *Taming decentralized pomdps: Towards efficient policy computation for multiagent settings*, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), http://teamcore.usc.edu/nair/publications/nair-ijcai03.pdf

[116] Newell, A. (1990): *Unified Theories of Cognition*, Cambridge, Massachusetts: Harvard University Press.
[117] NIIIP. Mother NIIIP Homepage, 1996, http://www.niiip.org
[118] Norman, T. J., Jennings N. R., Faratin, P. y Mamdani, E. H. (1997): *Designing an Implementing a Multi-Agent Architecture for Business Process Management*, Proceedings of the ECAI'96 Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III, http://www.ecs.soton.ac.uk/~nrj/download-files/argmas06.pdf
[119] Paruchuri, P., Bowring, E., Nair, R., Pearce, J.P., Schurr, N., Tambe, M. y Varakantham, P. (2006): *Multiagent Teamwork: Hybrid Approaches*, Computer society of India Communications (Invited), http://teamcore.usc.edu:8080/teamcore/papers/2006/tambe.pdf
[120] Parunak, H.V.D. (1987): *Manufacturing experience with the contract net*, In M. Huhns, editor, Distributed Artificial Intelligence. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1987, pp 285–310, http://www.newvectors.net/staff/parunakv/YAMS87.pdf
[121] Parunak, H.V.D. (1998): *Practical and Industrial Applications of Agent-Based Systems*, Industrial Technology Institute, http://agents.umbc.edu/papers/apps98.pdf
[122] Pavón, J. (2003), *Agentes inteligentes - Comunicación entre agentes*, Departamento de Sistemas Informáticos y Programación, UCM, http://grasia.fdi.ucm.es
[123] Pérez, M. C., Explotación de los córpora textuales informatizados para la creación de bases de datos terminológicas basadas en el conocimiento, Universidad de Málaga, Volumen 18 (2002), ISSN: 1139-8736, http://elies.rediris.es/elies18/index.html
[124] Prabhakar, S. y Goel, A. K. (1998): *Functional modelling for enabling adaptive design of devices for new environments*, Artificial Intelligent in Engineering, 12(4), 417–444.
[125] Pynadath, D. y Tambe, M. (2002): *The Communicative Multiagent Team Decision Problem: Analyzing Teamwork Theories and Models*, Journal of Artificial Intelligence Research 16 (2002), pp. 389-423, AI Access Foundation and Morgan Kaufmann Publishers, http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume16/pynadath02a.pdf
[126] Pynadath, D., Tambe, M., Chauvat, N. y Cavedon, L. (1999): *Toward team-oriented programming*, Proceedings of the Agents, theories, architectures and languages (ATAL'99) workshop, published as Springer Verlag LNAI "Intelligent Agents VI", http://citeseer.ist.psu.edu/cache/papers/cs/11575/http:zSzzSzwww.isi.eduzSz~pynadathzSzResearchzSzatal99.pdf/pynadath99toward.pdf
[127] Obitko, M. y Marík, V. (2002): *Ontologies for Multi-Agents Systems in Manufacturing Domain*, Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA'02), IEEE.
[128] Object Management Group (2004): *Common Object Request Broker Architecture: Core Specification*, http://www.omg.org/docs/formal/04-03-12.pdf
[129] O'Donnell, F. y A. Duffy (2002): *Modelling design development performance*, International Journal of Operations & Production Management 22(11): 1198-1221.
[130] Oliveira, E., Fonseca, J. M. y Steiger-Garcao, A. (1997): *MACIV: A DAI based resource management system*, Applied Artificial Intelligence, vol. 11(6), pp 525–550, http://paginas.fe.up.pt/usr/users1/deec/eol/public_html/PUBLICATIONS/aaij97.ps
[131] Ören, T.I. (1990): *A Paradigm for Artificial Intelligence in Software Engineering*, In Advances in Artificial Intelligence in Software Engineering - Vol. 1, T.I. Ören (ed.), JAI Press, Greenwich, Connecticut, pp. 1-55.
[132] Ören, T.I. (1994): *Artificial Intelligence and Simulation*, Annals of Operations Research, vol. 53, pp. 287-319.
[133] Ören, T.I. (1995): *Artificial Intelligence and Simulation: A Typology*, Proceedings of the 3rd Conference on Computer Simulation, S. Raczynski (ed.), Mexico City, Nov. 15-17, pp. 1-5.
[134] Ören, T.I. (2000): *Understanding Systems: A Taxonomy and Performance Factors*, Proceedings of FOODSIM'2000, Nantes, France. SCSCI, San Diego, CA, pp. 3-10, June 26-27, http://www.site.uottawa.ca/~oren/pubs/pubs-2000-03-understanding.pdf
[135] Ören, T.I. (ed.) (2001): *Software Agents and Simulation*, Special Issue of Simulation Journal, June 2001.

