ARTÍCULO

Una Metodología para el Modelado de Sistemas de Ingeniería Orientado a Agentes

Aguilar, J., Bessembel, I., Cerrada, M., Hidrobo, F., Narciso, F.

Universidad de Los Andes
Mérida, Venezuela, 5101
{aguilar,ibc,cerradam,hidrobo,fnarciso}@ula.ve

Resumen

En este artículo se presenta una metodología que comprende las fases de conceptualización, análisis, diseño, codificación y pruebas de sistemas de ingeniería basados en agentes, fundamentada en la metodología *MultiAgent Systems for INtegrated Automation* (MASINA), desarrollada para especificar sistemas multiagentes en ambientes de automatización industrial. La metodología propuesta usa el Lenguaje de Modelado Unificado (UML), ampliamente usado para modelar sistemas de software, y la Técnica de Desarrollo de Sistemas de Objetos (TDSO), la cual es una herramienta para la especificación formal de modelos orientados a objetos. Siguiendo los lineamientos metodológicos para la especificación de sistemas de ingeniería, MASINA se inicia con la fase de conceptualización que permite identificar aquellos componentes del sistema que serán considerados agentes y proponer la arquitectura del sistema multiagentes correspondiente. Estos agentes y sus interrelaciones son especificados e implementados en las fases restantes de la metodología propuesta, usando diagramas UML en la fase de análisis y diseño, y plantillas de TDSO en la fase de diseño.

Palabras clave: Sistemas Multiagentes, Metodología para Modelado de Sistemas, Modelado de Sistemas Multiagentes, Sistemas de Ingeniería.

1. Introducción

La tecnología de agentes está siendo ampliamente utilizada en la academia en los últimos años; como consecuencia, la industria se ha interesado en adoptar esta tecnología para desarrollar sus propios productos [4, 6, 7, 19, 32, 33, 35, 37, 42, 51]. En general, las metodologías para el desarrollo de Sistemas Multi-Agentes (SMA) existentes han surgido como extensiones tanto de metodologías orientadas a objetos, como de metodologías de ingeniería del conocimiento, dada la estrecha relación entre éstas. Ahora bien, no existe una metodología dominante en esta área, y las existentes contienen debilidades y desventajas importantes que no permiten su utilización en ambientes de diseño complejo.

Atendiendo a la creciente demanda de aplicaciones basadas en agentes en el área de control y automatización [19, 32, 33, 35, 51], la metodología *MultiAgent Systems for INtegrated Automation* (MASINA) [8] surge como una propuesta metodológica para la especificación e implementación de sistemas basados en agentes en ambientes de automatización industrial, la cual ha sido usada en [3, 4, 7, 9, 10, 14, 15, 18, 55]. En particular, existen tres características fundamentales en ambientes industriales que requieren especial atención desde el punto de vista del modelado orientado a agentes: por un lado, los requerimientos de tiempo real, que demandan la existencia de modelos de coordinación y comunicación que sean precisos y eficientes; por otro lado, la generación de conocimiento, que debe ser incorporado en la dinámica de gestión de los agentes. Finalmente, el problema de heterogeneidad, que debe ser integrado en las formas de articulación comunitaria por los agentes.

Las diferentes contribuciones de MASINA, en relación a las metodologías existentes, son varias. Por un lado, permite plasmar en un modelo de

inteligencia el proceso de inteligencia a nivel individual (mecanismos de aprendizaje, razonamiento, etc.) y colectivo (modelado de la inteligencia colectiva usando ontologías, etc.). Por otro lado, permite el uso de técnicas inteligentes para especificar tareas que debe realizar un agente. Además, permite caracterizar una gran cantidad de aspectos necesarios en los procesos de coordinación entre agentes: planificación emergente, formas de resolución de conflictos, etc., la comunicación directa o indirecta entre agentes, las conversaciones entre los agentes a través de actos de habla (interacciones), entre otras cosas, para lo cual propone modelos de coordinación y comunicación precisos. También es posible representar aspectos como el uso modelos de referencia para especificar agentes, o la especificación del nivel de abstracción al que pertenece un agente (si forma parte de un SMA que tiene múltiples capas), o especificarlo como un SMA, entre otras cosas.

