Estudio de métodos de desarrollo de sistemas multiagente

Vicente J. Julián, Vicente J. Botti

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n
Valencia, 46071
{vinglada, vbotti}@dsic.upv.es

Resumen

Dentro del marco de la Inteligencia Artificial, los Sistemas Multiagente se han caracterizado por ofrecer una posible solución al desarrollo de problemas complejos con características distribuidas. A la hora de abordar el desarrollo de sistemas multiagente es indudable un notable aumento de complejidad, así como la necesidad de adaptar técnicas ya existentes o, en ocasiones, el desarrollo de técnicas y herramientas nuevas. Resulta evidente que el desarrollo de cualquier tipo de software necesita de la existencia de métodos y herramientas que faciliten la obtención de productos finales fiables. En esa línea, en los últimos años han aparecido diferentes trabajos que tratan de proponer procesos de desarrollo de sistemas multiagente. En este artículo se presenta fundamentalmente un análisis de diversos métodos de desarrollo de sistemas multiagente, así como una ampliación de uno de ellos para su adecuación a entornos de tiempo real.

Palabras clave: Metodologías, Agentes, Sistemas Multiagente, Tiempo Real.

1. Introducción

El paradigma de agentes y sistemas multiagente constituye actualmente un área de creciente interés dentro de la Inteligencia Artificial, entre otras razones, por ser aplicable a la resolución de problemas complejos no resueltos de manera satisfactoria mediante técnicas clásicas. Numerosas aplicaciones basadas en este nuevo paradigma vienen ya siendo empleadas en infinidad de áreas [Jennings98], tales como control de procesos, procesos de producción, control de tráfico aéreo, aplicaciones comerciales, gestión de información, comercio electrónico, aplicaciones médicas, juegos, etc..

Uno de los problemas en el área de agentes es el hecho que cada vez son más necesarios métodos, técnicas y herramientas que faciliten el desarrollo de aplicaciones basadas en dicho paradigma. El progreso en la ingeniería del SW en los últimos años se ha realizado gracias al desarrollo de abstracciones más poderosas y naturales para modelar sistemas más complejos. Se podría pensar en el concepto de agente como un avance similar en cuanto a abstracción [Wooldridge98]. Si consideramos a los agentes con el suficiente potencial como un paradigma de la ingeniería del SW, entonces es necesario desarrollar técnicas de ingeniería del SW que sean específicamente aplicables a este paradigma. Por otra parte, la aceptación de métodos en la industria y/o empresa depende de la existencia de las herramientas necesarias que soporten el análisis, diseño e implementación de agentes software.

En los últimos años han aparecido diferentes aproximaciones que tratan de presentar una

metodología apropiada para el desarrollo de sistemas multiagente. En este artículo se presenta una visión generalizada de dichas propuestas analizándolas desde diversos puntos de vista. Posteriormente, el artículo indica las extensiones realizadas sobre una de las aproximaciones más recientes para permitir su aplicación a entornos de tiempo real. De esta forma, se propone la construcción de Sistemas Multiagente de Tiempo Real, donde es necesario conjugar técnicas inteligentes con respuestas en tiempo real en un entorno distribuido.

El resto del artículo está organizado de la siguiente forma: en el punto 2 se presentan distintos trabajos relacionados con el desarrollo de sistemas multiagentes, así como un análisis de los mismos. En el punto 3 se plantea una extensión para el desarrollo de sistemas multiagentes de tiempo real. Finalmente, en el punto 4 se presentan las conclusiones.

2. Trabajos existentes

La aplicación de la Ingeniería del Software al paradigma de agentes, conocida como Ingeniería del Software Orientada a Agente (AOSE – Agent Oriented Software Engineering) [Jennings00] ha generado en los últimos años numerosos trabajos relacionados. Algunos de dichos trabajos vienen recogidos en análisis como los de [Iglesias99] o de [Wooldridge01]. Hasta el momento, los trabajos existentes se han centrado en intentar buscar métodos de desarrollo para poder modelar sistemas reales complejos y con características claramente distribuidas.

Estos trabajos se basan, en su mayoría, en una visión de un sistema como una organización computacional consistente en diferentes entidades interactuando. Cuando se habla de sistemas complejos, el poder identificar los diferentes subsistemas que forman parte del sistema global y las posibles interacciones y dependencias entre ellos es crucial a la hora de poder abordar su construcción.

Entre las aproximaciones existentes podemos citar los trabajos de:

- *Modelado y diseño de sistemas multiagente en BDI* [Kinny96]. Esta aproximación trata de explorar cómo las técnicas de modelado OO se pueden extender para aplicarse a sistemas de agente basados en la arquitectura BDI [Georgeff91] [Georgeff95]. Para ello se proponen técnicas de modelado para describir las perspectivas internas y externas de sistemas multi-agente basados en la arquitectura BDI.
- *Método de Burmeister* [Burmeister96]. En este trabajo se especifican tres modelos para el análisis orientado a agentes de un sistema y se dan ciertas guías para la construcción del sistema a partir de aquí. El método está basado en técnicas orientadas a objetos. Esta aproximación, aunque poco detallada sobre todo en su fase de diseño, introduce elementos clave a detectar y modelar en el proceso de desarrollo de un sistema multiagente: organización, agente e interacciones. Partiendo de esta aproximación, propuestas más recientes tienen también en cuenta estos elementos.
- *MAS-CommonKADS*. La metodología MAS-CommonKADS [Iglesias98] está basada en CommonKADS [Schreiber00], aportando una serie de modelos para desarrollar las fases de análisis y de diseño de sistemas multiagente. La principal característica es la incorporación de técnicas orientadas a objetos a CommonKADS, la cual es tomada como eje fundamental a lo largo de todo el proceso.
- *GAIA*. La metodología GAIA [Wooldridge98] [Wooldridge00] se centra en la idea de que la construcción de sistemas basados en agente es un proceso de diseño organizacional. Cabe resaltar el hecho de que la metodología que proponen es muy genérica, no indicando posibles mecanismos para poder llegar a un sistema directamente ejecutable.
- *DESIRE* [Brazier97]. La principal contribución de DESIRE es que constituye un entorno lo suficientemente expresivo para permitir a los diseñadores de sistemas multiagente centrarse en el diseño conceptual y la especificación de su sistema. El entorno de modelización de alto nivel de DESIRE permite automáticamente generar prototipos de aplicaciones directamente desde la especificación.
- *MASSIVE*. El método de desarrollo de sistemas multiagente MASSIVE (Multi-Agent SystemS Iterative View Engineering) desarrollado en el DFKI [Lind99] está constituido por un conjunto de vistas diferentes del sistema a construir donde el desarrollo que se sigue consiste en una visión iterativa del mismo. En él se combinan procesos de reingeniería junto con un método en cascada mejorado que permite realizar refinamientos.
- *AUML* [Odell00a] [Odell00b]. Este trabajo, desarrollado fundamentalmente por Parunak y Odell, no es en sí una metodología o un método sino que se centra más en intentar adaptar herramientas de desarrollo ya existentes y que están teniendo éxito para aplicaciones industriales reales, como es el caso de UML, tratando de orientarlas hacia el campo de los agentes.

- *Tropos*. En Tropos [Castro02] se presenta una metodología de desarrollo de software basado en agentes mediante extensiones de UML y empleando un entorno de modelado denominado *i** [Yu96]. El concepto principal sobre el que se desarrolla el proceso de análisis y modelado es el de actor, así como sus objetivos y posibles dependencias con otros actores.
- *MaSE* (Multiagent System Engineering) [Wood00]. Es una metodología desarrollada en el Air Force Institute of Technology. Dicha metodología trata de cubrir todas la etapas en el proceso de construcción de un sistema multiagente, partiendo de la especificación del mismo hasta su implementación. Dispone de un lenguaje de especificación basado en UML+OCL [Robinson00] y una herramienta de desarrollo denominada AgentTool que trata de cubrir la totalidad de fases de la metodología pero que por el momento se queda en sólo una parte de ellas.
- *MESSAGE* (Methodology for Engineering Systems of Software Agents) [EURESCOM00] [EURESCOM01]. Ésta es una metodología orientada a agentes la cual incorpora técnicas de ingeniería del software cubriendo el análisis y diseño de sistemas multiagente. La metodología provee un lenguaje, un método y unas guías de cómo aplicar la metodología, centrándose en las fases de análisis y diseño y lanzando ideas sobre el resto de etapas como implementación, pruebas e implantación.