[136] Ören, T.I. (2001): *Software Agents for Experimental Design in Advanced Simulation Environments*, Proceedings of the 4th St. Petersburg Workshop on Simulation, pp. 89-95, June 18-23, http://www.site.uottawa.ca/~oren/pubs/pubs-2001-03-StPetersburg.pdf

[137] Ören, T.I. (2002): *Future of Modelling and Simulation: Some Development Areas*, Proceedings of the 2002 Summer Computer Simulation Conference.

[138] Ören, T.I (2005): *Toward the Body of Knowledge of Modelling and Simulation*, Interservice/Industry Training, Simulation, y Education Conference (I/ITSEC), Paper No. 2025.

[139] Pavón, J., Gómez-Sanz, J.J. (2003): *Agent Oriented Software Engineering with INGENIAS*, CEEMAS 2003, Lectures Notes in Computer Science 2691, Springer-Verlag, 394 – 403, http://ingenias.sourceforge.net/

[140] Amchurn, S. D., Sierra, C., Godo, L. y Jennings, N. R. (2004): *Devising a trust model for multi-agent interactions using confidence and reputation*, International Journal of Applied Artificial Intelligence 18(9-10) pp. 833-852, http://eprints.ecs.soton.ac.uk/10155/01/jaai04.pdf

[141] Ramesh, B. y Dhar, V. (1992): *Supporting systems development using knowledge captured during requirements engineering*, IEEE Trans. Softw. Eng., 18(6), 498–511.

[142] Ramesh, B. y Sengupta, K. (1995): *Multimedia in a design rationale decision support system*, Decision Support Syst., 15(3), 181–196.

[143] Regli, W. C, Hu, X., Atwood, M. y Sun, W. (2000): *A Survey of Design Rationale Systems: Approaches, Representation, Capture and Retrieval*, Engineering with Computers (2000) 16: 209–235, Springer-Verlag London Limited, http://www.pages.drexel.edu/~sunwei/WSUN-Papers/Survey-Hu.pdf

[144] Rich, C. y Sidner, C. (1997): *Collagen: When agents collaborate with people*, Proceedings of the International Conference on Autonomous Agents (Agents 97), Marina del Rey, CA, pp. 284-291.

[145] Robinson, M., Clegg, C. y Crowder, R. (2006): *Organisational Modelling (HIPARSYS)*, UTP Spring Conference 2006.

[146] Rogers, R.V. (1997): *What Makes A Modelling & Simulation Professional?: The Consensus View From One Workshop*, Proceedings of 1997 WSC, (Atlanta, GA, Dec. 7-10). IEEE, Piscataway, New Jersey, pp. 1375-1382, http://www.scs.org/scsarchive/getDoc.cfm?id=1133

[147] Romero, M.C., Crowder, R. M., Sim, Y. W. y Payne, T. R. (2008): *Applying Multi-Agent Systems To Organizational Modelling In Industrial Environments*, Actas ICEIS 2008 – Artificial Intelligence and Decision Support Systems, 11th International Conference on Enterprise Information Systems, 181– 186.

[148] Rosenschein, J.S. (1985): *Rational Interaction: Cooperation Among Intelligents Agents*, PhD thesis, Computer Science Department, Standford University, Stanford, CA 94395.