Por otro lado, el Lenguaje de Modelado Unificado (UML, siglas en inglés) es un lenguaje gráfico que permite visualizar, especificar y documentar cada una de las partes que comprende el diseño de software. UML permite modelar de manera conceptual aspectos tales como: objetos, procesos y reglas de negocio, así como también elementos concretos de sotfware como: clases, bases de datos y componentes reusables [40].

La Técnica de Desarrollo de Sistemas de Objetos (TDSO) se utiliza para la especificación de componentes de software, que soporta todos los constructos de la orientación por objetos, con la inclusión de una guía para el desarrollo de las pruebas del sistema modelado [13, 38].

En este artículo se propone una metodología que combina los aspectos considerados en MASINA [8] con las versatilidades ofrecidas por UML, para visualizar gráficamente las actividades/funciones/comunicaciones de los agentes, y por TDSO, para la definición del universo de clases, y la especificación de las clases y los métodos de los componentes de software que implementan los agentes concebidos, dando lugar a una metodología formal para modelar sistemas de ingeniería orientados a agentes. De esta manera, la principal contribución de esta nueva metodología propuesta es incorporar a las bondades de MASINA, elementos de modelado gráfico y de especificación de componentes que permite definir claramente la arquitectura de agentes para un sistema de software, resaltando la diferencia entre componentes y agentes, actividades, servicios y tareas, aspectos que no están claramente diferenciados en las metodologías existentes. De esta manera, la metodología propuesta en este trabajo permite especificar sistemas multiagentes de forma precisa, facilitando el proceso de implantación en ambientes reales complejos.

Esta metodología ha sido utilizada previamente en los proyectos industriales financiados por PDVSA (Industria Petrolera Venezolana): "Desarrollo de la Ingeniería de Diseño de la Arquitectura de Software Sistémica que permita implementar las funcionalidades y tecnologías identificadas en las mesas corporativas de Arquitectura y de Aplicaciones sobre la plataforma Net-DAS 2.0. de PDVSA-Sur", y "Desarrollo del Medio de Gestión de Servicios de la Arquitectura de Aplicaciones sobre Plataforma Net-Das 2.0"; y en el diseño de un sistema de control y supervisión propuesto en [45].

2. Marco Referencial

La mayoría de las metodologías propuestas para el desarrollo de sistemas basados en agentes han surgido como extensiones y/o modificaciones de otras metodologías. En este sentido, se pueden distinguir tres grandes grupos, el primero basado en metodologías orientadas a objetos [17, 21, 22, 29, 34, 48, 50, 52]; el segundo en metodologías provenientes de ingeniería del conocimiento [26, 29, 30, 31, 47]; y el tercero, en el paradigma de agentes [12, 16, 20, 23, 41, 43, 53, 54]. En esta sección describiremos algunos aspectos relevantes de estas metodologías, en [27, 28] se presentan análisis exhaustivos de diferentes metodologias para el desarrollo de sistemas basados en agentes.

Entre las metodologías para especificación de agentes basadas en orientación a objetos se destacan las propuestas de Kinny [34], Burmeister [17] y MASE propuesta por Wood [52]. Kinny define una metodología para SMA que amplía OMT (Object Modelling Technique); por su lado, Burmesiter describe una metodología para diseñar agentes, extendiendo metodologías orientadas a objetos, en la cual se distinguen tres modelos: modelo del agente, modelo de organización y modelo de cooperación. Sin embargo, en su modelo de cooperación, las interacciones son demasiado simples y no permiten considerar aspectos complejos de coordinación. Wood parte de una especificación inicial del sistema y produce un conjunto de documentos de diseño formal en un estilo

basado en gráficos. Sin embargo, ninguna de estas metodologías toma en consideración el uso de modelos de análisis genéricos y todas tratan al agente como un objeto completo, lo que constituye una visión errónea puesto que los agentes representan un nivel de abstracción más elevado que los objetos.

Con respeto a las metodologías surgidas de la ingeniería del conocimiento, existen dos metodologías relevantes que comparten sus bases en la metodología CommonKADS [26, 29, 30, 47], conocidas como CoMoMAS [26] y MAS-CommonKADS [31], siendo esta última la que mejor cubre los elementos de especificación de un SMA. La metodología MAS-CommonKADS es una extensión de la metodología CommonKADS para modelar SMA, agregando aspectos de las metodologías orientadas a objetos como OMT, Object Oriented Software Engineering (OOSE) y Responsibility Driving Design (RDD). Consiste en seis fases, de las cuales se derivan siete modelos que integrados generan la solución basada en agentes al problema planteado. En sí, estos modelos representan a los componentes (agentes) del sistema, sus interrelaciones, tareas, entre otros aspectos. Sin embargo, esto no significa que esté exenta de debilidades, entre las que se pueden mencionar: la dificultad para especificar un agente como un SMA, la falta de manejo de esquemas dinámicos de comunicación, y la ausencia de modelos de aprendizaje y de coordinación.