A continuación se presenta un análisis y comparación de los distintos métodos existentes, aunque es necesario remarcar que las aproximaciones más recientes se nutren en parte de anteriores trabajos. El análisis de las aproximaciones se centrará fundamentalmente en las etapas de análisis y diseño en sistemas multiagente, ya que son las fases que contemplan la práctica totalidad de las propuestas.

La adopción de una orientación dirigida a agentes en un método de desarrollo conlleva según [Jennings01] tener en cuenta tres elementos clave: agentes, interacciones y organizaciones. De acuerdo a esta idea, la práctica totalidad de aproximaciones existentes tratan de dar cabida a estas abstracciones en sus propuestas. El estudio se realiza fundamentalmente a tres niveles: a nivel conceptual, a nivel general del método y a nivel de las fases de desarrollo consideradas. A continuación se presentan cada uno de ellos.

2.1 A nivel conceptual

Conceptualmente la mayoría de propuestas introducen, a lo largo de su desarrollo, términos muy similares. De esta forma, conceptos como *objetivo*, *tarea*, *interacción*, *rol*, *responsabilidad*, *organización*, *componente* y evidentemente *agente* aparecen en la mayoría de los trabajos de una forma u otra, y su significado aunque con matices suele ser bastante uniforme. La mayoría de las definiciones son bastante parecidas y vienen a representar en el fondo las mismas ideas. De este conjunto de conceptos destacamos los de *agente*, *rol* e *interacción* pues vienen a constituir aspectos fundamentales en un sistema multiagente.

Por lo que respecta, ya más en concreto, al concepto de *agente*, éste es un concepto que se ha definido en multitud de ocasiones. Algunos de los métodos existentes plantean definiciones propias de agente, lo cual remarca quizás más la controversia existente en este aspecto. Aunque en la mayoría de ocasiones las definiciones únicamente varían en pequeños matices. Podemos citar algunos ejemplos como en el caso de MaSE, donde un agente es visto como una abstracción útil para resolver problemas en dominios específicos. Según esto, un agente vendría caracterizado por lo siguiente:

- Los agentes son entidades que están distribuidas.
- Los agentes son entidades autónomas, dirigidas por objetivos y sociales.
- Los agentes comparten información con otros agentes de forma interactiva, conformando sistemas multiagente.

Por otro lado, por ejemplo en MESSAGE, un agente es definido a partir de un conjunto de características que deben tener aquellas entidades etiquetadas como agentes. Estas características son:

- Un agente tiene cierto conocimiento del mundo donde vive.
- Un agente es responsable de alcanzar y mantener ciertos objetivos que caracterizan su conducta.
- Un agente es capaz de observar el estado de ciertos objetos en el entorno y de sentir ciertos eventos.
- Las interacciones entre agentes son descritas en términos de acciones comunicativas.
- Un agente puede ejecutar acciones que afecten a los objetos del entorno.

Otro concepto, el de *rol*, es un concepto muy empleado sobre todo para poder indicar posibles cambios de conducta de un agente. Por ejemplo, en MaSE un rol es definido como una función de una entidad (agente) que contiene objetivos del sistema y que tiene la responsabilidad de cumplir. De la misma forma, en GAIA un rol es definido como una descripción abstracta de una función esperada de la entidad, en este caso un agente. El rol también aparece en MASSIVE con un sentido muy parecido.

En MESSAGE el término rol es empleado para describir características que un agente toma debido a un motivo (por ejemplo, una condición que se cumple en un momento dado). En definitiva el concepto de rol es empleado para modelar diferentes comportamientos que posteriormente un agente dado podría llevar a cabo. El rol también permite establecer cierta organización entre los agentes que componen un sistema al otorgarse ciertos privilegios a determinados roles frente a otros.

Finalmente, otro término muy empleado en los diferentes métodos presentados es el de *interacción*. Una interacción es empleada para modelar el comportamiento social de los agentes. En muchos métodos existen modelos o vistas específicos para modelar las interacciones existiendo en la mayoría de los casos bastante similitud en el significado otorgado a este término. Únicamente citar el caso de MASSIVE donde el término de interacción es considerada una forma generalizada de resolución de conflictos no limitada a una forma particular como puede ser la comunicación.

2.2 A nivel general del método

A nivel general existen ciertos aspectos que pueden ser analizados sobre los trabajos existentes hasta el momento (ver resumen en la tabla 1):

- La mayoría de propuestas intentan cubrir fundamentalmente las etapas de análisis y diseño de sistemas multiagente. El resto de etapas no son consideradas o bien se deja total libertad para su desarrollo. TROPOS si que incluye la etapa de implementación, mientras que MaSE la propone pero no la desarrolla.
- Un aspecto interesante es el de proveer guías de desarrollo para las fases no consideradas. En este punto, MAS-CommonKADS da pequeñas pinceladas de los procesos de implementación e implantación. En MaSE por su parte se habla de la generación automática de código como último paso de su propuesta pero éste no está concretado. Finalmente, en MESSAGE se incorporan guías a considerar para el resto de fases típicas en el proceso de desarrollo de software.
- El uso de UML para especificar tanto los aspectos dinámicos como estáticos es frecuente. Este aspecto es generalizado en las propuestas más recientes. Adicionalmente, también se suelen emplear esquemas (a modo de tablas) para especificar los atributos o características de ciertas entidades (agentes, roles, interacciones).
- En principio el tipo de sistemas a desarrollar no son dinámicos (aparición o desaparición de agentes considerados o no en el desarrollo) y por tanto aparentemente son cerrados. Esta afirmación es segura en ciertos métodos como por ejemplo MaSE donde se hace explícito. En otros métodos no es comentado nada al respecto, pero no se han observado mecanismos que permitan la incorporación dinámica de nuevas entidades.
- Por lo que respecta a la orientación de la propuesta, la mayoría toma elementos de la orientación a objetos. Cabe resaltar en algunos casos la orientación del proceso de análisis hacia el objetivo, como en TROPOS y MaSE. En este punto destacar la propuesta más variable de MESSAGE, existiendo distintas posibilidades según los autores, dependiendo de si se empieza por especificar el modelo de organización o el modelo de objetivos/tareas.
- La mayoría excluyen los agentes móviles como tipo de agente a tener en cuenta en el proceso de desarrollo de los sistemas. La inclusión de esta propiedad acarrea numerosos condicionantes que complicarían el desarrollo y no son planteados de momento. Destacar que MaSE plantea en trabajos más recientes la posibilidad de su inclusión.
- No se hace ningún tipo de comentario respecto a la posibilidad de incorporar restricciones temporales en la ejecución de tareas y/o interacciones, ya sean estrictas o no. Esto impide en un primer análisis el empleo de estos métodos para problemas en entornos de tiempo real.
- La existencia de herramientas de desarrollo asociadas no es ni mucho menos un aspecto generalizado. MaSE incorpora una herramienta de desarrollo (agentTool) para facilitar el proceso de desarrollo, aunque por el momento no incorpora la totalidad de etapas. En MAS-CommonKADS existe una herramienta conocida como AgentEditor, mientras que en TROPOS se comenta que se está en fase de desarrollo de una herramienta que contemple todo el proceso. Finalmente, MESSAGE propone una guía de recomendaciones sobre herramientas útiles para MESSAGE. Básicamente, en MESSAGE se propone hacer uso de herramientas para UML, como Rational Rose o la herramienta metacase MetaEdit+ (http://www.metacase.com/).
- En cuanto a posibles limitaciones impuestas por los propios métodos, comentar las restricciones en cuanto el número de clases de agente a desarrollar de MaSE o GAIA, el resto de métodos no comenta nada al respecto pero es cierto que dada la dificultad del proceso de depuración de un sistema multiagente, un elevado número de agentes complicaría notablemente su realización.
- Otras limitaciones se refieren al propio método, pues ocurre en ocasiones que éste está falto de un mayor detalle en los pasos que presenta, caso del método propuesto por Burmeister que es poco conciso. En otros casos la falta de ejemplos complica el entendimiento de los procesos de

modelado, caso de GAIA, aunque este aspecto se podría generalizar a la totalidad de las propuestas.