[149] Scerri, P., Farinelli, A., Okamoto, S. y Tambe, M. (2005): *Allocating tasks in extreme teams*, In AAMAS, http://www.cs.cmu.edu/~pscerri/papers/LADCOP-AAMAS05.pdf

[150] Scerri, P., Pynadath, D. y Tambe, M. (2002): *Towards adjustable autonomy for the real-world*, JAIR, 17:171–228, http://www.cs.cmu.edu/~pscerri/papers/JAIR-AA.pdf

[151] Schreiber, G., Wielinga, J., Akkermans, J. y de Velde, W.V. (1994): *Common-KADS: A comprehensive methodology for KBS development*, University of Amsterdam, Netherlands Energy Research Foundation ECN and Free University of Brussels.

[152] Schreiber, G. et al (2000): *Knowledge engineering and management*, The MIT Press, ISBN 0-262-19300-0.

[153] Schwuttke, U. M. y Quan, A. G. (1993): *Enhancing performance of cooperating agents in real-time diagnostic systems*, In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI- 93), Chambéry, France, pp. 332–337.

[154] Shadbolt, N. y Milton, N. (1999): *From knowledge engineering to knowledge management*, British Journal of Management, Vol. 10, 309-322.

[155] Shakeri, C. y Brown D.C. (1998): *Mapping Multi-Disciplinary Design Processes*, InterJournal of Complex Systems.

[156] Shipman III, F. M., McCall, R. J. (1997): *Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication*, Artificial Intelligent for Engineering Design, Analysis and Manufacturing, 11(2), 141–154.

[157] Shoham, Y. (1993): Agent-oriented programming, Artificial Intelligence, Volume 60(1) pp. 51–92, http://www.damas.ift.ulaval.ca/~coursMAS/Agent-Oriented-Programming.pdf
[158] Smith R (1980): *The contract net protocol: high level communication and control in distributed problem solver*, IEEE Transactions on Computers, 29:1104-1113.
[159] Sprumont, F. y Müller, J. P. (1997): *Amacoia: A multi-agent system for designing flexible assembly lines*, Applied Artificial Intelligence, vol. 11(6) pp. 573–590.
[160] Steve, G. , Gangemi, A., Pisanelli, D. (1998), *Ontology Integration: Experiences with Medical Ontologies*, N. Guarino (ed.): 163-178.
[161] Sun Microsystems, Java Technology, http://java.sun.com/
[162] Sun Microsystems, JXTA Technology, http://www.sun.com/software/jxta/
[163] Sycara, K. P. (1988): *Resolving goal conflicts via negotiation*, In Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88), St. Paul, MN, 1988.
[164] Sycara, K. (1990): *Negotiation planning: An AI approach*, European Journal of Operational Research, vol. 46, pp 216–234.
[165] Sycara, K. P. (1990): *Persuasive argumentation in negotiation*, Theory and Decision, vol. 28, pp 203-242.
[166] Sycara, K.P. (1998): *Multiagent Systems*, AI Magazine 19(2), pp 79-92, http://www.aaai.org/AITopics/assets/PDF/AIMag19-02-2-article.pdf
[167] Sycara, K., Decker, K., Pannu, A., Williamson, M. y Zeng, D. (1996): *Distributed Intelligent Agents*. IEEE Expert, vol. 11(6), http://www.cs.cmu.edu/~softagents/papers/ieee-agents96.pdf
[168] Tambe, M., Pynadath, D.V., Chauvat, N., Das, A. y Kaminka, G. (2000): *Adaptive Agent Architectures for Heterogeneous Team Members*, Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000), pp 301-308, http://teamcore.usc.edu:8080/teamcore/papers/2000/icmas.ps
[169] Trappl, R. y Petta, P. (1997): *Creating Personalities for Synthetic Actors*, Springer-Verlag: Berlin, Germany.
[170] Tsvetovatyy, M., Gini, M., Mobasher, B. y Wieckowski, Z. (1997): *MAGMA: An agent-based virtual marketplace for electronic commerce*, Applied Artificial Intelligence, vol. 11(6) pp. 501–524, http://www.cs.umn.edu/Research/airvl/magnet/papers/magma.pdf
[171] Ullman, D. (1995): *A Taxonomy for Classifying Engineering Decision Problems and Support Systems*, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Cambridge University, 9, 427-438, http://web.engr.oregonstate.edu/~ullman/decision.htm
[172] Voland, G. (1999): *Engineering by Design*, Addison Wesley, ISBN 0-201-49851-0.
[173] Wang, H. y Wang, C. (1997): *Intelligent agents in the nuclear industry*. IEEE Computer, vol. 30(11), pp 28–34, http://ieeexplore.ieee.org/iel3/2/13781/00634838.pdf?tp=&arnumber=634838&isnumber=13781
[174] Wavish, P. y Graham, M. (1996): *A situated action approach to implementing characters in computer games*, Applied Artificial Intelligence, vol. 10(1), pp 53–74, http://docserver.ingentaconnect.com/deliver/connect/tandf/08839514/v10n1/s3.pdf?expires=1146743239&id=28856864&titleid=37&accname=National+Oceanographic+Library&checksum=4ECD0ADBAC80DA6D72AAAA5692AD60F5
[175] Wikipedia.org: *KADS*, http://en.wikipedia.org/wiki/KADS
[176] Whitfield, R. I., Coates, G., Duffy, A. H. B. y Hills, B. (2000): *Coordination Approaches and Systems – Part I: A Strategic Perspective*, Research in Engineering Design (2000)12: 48–60, Springer-Verlag London Limited, http://www.cad.strath.ac.uk/~ianw/publications/resources/Ref11.pdf
[177] Whitfield, R. I. et al (2002): *Distributed design coordination*, Research in Engineering Design 13 (2002), pp. 243 – 252.
[178] Woods, S.G. y Barbacci, M. R. (1999): *Architectural Evaluation of Collaborative Agent-Based Systems*, Technical Report, CMU/SEI-99-TR-025, Software Engineering Institute, Carnegie Mellon University, PA, USA, http://www.cs.toronto.edu/~mkolp/lis2103/99tr025.pdf
[179] 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.
[180] Wooldridge, M. (2002): *An Introduction to MultiAgent Systems*, Wiley, ISBN 047149691, UK.