Las metodologías surgidas de la propia teoría de agentes se fundamentan en una estructuración social, donde existen individuos (agentes), grupos y organizaciones. Entre estas metodologías se pueden señalar a Cassiopeia [20], Gaia [53, 54], HLIM [23], Prometheus [43], SODA [41] y Tropos [16]. El problema común en estas metodologías es que la fase de análisis y/o especificación se basa en el paradigma de agentes, y no se toma en cuenta que un modelo de análisis general puede dar como resultado que un esquema multiagentes no sea el más apropiado, esto es, algunos componentes del sistema pudieran no ser necesariamente concebidos como agentes; además, el principal elemento de diseño es el papel social del agente y no sus componentes.

En [27, 28] se presentan análisis y comparaciones entre metodologías existentes para el desarrollo de sistemas de software orientados a agentes, donde se resalta la necesidad de disponer de metodologías que resulten en la definición de modelos de interacción y cooperación que no sean abstractos y que capturen las relaciones y dependencias entre agentes, así como sus roles dentro del sistema.

El paradigma de agentes está siendo ampliamente usado como enfoque de modelado de sistemas de control y desarrollos industriales, debido a la naturaleza descentralizada de los problemas de dichas áreas [33] y la complejidad de los ambientes de negocio y manufactura [35]. Por tanto, con el fin de integrar las múltiples perspectivas de los ambientes industriales y sus intereses (control, supervisión, mantenimiento, planificación, informática), y así lograr el alcance de un objetivo global, diversos trabajos han sido orientados a la proposición de arquitecturas basadas en SMA [37, 42, 51]. Estos trabajos prestan más atención a los aspectos de comunicación con vistas generales a la fase de conceptualización.

Así pues, en vista de la potencialidad del uso de SMA en ambientes industriales, surge MASINA [8] a partir de los trabajos realizados en [4, 5, 6, 15, 32] como una metodología para la especificación de SMA en dichos ambientes. MASINA es una extensión del modelo orientado a objetos MAS-CommonKADS, y se basa en el mismo ciclo de desarrollo, con modificaciones que permiten incorporar comportamientos inteligentes (aprendizaje, razonamiento, etc.), especificar los aspectos de comunicación, coordinación, integración, y un agente como un SMA. Particularmente, la posibilidad de especificar un agente como un SMA, análogamente al concepto de holarquía en sistemas holónicos [25], es una posibilidad que permite la metodología en los casos que sea requerido por la complejidad del agente a diseñar, así como también, usar modelos de referencia en la especificación de un agente.

A continuación se presenta la metodología MASINA [8] y la técnica TDSO. Se omite la descripción de UML por ser una herramienta ampliamente conocida [1, 24, 40, 44, 46, 49].

2.1. MASINA

La metodología MASINA [8] es una extensión de MAS-CommonKADS, la cual consta de las fases de conceptualización, análisis, diseño, codificación y pruebas, integración, y operación y mantenimiento.

En la fase de *conceptualización* de MASINA, a diferencia de lo propuesto en MAS-

CommonKADS, no sólo se definen los servicios requeridos del sistema y quiénes lo requieren, sino que se hace una primera identificación de aquellos componentes del sistema que pueden ser considerados agentes y se propone una arquitectura preliminar del SMA. En MAS-CommonKADS, esta arquitectura es concebida en la fase de diseño. MASINA sigue haciendo uso de los diagramas de casos de uso de UML en esta fase de conceptualización.

En la fase de *análisis* de MASINA se reducen a cinco los seis modelos propuestos en MAS-CommonKADS, los cuales se consideran suficientes para describir las características básicas de los SMA (ver figura 1). Particularmente, MASINA hace uso de los modelos de agente, tareas, comunicación, coordinación y sustituye el modelo de experiencia de MAS-CommonKADS por el modelo de inteligencia.