- Por lo que respecta a la caracterización de las interacciones entre agentes o roles es en ocasiones independiente de cualquier modelo, como en el caso de MaSE o GAIA. Los trabajos de Bursmeister, MAS-CommonKADS y MASSIVE están orientados al empleo de un lenguaje concreto como KQML, quizás porque era la propuesta más extendida en su momento. Por lo que respecta a los casos de TROPOS y MESSAGE se emplea en la especificación AUML, esto es, protocolos de interacción FIPA especificados mediante UML.
- Finalmente, en lo que se refiere al empleo de una arquitectura de agente concreta, destacar que existen distintas alternativas. Diversos métodos (GAIA, MaSE, Bursmeister y MAS-CommonKADS) optan por cierta independencia y la posibilidad de selección entre varias opciones. Otras propuestas optan por una orientación a una arquitectura concreta, caso del método de Kinny con BDI. TROPOS marca en principio un proceso independiente pero opta en su fase de implementación por el empleo de JACK. En el caso de MESSAGE, se permite un diseño independiente de una arquitectura concreta o bien un diseño orientado, en su caso, hacia el modelo de agente empleado en JADE.

2.3 A nivel de fases consideradas

Tal y como se ha comentado en el punto anterior, las fases de análisis y diseño son las fases consideradas en la totalidad de los métodos analizados. En dichas fases se desarrollan todos los procesos propuestos de construcción de sistemas multiagente.

En este punto se pasa a analizar las subfases, modelos o vistas que se realizan en cada una de las propuestas. En la tabla 2 se recoge de forma resumida dicha información. Sobre ella se pueden realizar diversas consideraciones que a continuación se exponen.

Análisis

Por lo que respecta a la fase de análisis, se pueden destacar cuatro elementos o aspectos que tratan la mayoría de propuestas, éstos son, la identificación de objetivos, modelado de la organización, detección de los agentes necesarios y establecimiento de las interacciones necesarias.

Respecto a estos puntos se puede resaltar lo siguiente:

- La identificación de objetivos a conseguir en el sistema se repite en todas las propuestas aunque en diferentes vistas o modelos. En algunos casos se desarrolla un modelo específico de objetivos como en MaSE y MESSAGE. En MESSAGE esta etapa incorpora la identificación de tareas y subtareas asociadas a dichos objetivos, mientras que en MaSE las tareas se construyen entre las fases de captura de objetivos y transformación de roles. En otros casos los objetivos quedan enmascarados dentro de la especificación de otras entidades como roles (GAIA), tareas (MAS-CommonKADS, MASSIVE) o agentes (Burmeister).
- El modelado de la organización existente en el sistema dispone de vistas específicas en distintas aproximaciones. En todas ellas se trata de dotar de una estructura organizacional a las entidades que componen el sistema. En ocasiones dicha organización es representada a través de la especificación de los roles en un modelo o vista de roles. Destacar el caso de MaSE, donde no existe un modelado de la organización como tal, aunque la orientación hacia el objetivo del análisis en esta propuesta puede dar sentido a este hecho. Los posibles recursos externos que suelen aparecer en este modelo pueden ser encapsulados en MaSE como un rol que actúa como interfaz con el resto de roles que pretenden acceder a dichos recursos.
- Este análisis de la organización en muchas propuestas está condicionado completamente por una visión funcional del sistema, dejando de lado una visión estructural o arquitectónica del mismo. El sistema se construye en base a la funcionalidad que debe tener, especificada por medio de conceptos como objetivos, roles, responsabilidades, tareas o servicios, dejando de lado la forma o estructura del sistema. La forma de un sistema depende de su propia estructura, condicionada por aspectos como por ejemplo, la plataforma en la que debe funcionar el sistema. Estas dos visiones (función y forma) deben ir construyéndose y evolucionando de forma paralela y deben ser compatibles una con respecto a la otra.

	KINNY	GAIA	BURMEISTER	MAS-C. KADS
Fases[*] consideradas	AD	AD	AD	AD
UML	Empleo de OMT	–	–	Empleo notación OO en algunas fases
Tipo Sistema	Cerrado, no dinámico	Cerrado, no dinámico	Cerrado, no dinámico	Cerrado, no dinámico
Orientación	Extensiones OO	–	Extensiones OO	Ing. del conocimiento
Movilidad	NO	NO	NO	NO
Tiempo Real	NO	NO	NO	NO
Herramienta desarrollo	–	–	NO	SI, AgentEditor
Limitaciones	Sólo agentes BDI	Diseño alto nivel Máx. 100 tipos de agentes	Poco detallada	Complejidad Common-KADS
Guías resto	NO	NO	NO	Pequeños trazos
Caract. interacciones	No indicado	Modelado de interacciones independiente	Empleo KQML	Empleo KQML
Arquitectura agente	BDI	Independiente	Independiente	Independiente Empleo arq.propia
	MASSIVE	**TROPOS**	**MaSE**	**MESSAGE**
Fases[*] consideradas	ADI	ADI	AD(I)	AD
UML	SI	SI	SI (UML+OCL)	SI
Tipo Sistema	Cerrado, no dinámico	Cerrado, no dinámico	Cerrado, no dinámico	no dinámico
Orientación	Iterative View Engineering	Basada en el Objetivo Modelo i*	Basada en el Objetivo	Ing. Software + Teoría agentes
Movilidad	NO	NO	NO (en un futuro)	NO
Tiempo Real	NO	NO	NO	NO
Herramienta desarrollo	–	En un futuro	Si AgentTool	Guía herramientas a emplear
Limitaciones	–	Sin finalizar	Max. 10 clases de agente	–
Guías resto fases	-	Req. iniciales e implementación	Implementación en un futuro	SI
Caract. interacciones	Empleo KQML	AUML	Independiente, no broadcast	AUML
Arquitectura agente	Independiente	Uso de JACK (BDI)	Independiente	Según diseño

Tabla 1 Comparativa general

[*] *A – Análisis, D – Diseño, I - Implementación*

- El inicio en el estudio de las interacciones que derivan en posibles cooperaciones o negociaciones entre agentes también suele ser una fase del proceso de análisis en la mayoría de propuestas, salvo el caso de MASSIVE que lo incorpora en la etapa de diseño. Realmente el modelado de las interacciones tiene una continuación en todos los casos en el diseño, en la fase de análisis se trata únicamente de identificar y dar forma a las posibles interacciones del sistema. Esto conlleva la existencia de vistas o modelos de interacción que dotan de herramientas para gestionar este aspecto. Por ejemplo, entre otros en MESSAGE, GAIA o MASCommonKADS existe un modelo específico. En el caso de MaSE esto se realiza en la fase de aplicación de casos de uso.
- Por último, la detección de los agentes existentes en el sistema se podría decir que se encuentra a caballo entre el proceso de análisis y diseño. La especificación de un agente consiste fundamentalmente en situarlo dentro de la organización del sistema e indicar sus responsabilidades, capacidades y recursos, por medio de la asignación de roles, objetivos y tareas identificados con anterioridad. Un modelo específico de agente existe en propuestas como GAIA (ya en diseño), MAS-CommonKADS, MaSE o MESSAGE. En dichos modelos se suele emplear una notación gráfica tipo UML junto con esquemas que recojan los aspectos que modelan al agente en cada caso.