[181] Wooldridge, M. y Jennings, N.R. (1995): *Intelligent Agents: theory and practice*, The Knowledge Engineering Review, 10(2), pp 115-152, http://www.csc.liv.ac.uk/~mjw/pubs/ker95.pdf
[182] W3C: *SOAP Specifications*, http://www.w3.org/TR/soap/
[183] W3C: Extensible Markup Language (XML), http://www.w3.org/XML/
[184] Xu, L. y Weigand, H (2001): *The Evolution of the Contract Net Protocol*, In: The 2nd International Conference on Web-Age Information Management (WAIM2001), July, Lecture Notes of Computer Science, Springer-Verlag, http://www.cs.uu.nl/people/xu/papers/WAIM01.pdf
[185] Yakemovic, K.C.B. y Conklin, J. (1989): *The Capture of Design Rationale on an Industrial Development Project: Preliminary Report*, MCC Technical Report Number STP-279-89, July.
[186] Yakemovic, K.C.B. y Conklin, J. (1990): *Report on a Development Project Use of an Issue-Based KNOWLEDGE System*, MCC Technical Report Number STP-247-90, June.
[187] Zeigler, B.B. (1976): *Theory of Modelling and Simulation*, John Wiley & Sons, New York.
[188] Zeigler, B.P., Elzas, M.R., Klir, G.J. y Ören, T.I. (eds.) (1979): *Methodology in System Modelling & Simulation*, North-Holland, Amsterdam, 537 pages.
[189] Zeigler, B.P. y Oren, T.I. (1986): *Multifacetted, Multiparadigm Modelling Perspectives: Tools for the 90's*, Proceedings of the 1986 Winter Simulation Conference, ACM Press, Washington, D.C., United States, ISBN:0-911801-11-1, pages: 708 – 712.
[190] Zeng, D. y Sycara, K. (1997): *Benefits of learning in negotiation*, Proceedings of AAAI-97, Providence, Rhode Island, USA, http://www-ec.njit.edu/~bartel/papersCIS767/learningnegotiations.pdf