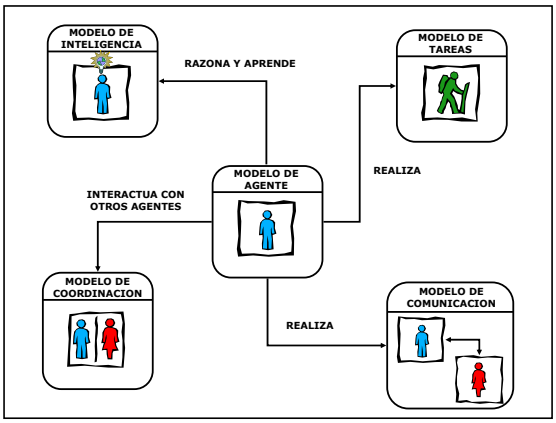

Figura 1: Modelos de MASINA para análisis

En el modelo de agente, MASINA especifica, al igual que MAS-CommonKADS, las características de un agente tales como habilidades, y servicios, entre otras. Además, a este modelo se le agregan dos atributos: *componentes del agente* que permite indicar si el mismo es un SMA (permite representar niveles de abstracción o jerarquías en el diseño del SMA), y *marco de referencia* para indicar si la arquitectura del agente bajo diseño esta basada en un modelo de referencia dado.

En el modelo de tareas, MASINA agrega nuevos atributos, uno para especificar las tareas que requieren el uso de técnicas inteligentes, y otro para describir el macro-procedimiento (sub-tareas) que se debe seguir para la ejecución de dicha tarea. Así, los agentes pueden usar técnicas inteligentes (Redes Neuronales Artificiales, Algoritmos Genéticos, etc.) en función del tipo de tareas que realizan (sean o no ellos inteligentes).

En el modelo de experiencia de MAS-Common KADS sólo se especifica lo concerniente a la experiencia que se va acumulando en un agente, pero no se consideran otros elementos que permiten al agente tener un comportamiento inteligente. MASINA, en su modelo de inteligencia, describe todos los aspectos necesarios para incorporar la noción de inteligencia a un agente.

El modelo de inteligencia propuesto es presentado en la figura 2, con sus atributos, las relaciones entre ellos, y su interrelación con los demás modelos de MASINA. Se propone un esquema que integra conceptos como experiencia, representación de conocimiento (conocimiento de dominio, conocimiento estratégico, conocimiento de tareas), mecanismo de aprendizaje, y mecanismo de razonamiento. Este modelo se activa a través del modelo de tareas, por tareas especificas que conlleven a procesos de razonamiento, aprendizaje, etc.

En MASINA, el modelo de coordinación permite especificar las conversaciones que se dan entre los agentes. A diferencia de MAS-Common KADS, el modelo de coordinación de MASINA se centra en la definición de las conversaciones que permiten una comunicación coordinada entre los agentes, y no en los actos de habla específicamente involucrados, los cuales pasan a ser detallados en el modelo de comunicación de MASINA. Las interacciones (actos de habla) presentes en una conversación dada entre agentes, se representan, a nivel gráfico, a través de un diagrama de secuencia de UML (ver figura 3). Estos actos de habla son detallados en el modelo de comunicación.

El modelo de coordinación de MASINA permite una descripción mas detallada de cada conversación, tal y como se muestra en la figura 4. En este modelo se especifica el esquema de coordinación, la planificación, el mecanismo de comunicación directa o de comunicación indirecta, el metalenguaje y la ontología.

MAS-Common KADS no describe el modelo de comunicación detalladamente, sino que lo supone derivado del modelo de coordinación. En MASINA, el modelo de comunicación considera las interacciones de una manera mas amplia y propone un modelo de comunicación que describe los actos de habla involucrados en las conversaciones entre los agentes del SMA, detalladas en el modelo de coordinación.