Diseño

Por lo que respecta a la fase de diseño, según [Jacobson99] en esta fase se trata de modelar el sistema y encontrar su forma, de tal manera que de soporte a todos los requisitos que se le suponen y que fundamentalmente son resultado de la fase previa de análisis.

En el caso de los sistemas multiagente el diseño se centra en el modelado de la arquitectura del sistema mediante refinamiento de los modelos del análisis, el modelado de los componentes internos de los diferentes agentes y de sus interacciones. Al igual que en la fase de análisis respecto a estos puntos se puede resaltar lo siguiente:

- El modelado de la arquitectura del sistema viene dado en la mayoría de los casos por una visión general del sistema. Esto se realiza en algunas propuestas mediante un refinamiento de la organización y de las interacciones identificadas en la fase de análisis. Este refinamiento, en general, permite el perfeccionamiento de los tipos de agentes necesarios, permitiéndose en muchos casos la adición de nuevos tipos de agentes que no hubiesen sido aún detectados o la fusión de agentes muy relacionados en uno sólo. En MAS-CommonKADS, el diseño de red permite una visión uniforme del sistema, en GAIA una visión similar se da en el modelo de conocimiento. En el caso de MASSIVE la vista de sistema permite modelar todos aquellos aspectos relacionados con el sistema en su conjunto.
- El modelado de los componentes internos de los diferentes agentes presenta diferentes visiones en los trabajos analizados. En algunos casos se presentan diversas alternativas según las características de un agente concreto. Para ello se presentan diversos patrones de arquitecturas, como en MaSE. En otros casos como en MESSAGE se desarrolla el diseño sobre una arquitectura genérica que poco a poco va orientándose, o bien, se puede diseñar directamente sobre una arquitectura específica siguiendo la propuesta de la plataforma JADE. Finalmente, algunas propuestas optan por el desarrollo de arquitecturas específicas como Kinny con la arquitectura BDI.
- El diseño de las interacciones presenta también la opción de un diseño independiente de los modelos de interacción existentes, como el caso de MaSE donde a la hora de especificar las interacciones entre roles (o agentes), en el diseño se utilizan dos diagramas (autómatas de estados finitos) uno para el emisor y otro para el receptor. Mientras en otros trabajos se emplean en el diseño protocolos de interacción predefinidos en AUML, esto es, protocolos FIPA especificados mediante UML. Esto, en cierto modo, facilita la comprensión de los modelos obtenidos al emplear una notación bastante extendida. Finalmente, en otros trabajos el detalle en el diseño de las interacciones es muy bajo, impidiendo una comprensión total de dicho proceso.

Destacar que en algunos casos la fase de diseño se limita a un refinamiento de modelos del análisis sin especificar nada más, o bien se muestra un diseño de muy alto nivel que necesita de una mayor profundidad para poder implementar el sistema.

2.4 Consideraciones

En este trabajo no se trata de determinar si una aproximación es mejor o peor que otra, sino más bien establecer que métodos pueden resultar más adecuados para ser empleados como base de futuros desarrollos. En este sentido, es evidente que las últimas propuestas son más refinadas y completas al recoger los progresos que se han ido desarrollando en trabajos anteriores. Éste es el caso, por ejemplo, de métodos como MaSE, Tropos o MESSAGE.

	VISTAS O FASES EN ANÁLISIS	*VISTAS O FASES EN DISEÑO*
Kinny	Modelo de Agente Modelo de Interacción Modelos de Creencias Objetivos y Planes	Refinamiento de los modelos del análisis
GAIA	Modelo de Roles Modelo de Interacción	Modelo de Agente Modelo de Servicios Modelo de Conocimiento
Burmeister	Modelo de Agente Modelo de Organización Modelo de Cooperación	Refinamiento de los modelos del análisis
MAS-C. Kads	Fase de Conceptuación Modelo de Organización Modelo de Agente Modelo de Tareas Modelo de la Experiencia M. Comunicación y Coord.	Diseño de Red Diseño de Agentes Diseño de Plataforma
MASSIVE	Vista de Tareas Vista de Entorno Vista de Sistema	Vista de Roles Vista de Interacción Vista de Sociedad Vista de Arquitectura Vista de Sistema
TROPOS	Requerimientos Iniciales Requerimientos Finales (Modelo actores y organización)	Diseño Arquitectónico (Refinado y creación de agentes) Diseño Detallado (Modelo interno de agentes)
MaSE	Captura de Objetivos Aplicación casos de uso Transformación de Roles Creación clases de agente	Construcción de conversaciones Ensamblado clases de agente Diseño del sistema
MESSAGE	Modelo de Organización Modelo de Objetivos/Tareas Modelo de Agente Modelo de Interacción Modelo de Dominio	Refinamiento de modelos (Organización e Interacción) Modelo interno de agentes (Elección de Arquitectura)

Tabla 2 Comparativa de vistas en análisis y diseño

La relativa juventud de estos trabajos conlleva que no exista un estándar de facto para el desarrollo de sistemas multiagente y que por tanto actualmente surjan nuevas propuestas o se modifiquen o amplíen anteriores aproximaciones.

3. Extensiones para Tiempo Real

En el punto anterior se ha planteado la falta de consideración de aspectos temporales en la totalidad de las propuestas comentadas. Si se desease aplicar alguna de estas metodologías en entornos de tiempo real nos encontraríamos con la imposibilidad de modelar determinados conceptos. De esta forma, en este punto se plantea una extensión de una de las propuestas existentes para su posible aplicación en entornos de tiempo real. La propuesta seleccionada es MESSAGE. El porqué de esta elección se basa entre otras razones en que MESSAGE es, de entre los trabajos existentes, el que cubre más aspectos en lo que respecta al proceso de análisis, mientras que el diseño destaca frente al resto por su flexibilidad. Por otro lado, el empleo de UML, la disponibilidad de ejemplos desarrollados y la existencia de guías orientativas para el resto de fases del proceso de desarrollo, hacen sumamente interesante esta metodología.

Como consecuencia de la extensión de MESSAGE, se propone un método de desarrollo para sistemas multiagente de tiempo real denominado RT-MESSAGE [Julian02a] [Julian02c]. Dicho método, por tanto, está basado en la metodología de desarrollo de sistemas multiagente MESSAGE y en el método RT-UML [Douglass99] en lo que se refiere al desarrollo de sistemas de tiempo real. La idea fundamental en el método propuesto es proveer de los elementos necesarios para a partir de la especificación de requerimientos de un problema al uso, se pueda obtener directamente un prototipo ejecutable del sistema a desarrollar.

Es interesante resaltar el porqué MESSAGE por si sólo no permite el desarrollo de sistemas multiagente de tiempo real. Esto es debido principalmente a que:

- Al igual que el resto de aproximaciones, esta metodología es de propósito general, por tanto, no es contemplado ningún aspecto relacionado con el desarrollo de sistemas de tiempo real.
- Su etapa de diseño de bajo nivel debería adaptarse al caso concreto de agentes de tiempo real y en concreto, a una arquitectura específica de agente de tiempo real y a sus particularidades.
- Es necesaria una ampliación de diferentes modelos de MESSAGE, como el de organización e interacción de modo que recojan aspectos temporales.
- Es necesario ampliar los tipos de objetivos en el modelo de objetivos de tal forma que se tenga en cuenta la criticidad de los mismos y, por tanto, también las tareas asociadas al cumplimiento de dichos objetivos.

La imposibilidad de especificar sistemas de tiempo real hace necesaria su adaptación y extensión para emplearla como base para el desarrollo de sistemas multiagente de tiempo real.

El método de desarrollo RT-MESSAGE, tal y como se ha planteado, está orientado a la construcción de sistemas multiagente de tiempo real. Los sistemas a desarrollar se basan en la plataforma de sistemas multiagente de tiempo real SIMBA [Soler02] [Julián02b]. El principal componente de SIMBA es la arquitectura de agente de tiempo real ARTIS [Botti99]. Por medio de ARTIS se dispone de una arquitectura de agente donde confluyen características propias de sistemas de tiempo real estricto con componentes inteligentes. Más información acerca de los componentes y funcionalidades de una agente ARTIS se puede obtener en [Botti99], [Soler00] y [Julian02a]

En el método propuesto únicamente se consideran las actividades de Análisis, Diseño e Implementación. Por lo que respecta a los requerimientos, se considera que éstos no difieren respecto a otros paradigmas y por tanto no es significativo en este trabajo. En cuanto a la actividad de pruebas, no ha sido, de momento, objeto del presente trabajo. De esta forma, de manera resumida, en el análisis del sistema se trata de especificar de la forma más apropiada posible el problema a resolver, para ello se emplean los modelos propuestos por MESSAGE ampliados para poder modelar las características propias de sistemas de tiempo real. Por lo que respecta al diseño, toma como entrada todos los artefactos generados en el análisis, centrándose en la transformación de las entidades especificadas en dichos artefactos en entidades computacionales. Para ello, se emplea la arquitectura de sistema multiagente de tiempo real SIMBA, la cual nos permite diseñar las interacciones, organización y estructura interna de los agentes de tiempo real del sistema. Finalmente, proponemos la implementación del diseño realizado, esto es, de los componentes, interacciones y demás elementos diseñados. Para ello, se dispone de una herramienta de desarrollo que permite la obtención de un prototipo directamente ejecutable del sistema.

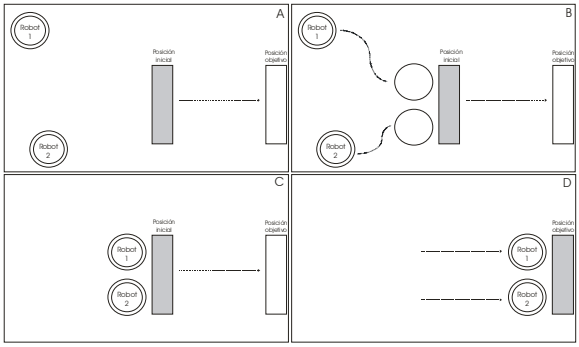

Figura 1 Posible escenario del ejemplo

Con el objeto de ilustrar las diferentes fases del método de desarrollo, se plantea un sencillo ejemplo que consiste en un sistema que gestione a diferentes robots situados en un determinado entorno, que traten de mover un objeto de forma conjunta de una posición inicial a una posición objetivo. Para simplificar el problema se asumirá que uno de los robots adquirirá un rol de supremacía respecto al resto con el objetivo de distribuir las correspondientes órdenes de movimiento.

La funcionalidad del sistema se puede resumir en la siguiente secuencia de etapas que deben ser tenidas en cuenta en el proceso global:

1. Establecimiento inicial de los componentes del equipo de robots.
2. Realizar la distribución de los robots alrededor

del objeto a desplazar.
3. Desplazamientos de los robots a las posiciones de inicio.
4. Inicio del proceso de desplazamiento del objeto.
5. Desarrollo de un control del proceso de desplazamiento para poder tomar las acciones correctoras adecuadas.

En la figura 1 se puede observar una posible secuencia de los pasos a seguir partiendo de un posible escenario inicial donde dos robots se ponen de acuerdo para desplazar conjuntamente un objeto de la posición inicial a la posición objetivo.

3.1 Análisis

El proceso de análisis de un sistema identifica todas las características del mismo que son esenciales. Esto permitiría un mejor entendimiento del sistema y facilitaría el diseño de la solución al problema.

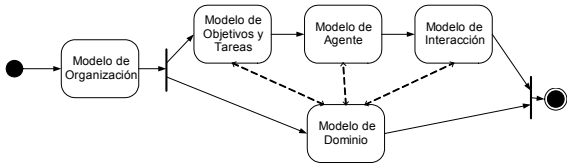

Figura 2 Diagrama de la actividad de análisis

El análisis en RT-MESSAGE está formado por el mismo conjunto de modelos de MESSAGE. De esta forma a partir de los modelos especificados en MESSAGE se plantean ampliaciones para poder emplear dicha metodología en su fase de análisis para construir sistemas multiagente para entornos de tiempo real. En la figura 2 se muestra la secuencia de pasos en la construcción de los modelos que constituyen la actividad del análisis. En los siguientes puntos se presentan de forma más detallada dichos modelos.

Modelo de Organización

Este modelo permite definir la estructura y la conducta de un grupo de agentes que trabajan de forma conjunta para alcanzar ciertos objetivos. Vendría a representar la organización en términos de suborganizaciones relacionadas, proveyendo una abstracción para intentar entender la estructura completa del sistema multiagente.

Uno de los elementos fundamentales de este modelo es el diagrama de organización donde quedan representadas las distintas entidades del sistema. En la figura 3 se muestra el diagrama de organización para el sistema del ejemplo anteriormente planteado.

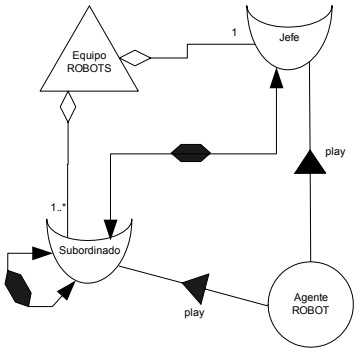

Figura 3 Diagrama de organización

Las principales aportaciones realizadas sobre el modelo de organización de MESSAGE se pueden centrar fundamentalmente en el reforzamiento de la visión de los aspectos de conducta del sistema. Se ha incluido la posibilidad de emplear diagramas en la especificación de conductas que permiten modelar el tiempo como elemento crucial (diagramas de secuencia, estados y actividad).

Además, se ha incorporado la definición de una lista de eventos externos a controlar por el sistema con posibles restricciones temporales. En la tabla 3 se muestra la lista de eventos para el ejemplo del equipo de robots donde quedan expresadas a alto nivel las restricciones temporales del sistema a desarrollar. Por último, también se ha modificado el esquema de propósitos de una organización permitiendo la definición de restricciones temporales asociadas a la organización.

Evento	Descripción	Dirección	Patrón de llegada	Tiempo mínimo entre ocurrencias
ev_obstaculo	Obstáculo detectado	agente robot	periódico	300 ms
ev_estado_bat	Estado batería	agente robot	periódico	10000 ms
ev_fallo_hw	Fallo general en el robot	agente robot	periódico	--

Tabla 3 Lista de eventos temporales

Modelo de Objetivos/Tareas

Este modelo trata de responder a las preguntas de *¿porqué, quien y cómo?* a lo largo del proceso de análisis. El *¿porqué?* se refiere a los objetivos que se definan para el sistema, el *¿quién?* hace referencia a los agentes a los cuales se les responsabiliza de la consecución de los objetivos y el *¿cómo?* es el conjunto de tareas definidas para conseguir los objetivos.

Nombre	Evitar obstáculo
Definición	Esta tarea determina la siguiente acción atómica para evitar chocar con un obstáculo en función de la situación del robot.
Realizada por	Agente Robot
Entradas	Ultimo estado disponible de los bumpers y sónares
Salidas	Acción atómica
Pre-condiciones	--
Post-condiciones	--
¿Es crítica?	Sí
¿Es periódica?	Sí
Periodo	Cada tres lecturas de datos (300 ms)
Plazo Máximo	100 ms

Tabla 4 Esquema de tarea

Las principales modificaciones realizadas en el modelo de objetivos/tareas de MESSAGE consisten en la inclusión de una clasificación de los objetivos en el sistema, atendiendo a criterios temporales. Esto conlleva también a la existencia de distintos tipos de tareas en el modelo. Para su expresión se han ampliado los esquemas de objetivos y de tareas presentes en MESSAGE, de tal forma que incorporan características de tiempo real (ver ejemplo de esquema de especificación de tarea en la tabla 4).