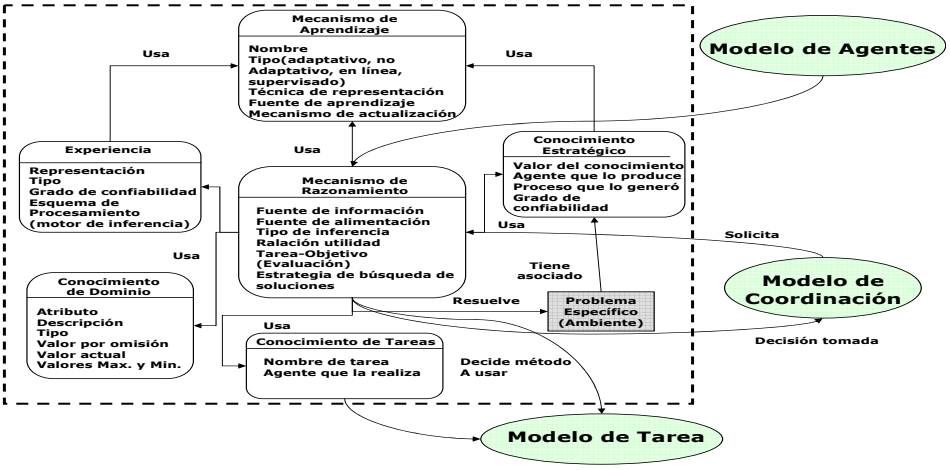

Figura 2: Modelo de inteligencia de MASINA

En la fase de *diseño* de MASINA se obtienen los siguientes modelos como paso previo a la implementación del SMA:

- Diseño del SMA: Se toman en consideración los agentes resultantes de los modelos generados en la fase de análisis para generar la arquitectura final del SMA. Se plasmarán los niveles de abstracción, es decir, la visión holística del SMA.
- Diseño de red: Se describen los aspectos relevantes a la plataforma del SMA como las bases de conocimiento, la arquitectura de red, etc.
- Diseño de la plataforma: Se determina la plataforma de desarrollo del SMA, es decir, se escogen las tecnologías (hardware y software) para implantar la plataforma.

En la fase de *codificación y pruebas* de MASINA se codifica y prueba cada agente utilizando las herramientas escogidas para tal fin. Las dos tendencias principales son el uso de lenguajes de propósito general o de lenguajes orientados a agentes. Cada agente se prueba para asegurar que no tenga fallas, es decir que funcionen de acuerdo con las especificaciones definidas para el SMA en las fases de conceptualización y análisis.

En la fase de *integración* se realiza el acoplamiento entre los agentes del SMA y de éste con la plataforma real donde funcionará.

La fase de *operación y mantenimiento* consiste en el funcionamiento del sistema propiamente dicho, y en la cual se realizan tareas de actualización (mantenimiento) de los componentes del sistema para adaptarlos a la evolución del entorno o a nuevos requisitos.

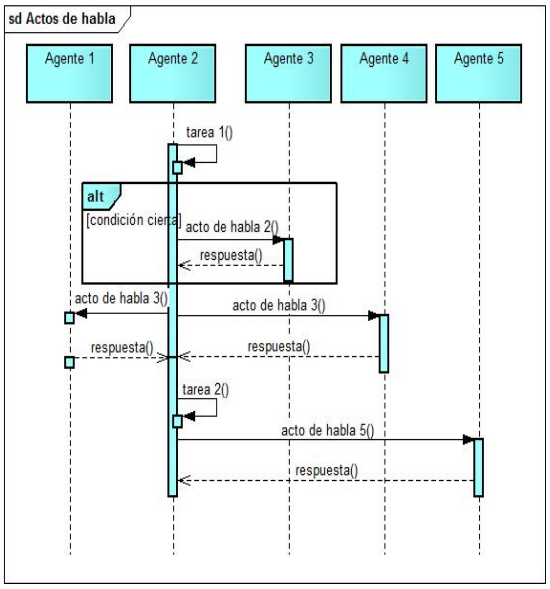

Figura 3: Actos de habla en MASINA

La metodología MASINA ha sido usada para especificar modelos de referencia basados en SMA [3, 4, 7, 9, 10, 11, 18, 14, 19, 55] que permiten implementar tareas propias de ambientes distribuidos. En particular, el desarrollo de modelos de referencia basado en sistemas multi-agentes que permitan implementar tareas de automatización y control de procesos, ha sido de especial interés.