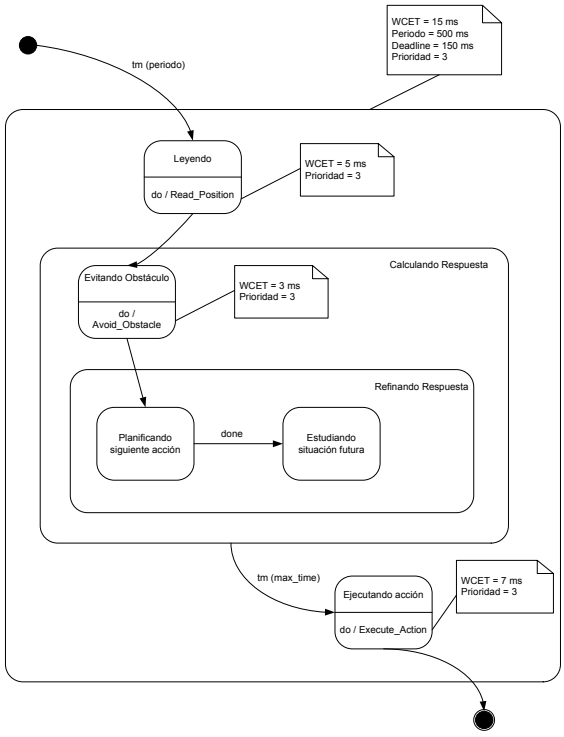

Figura 4 Diagrama de comportamiento

Por otro lado, para poder reforzar la especificación de conductas de tipo temporal se ha incluido la posibilidad de emplear diagramas que incorporan distintas marcas de tipo temporal con las que poder modelar estos aspectos. En la figura 4 se muestra la especificación del comportamiento de una tarea en el ejemplo, en ella quedan expresadas distintas restricciones de tipo temporal asociadas a la tarea en cuestión. Por último, se ha mostrado como poder especificar conductas de métodos usuales en el área de la Inteligencia Artificial de Tiempo Real mediante sus correspondientes diagramas.

Modelo de Agente

El modelo de agente está formado por un conjunto de agentes y roles que son descritos de forma individual. Cada elemento del modelo de agente reúne la información específica de un agente o rol incluyendo sus relaciones con otras entidades. Para especificar cada agente o rol se emplean unos esquemas como el de la tabla 5. En dicho esquema quedan representados distintos aspectos de caracterización de un agente o rol. En el caso concreto del ejemplo se muestra el esquema del rol jefe el cual sería asumido por uno de los robots del sistema.

Esquema de Rol: Jefe
Identidad: Controla al equipo de robots, enviando órdenes al resto de componentes y solicitando donde se encuentran para poder estimar nuevas acciones conjuntas
Requerimientos de los agentes: Un agente dispone de inicio de las capacidades que el rol si dispone, es por tanto una decisión de diseño indicar el robot que es considerado jefe
Interacciones con el entorno: *Entrada sensores:* Mensajes enviados por subordinados: respuestas a interacciones iniciadas por el rol *Conocidos:* Agentes en el sistema que dispongan del rol subordinado *Recursos:* – *Acciones:* Envío de mensajes de: - petición de formación de equipo - solicitar posiciones a robots del equipo - envío de ordenes a robots del equipo: ir-a, parar, empujar
Estado mental y conducta: *Propósito:* Establecer equipo (soft-goal), planificar trabajo, desplazarse a posición, controlar acciones individualmente *Conducta:* Obtener agentes disponibles, enviar propuesta, formalizar equipo, obtener posiciones equipo, recalcular posición objeto, estimar acciones, comunicar acciones, activar acciones individuales, detectar micromundo, desplazarse a, monitoriza acción, comprueba y corrige acción. *Conocimiento y creencias:* - Conocimiento del estado del micromundo (otros robots), con marcas temporales asociadas.
Información adicional: La tarea *enviar propuesta* relacionada con el objetivo de *establecer equipo* tiene un plazo máximo especificado.

Tabla 5 Esquema de rol

Las aportaciones principales en el modelo de agente de MESSAGE han consistido en la ampliación de los criterios de identificación de agentes atendiendo a consideraciones estructurales, la incorporación al modelo del concepto de agente de tiempo real (estricto y no estricto) y la posibilidad de la existencia de roles con restricciones temporales no críticas. Para su expresión se han ampliado los correspondientes esquemas de agente y rol que se proponen en este modelo.

Modelo de Dominio

El modelo de dominio permite definir los conceptos específicos del dominio con el que los agentes deben trabajar. La forma en que se expresa esto es mediante un modelo donde se muestran las clases necesarias del dominio, los atributos de cada clase y las relaciones entre dichas clases. En otros trabajos, como en [Bergenti01], se presenta algo similar mediante un diagrama de ontología donde las clases especificadas representan entidades comprendidas en la ontología del problema. En la figura 5 se presenta el diagrama de dominio del ejemplo planteado, donde quedan representadas fundamentalmente las clases oportunas.

Las aportaciones en el modelo de dominio de MESSAGE han consistido en delimitar en este caso el modelo a únicamente conocimiento de tipo factual, y la adición de información temporal asociada a aquellos datos que así lo requieran. En relación con este aspecto, se ha presentado una clasificación de la información temporal en función de diferentes criterios. Por último, se recomienda la posible división en subdominios con el objetivo de facilitar la especificación de un entorno complejo, como pueden ser ciertos problemas de tiempo real.

Modelo de Interacción

Este modelo captura la forma en que los agentes (o roles) intercambian información con otros (y también con el entorno). El modelo de interacción estará constituido por un conjunto de interacciones de alto nivel, un conjunto de protocolos de interacción más detallados y un conjunto de descripciones de mensajes. La especificación de una interacción se iniciaría mediante la confección de un sencillo esquema de interacción donde también expresar posibles restricciones temporales. En la tabla 6 se muestra el esquema de una interacción en el ejemplo que nos serviría para indicar la necesidad de comunicación por parte de las entidades del sistema para la formación inicial de los componentes del equipo.

Las principales aportaciones en el modelo de interacciones de MESSAGE han consistido en ampliar el espectro de posibles interacciones a modelar, incorporando aquellas que tienen en cuenta el tiempo como factor importante o esencial. Para ello, se han ampliado los esquemas de interacciones, se ha incorporado información de tipo temporal en los esquemas correspondientes, con la posibilidad de especificar conceptos tales como la utilidad de la interacción.

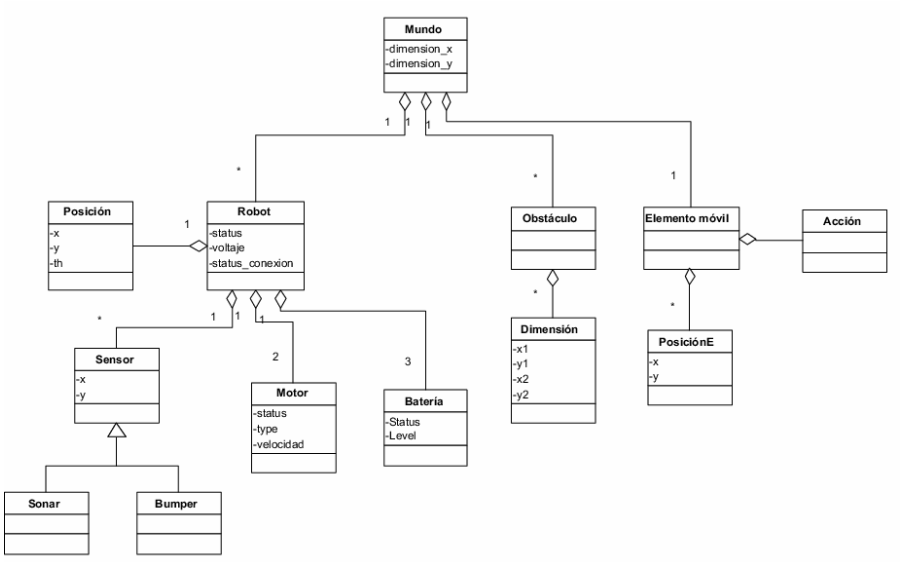

Figura 5 Diagrama de dominio

Motivación	Solicitar formación de equipo
Iniciador	Jefe
Colaboradores	Todos los agentes robots disponibles
Entradas	Petición de formación de equipo
Salidas	Aceptación o no de la solicitud
Proceso	Formación equipo, iniciar proceso de resolución del objetivo del equipo
Restricciones	La realización de esta interacción tiene una duración máxima de 5 sg, plazo en el cual si no se ha cumplido supondría reiniciar el proceso hasta un máximo de 3 ocasiones

Tabla 6 Esquema de interacción

3.2 Diseño

El diseño se construye a partir de los artefactos obtenidos en los diferentes modelos de la actividad de análisis. En este caso el diseño que se propone correspondería al conjunto de agentes identificados que tengan responsabilidades de tiempo real. Es por ello que nos centraremos exclusivamente en el diseño de este tipo de agentes. El diseño puede dividirse en dos partes o subprocesos: diseño arquitectónico y de bajo nivel, los cuales se plantean a continuación.

Diseño arquitectónico

Consiste en el diseño de aquellos aspectos que afectan al sistema como un todo. El objetivo en este punto es diseñar el sistema desde un punto de vista de alto nivel, número y tipo de los agentes del sistema, comunicación y protocolos inter-agente.

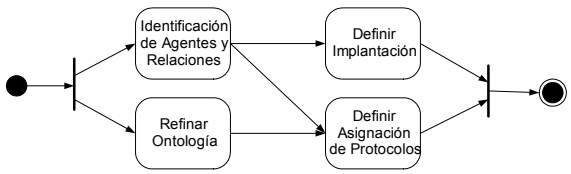

Figura 6 Pasos del diseño arquitectónico

Los pasos a realizar son (ver figura 6):

- *Identificación de Agentes ARTIS*. Consiste en determinar a partir del análisis realizado, los agentes del sistema.
- *Definición y asignación de protocolos*. El refinado de las interacciones detectadas en el modelo de interacción consiste en determinar principalmente las tareas asociadas a la interacción y en la especificación de los mensajes y su contenido.
- *Definir la implantación*. Llegado este punto lo que se pasa a definir son las características de la plataforma SIMBA que se requiere. En la figura 7 se muestra el diagrama de implantación desarrollado en el ejemplo.

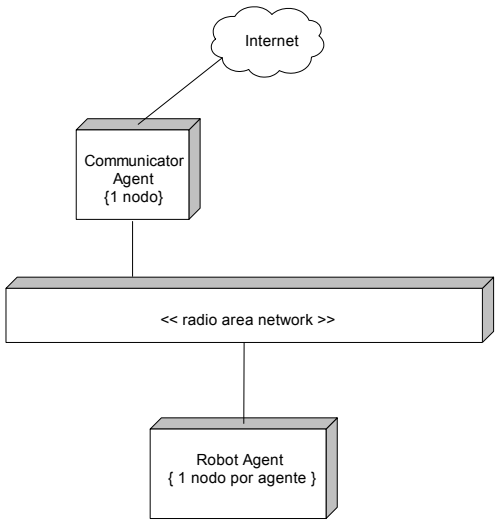

Figura 7 Diagrama de implantación

Diseño de bajo nivel

Se refiere a definir la estructura interna y la conducta de cada agente. El objetivo en este punto es el diseño detallado del agente, empleo de sus componentes de diseño, detalles de sus tareas o funcionalidad, de sus datos, comunicación intra-agente, estudio del cumplimiento de sus restricciones de tiempo real.

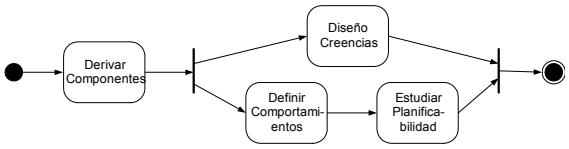

Figura 8 Pasos en el diseño de bajo nivel

Los pasos a seguir en este caso son (ver figura 8):

- *Derivar componentes de los agentes*. En esta fase se determinan los interiores de un agente ARTIS a través de la especificación de sus componentes. Consiste en identificar a partir de las tareas asociadas para cada clase de agente ARTIS sus componentes.
- *Diseño de la estructura de creencias*. Teniendo como entrada el refinamiento realizado sobre el modelo de dominio y la identificación de agentes realizada, se puede pasar a diseñar las creencias de un agente concreto en función de cuales son sus responsabilidades concretas para con la organización y sus propias tareas. Para especificar las creencias de un agente ARTIS concreto se emplea un lenguaje basado en *frames* para la representación de la información tanto estática como temporal.

- *Definir comportamientos*. Los comportamientos representan la parte funcional del agente ARTIS, integrando todo el conjunto de acciones posibles del agente en una situación concreta. Un comportamiento está estructurado como una jerarquía de entidades de conocimiento de resolución y que en un momento dado o situación, un agente ARTIS sólo tiene activo un único comportamiento. Para su desarrollo se dispone de un conjunto de heurísticas para determinar dicha estructura jerárquica de entidades junto con un lenguaje de especificación de entidades de un agente ARTIS [Carrascosa97]. Por medio de dicho lenguaje se pueden expresar todos los componentes de un comportamiento, permitiendo definir detalladamente el conocimiento de resolución de un agente ARTIS.
- *Planificabilidad*. Una vez especificadas las entidades que componen el conjunto de comportamientos del agente robot se pasaría a estudiar su planificabilidad. Con la información temporal disponible, para cada agente puede derivarse qué debe poder ejecutar de manera obligatoria y se puede realizar un análisis de la planificabilidad off-line [Garcia97] del mismo, el cual determine la viabilidad del diseño propuesto. Si alguno de los agentes del sistema no fuese planificable, es porque no se puede asegurar la ejecución de sus partes críticas. En ese caso, se debe replantear su desarrollo.

Implementación

Para el desarrollo de esta actividad se debe implementar cada agente del sistema sobre una plataforma software que de soporte al componente social de dichos agentes. La plataforma SIMBA nos ofrece ciertos servicios a emplear en la implementación del sistema ya que es el marco social sobre el que construir los diferentes agentes ARTIS que constituyen el sistema multiagente SIMBA.

Durante el proceso de implementación de cada agente ARTIS se emplea la herramienta de desarrollo InSiDE (Integrated Simulation and Development Environment) [Julian00] [Soler00]. Dicha herramienta constituye un entorno de desarrollo visual para agentes basados en la arquitectura de agente ARTIS, incorporando los formularios necesarios para desarrollar, de forma gráfica, todos los componentes de un agente de este tipo. Como herramienta de construcción de agentes ARTIS, InSiDE permite especificar las creencias, conocimiento de resolución, conducta social, estructurando todo el conocimiento de forma jerárquica, dando forma al agente en desarrollo.

4. Conclusiones y Trabajo Futuro

En este artículo se ha realizado un análisis de distintas aproximaciones al problema de la construcción de sistemas multiagente. El análisis se ha realizado a diferentes niveles y entre otras consideraciones podemos indicar que aunque en la actualidad existen numerosos trabajos relacionados con dicha problemática, todavía son necesarios considerables esfuerzos para disponer de técnicas y herramientas que nos permitan un desarrollo apropiado. En concreto sería necesario la existencia de herramientas de desarrollo que permitiesen obtener código ejecutable a partir de las especificaciones realizadas. En esa línea se puede destacar la reciente propuesta de la metodología INGENIAS [Gómez02] donde se aborda este aspecto.