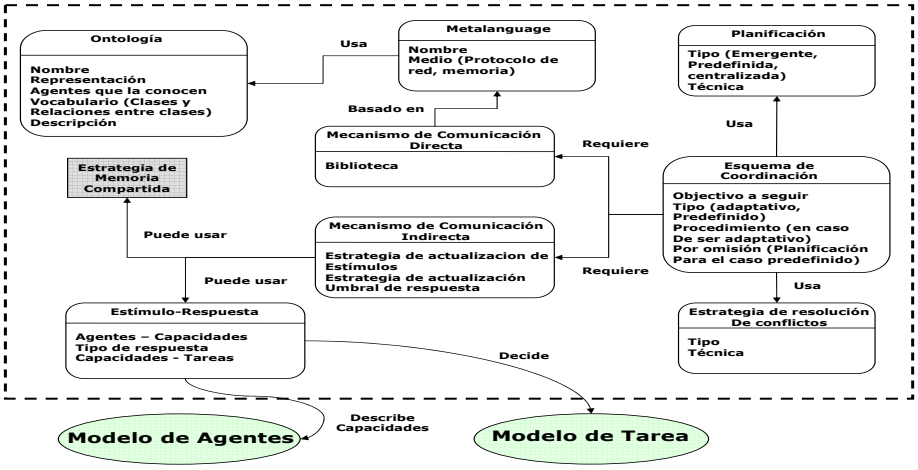

Figura 4: Modelo de coordinación de MASINA

2.2. Técnica de Desarrollo de Objetos

Existen diversas formas tradicionales de expresar la resolución programada de problemas anteriores al uso de la orientación por objetos. Entre ellas se tienen el refinamiento paso a paso, el diseño modular y estructurado, los algoritmos estructurados, el método deductivo, entre otros, que pueden ser utilizados independientemente unos de otros, aunque ellos normalmente se usan en conjunto, para analizar, diseñar y documentar sistemas programados.

Con el advenimiento de la orientación por objetos aparece la metodología OMT (Object- oriented modeling technique) [39], que describe todo el proceso de modelado de clases de objetos en el modelo de objetos, y que además incluye el soporte de las relaciones dinámicas y funcionales entre las clases a través de los modelos dinámico y funcional. La Técnica de Desarrollo de Sistemas de Objetos (TDSO) está basada en el método deductivo y en la metodología OMT [13]. Además, OMT fue utilizada como base para el desarrollo de UML [40]. Del primero contiene todas las fases, incluyendo además las de especificación formal. De la segunda se toman algunos de los diagramas que fueron transformados y adaptados para TDSO, que soportan todos los constructos de la orientación por objetos. La extensión de tales métodos se hace con la inclusión de una guía para el desarrollo de las pruebas de tales sistemas, utilizando el paradigma de la programación orientada por objetos.

TDSO permite:

- Definir el universo de clases y tipos de datos abstractos (TDA).
- Definir formalmente cada clase en términos de sus atributos, la especificación sintáctica de sus métodos y/o operaciones, la especificación semántica que presenta el escenario de cómo se deben utilizar los métodos u operaciones así como su comportamiento, y las declaraciones de las instancias de la clase, TDAs, atributos y/o variables que serán utilizadas en el escenario de pruebas de la especificación semántica.
- Especificar formalmente cada método u operación de una clase.

Una de las principales ventajas de TDSO es que permite documentar las clases y TDAs, los atributos y métodos de cada clase, las declaraciones de las instancias de una clase y/o TDAs, las variables y los casos de prueba de cada método, constituyendo una herramienta apropiada para el diseño de software.

3. Metodología para el Modelado de Sistemas Orientado a Agentes

MASINA ha sido usada para el desarrollo de modelos de referencia basados en SMA en ambientes de automatización y control de procesos, lo cual ha permitido su depuración y validación, llegando a desarrollarse una metodología general para

ser usada en el modelado de sistemas de ingeniería orientado a agentes. Como producto de esta depuración y validación, se han incorporado nuevas herramientas de modelado UML versión 2.0 [44], en las fases de conceptualización, análisis y diseño, se han desarrollado plantillas bien detalladas para especificar cada modelo en la fase de análisis, se ha incorporado el uso de las plantillas de TDSO en la fase de diseño y se han desarrollado plantillas para el diseño de los casos de pruebas.

En las siguientes secciones se describen detalladamente las fases de conceptualización, análisis, diseño y codificación y pruebas de la metodología para el modelado de sistemas orientado a agentes propuesta en este trabajo, usando el formato presentado en [36], el cual permite para cada fase:

- Definir el(los) objetivo(s).
- Definir el producto principal.
- Describir el flujo de trabajo mediante un diagrama de actividades.
- Describir los pasos, actividades, técnicas y notaciones y productos.