Por otra parte, en el artículo también se ha presentado una ampliación de la metodología MESSAGE, a la cual se le ha denominado RT-MESSAGE, para su posible aplicación a entornos de tiempo real. Como posibles líneas de trabajo futuras se plantea el avanzar en la mejora en la obtención de código automático por medio de la herramienta de desarrollo o en la confección de patrones de diseño para agentes basados en la arquitectura de agente ARTIS, fundamentalmente resultarían adecuados patrones de comportamientos.

Agradecimientos

Este artículo ha sido parcialmente financiado por la Comisión Interministerial de Ciencia y Tecnología (CICYT), proyecto nº DPI2002-04434-C04-02 y por la Generalitat Valenciana, proyecto nº CTIDIB/2002/61.

Referencias

[Bergenti01] Bergenti, F. and Poggi, A. (2001). Exploiting UML in the design of multi-agent systems. *LNCS*, 1972:106–113.

[Botti99] Botti, V., Carrascosa, C., Julian, V., and Soler, J. (1999). Modelling agents in hard real-time environments. In *MAAMAW'99*, volume 1647 of LNAI, pages 63–76. Springer-Verlag.

[Brazier97] Brazier, F., Dunin Keplicz, B., Jennings, N., and Treur, J. (1997). Desire: Modelling multi-agent systems in a compositional formal framework. International.

Journal of Cooperative Information Systems, 6(1):67–94.

[Burmeister96] Burmeister, B. (1996). Models and methodologies for agent-oriented analysis and design. *Technical Report* D-96-06, DFKI.

[Carrascosa97] Carrascosa, C., Julián, V., García-Fornés, A., and Espinosa, A. (1997). Un lenguaje para el desarrollo y prototipado rápido de sistemas de tiempo real inteligentes. In *Actas de la CAEPIA'97*, pages 685–694, Malaga, Spain.

[Castro02] Castro, J., Kolp, M., and Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: The Tropos project. Information Systems. Elsevier.

[Douglass99] Douglass, B. P. (1999). Doing Hard Time: Developing Real-Time Systemas with UML. Adisson Wesley, United States of America.

[EURESCOM00] EURESCOM (2000). MESSAGE: Methodology for engineering systems of software agents. Initial methodology. Technical Report P907-D1, EURESCOM.

[EURESCOM01] EURESCOM (2001b). MESSAGE: Methodology for engineering systems of software agents (Final). Technical Report P907-TI1, EURESCOM.

[Garcia97] Garcia-Fornes, A., Terrasa, A., Botti, V., and Crespo, A. (1997). Analyzing the schedulability of hard real-time artificial intelligence systems. *Engineering Applications of Artificial Intelligence*, pages 369–377.

[Georgeff91] Georgeff, M. and A., R. (1991). Modeling rational agents within a BDI-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, San Mateo, CA. Morgan Kaufmann.

[Georgeff95] Georgeff, M. and A., R. (1995). BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agents Systems (ICMAS-95)*, San Francisco, San Mateo, USA.

[Gomez02] Gomez, J. (2002). Modelado de Sistemas Multiagente. PhD thesis, Departamento de Sistemas Informáticos y Programación. Universidad Complutense de Madrid.

[Iglesias99] Iglesias, C., Garijo, M., and González, J. (1999). A survey of agent-oriented methodologies. In Müller, J., Singh, M., and Rao, A., editors, Proceedings of ATAL-98, volume 1555, pages 317–330, Heidelberg, Germany. Springer-Verlag.

[Iglesias98] Iglesias Fernández, C. A. (1998). Definición de una metodología para el desarrollo de sistemas multiagente. PhD thesis, Departamento de Ingeniería de Sistemas Telemáticos. Universidad Politécnica de Madrid.

[Jacobson99] Jacobson, I., Booch, G., and Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison Wesley.

[Jennings98] Jennings, N. and Wooldridge, M., editors (1998). *Agent Technology: Foundations, Applications and Markets*. Springer-Verlang.

[Jennings00] Jennings, N. R. and Wooldridge, M. (2000). Agent-oriented software engineering. *Handbook of Agent Technology* (ed. J. Bradshaw). AAAI/MIT Press.

[Jennings01] Jennings, N. R. (2001). Building complex, distributed systems: The case for an agent-based approach. *Comms. of the ACM (Special issue on agents in telecomms)*, 44(4):35–41.

[Julian00] Julian, V., Gonzalez, M., Rebollo, M., Carrascosa, C., and Botti, V. (2000). InSiDE: una herramienta para el desarrollo de agentes ARTIS. In *Proceedings of SEID'2000*, pages 79–87, Ourense, Spain.

[Julian02a] Julian, V. RT-MESSAGE: Desarrollo de Sistemas Multiagente de Tiempo Real. PhD thesis, Universidad Politécnica de Valencia, 2002.

[Julian02b] Julian, V., Carrascosa, C., Rebollo, M., Soler, J., and Botti, V. (2002). Simba: an Approach for Real-Time Multi-Agent Systems. In *Proceedings of V Conferencia Catalana d'Intel.ligència Artificial*, Castelló. Springer-Verlag.

[Julian02c] Julian, V., Botti, V. Developing Real-Time Multi-Agent Systems. En Actas de 4th Iberoamerican Workshop on Multi-Agent Systems (Iberagents'02), Málaga. 2002.

[Kinny96] Kinny, D. and Georgeff, M. (1996). Modelling and design of multi-agent systems. Technical Report 59, Australian Artificial Intelligence Institute, Melbourne, Australia.

[Lind99] Lind, J. (1999). MASSIVE: Software Engineering for Multi-agent Systems. PhD thesis, DFKI. Alemania.

[Odell00a] Odell, J., Parunak, H., and Bauer, B. (2000a). Extending UML for agents. In Proceedings of the Agent-Oriented Information Systems Workshop, pages 3–17.

[Odell00b] Odell, J., Parunak, H., and Bauer, B. (2000b). Representing agent interaction protocols in UML. In Proceedings of the AGENTS'2000, Barcelona, Spain.

[Robinson00] Robinson, D. J. (2000). A Component Based Approach to Agent Specification. School of Engineering. Air Force Institute of Technology. MS Thesis.

[Schreiber00] Schreiber, A., Akkermans, J., Anjewierden, A., and et al. (2000). *Engineering of Knowledge and Management: The COMMONKADS Methodology*. MIT Press.

[Soler00] Soler, J., Julian, V., Carrascosa, C., and Botti, V. (2000). Applying the ARTIS agent architecture to mobile robot control. In *Proceedings of IBERAMIA'2000*. Atibaia, Sao Paulo, Brasil, volume I, pages 359– 368. Springer Verlag.

[Soler02] Soler, J., Julian, V., Rebollo, M., Carrascosa, C., and Botti, V. (2002). Towards a real-time MAS architecture. In *Proceedings of Challenges in Open Agent Systems. AAMAS'02*, Bolonia, Italia.

[Wood00] Wood, M. F. (2000). Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems. Air Force Institute of Technology. MS Thesis.

[Wooldridge98] Wooldridge, M., Jennings, N. R., and Kinny, D. (1998). A Methodology for Agent-Oriented Analysis and Design. O. Etzioni, J. P. Muller, and J. Bradshaw, Seattle, WA.

[Wooldridge00] Wooldridge, M., Jennings, N. R., and Kinny, D. (2000). The GAIA methodology for agent-oriented analysis and design. Journal of Autonomous Agents and Multi-Agent Systems, 3.

[Wooldridge01] Wooldridge, M. and Ciancarini, P. (2001). Agent-oriented software engineering: The state of the art. In Ciancarini, P. and Wooldridge, M., editors, Agent-Oriented Software Engineering, volume 1957. LNAI, Springer-Verlag.

[Yu96] Yu, E. (1996). Modelling Strategic Relationships for Process Reengineering. PhD thesis, Department of Computer Science. University of Toronto, Canada.