3.1. Fase 1: Conceptualización

La fase de conceptualización consiste en la extracción y adquisición del conocimiento para obtener una primera descripción del SMA. Se identifican los elementos que componen el SMA, los procesos que realizan estos componentes del SMA y las relaciones que se establecen para cumplir los objetivos globales. La tabla 1 muestra la plantilla usada para describir los casos de uso.

Tabla 1: Plantilla de descripción de casos de uso

Caso de uso	Nombre del caso de uso
Descripción	Descripción detallada del caso de uso
Pre-condición	Condición necesaria para que exista el caso de uso
Actores	Componentes del sistema que participan en el caso de uso
Condición de fracaso	Elementos/eventos que impiden la culminacion exitosa del caso de uso
Condición de éxito	Elementos/eventos que indican el cumplimiento exitoso del caso de uso

3.1.1. Objetivo

Identificar cada componente del sistema, su conformación, las funcionalidades que provee, las actividades que realiza y las interacciones que se producen entre ellos.

3.1.2. Producto principal

El producto de esta fase es un documento de conceptualización, el cual contiene el análisis del problema, la descripción de los componentes del sistema, la especificación de los servicios y de las actividades para prestar los servicios ofrecidos por cada componente del sistema considerado como un agente y la descripción general de las relaciones entre los componentes del sistema.

3.1.3. Flujo de trabajo

En la figura 5 se presenta el diagrama de actividades que muestra los pasos que se desarrollan en esta fase. La tabla 2 contiene la descripción detallada de esas actividades, las técnicas a utilizar y los productos obtenidos en cada paso.

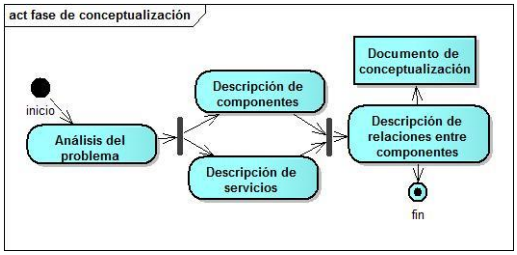

Figura 5: Flujo de trabajo en Conceptualización

3.2. Fase 2: Análisis

En esta fase se lleva a cabo la especificación detallada de todos los elementos propuestos en la fase de conceptualización, así como la especificación de las comunicaciones y coordinación entre el SMA. Esta fase se conoce, en muchas metodologías, como análisis de requisitos, ya que su objetivo o producto es la lista de requisitos formales para la construcción del SMA, y permite construir lo que se denomina el modelo de especificación.

Tabla 2: Especificación del flujo de trabajo de la fase de Conceptualización

Pasos	Actividades	Técnicas y Notaciones	Productos
Análisis del problema	.- Definir el problema .- Caracterizar el problema (descripción detallada)	.- Revision bibliográfica .- Levantamiento de información .- Entrevistas .- Reuniones	.- Documento de análisis del problema
Descripción de componentes	.- Describir los componentes del SMA (identificación del componente y su rol en el SMA)	.- Modelado de casos de uso en UML .- Uso de plantillas para la descripción de casos de uso (ver tabla 1)	.- Diagramas de casos de uso .- Descripción de los casos de uso
Descripción de servicios	.- Definir los servicios por cada componente del SMA y las actividades para prestar dichos servicios	.- Modelado de diagramas de actividades en UML	.- Diagramas de actividades
Descripción de relaciones entre componentes	.- Describir las relaciones entre los componentes del SMA, determinada por los servicios que requiere un componente y los servicios que éste presta	.- Modelado de diagramas de secuencia en UML	.- Diagramas de secuencia

Tabla 3: Especificación del flujo de trabajo de la fase de Análisis

Pasos	Actividades	Técnicas y Notaciones	Productos
Modelado de agentes	.- Especificar las características generales de cada uno de los agentes del SMA	.- Revisión del diagrama de actividades para cada agente .- Uso de plantillas para la especificación del agente (ver tabla 4)	.- Tabla de especificación de agentes
Modelado de tareas	.- Definir las tareas necesarias para el cumplimiento del servicio ofrecido .- Especificar las tareas definidas	.- Revisión de los servicios definidos para el agente .- Uso de plantillas para la especificación de tareas del agente (ver tabla 6)	.- Tabla de relaciones Servicios-Tareas. .- Tabla de especificación de tareas
Modelado de intelegencia	.- Identificar y analizar los comportamientos reactivos e inteligentes de cada agente .- Determinar el conocimiento estratégico, del dominio y de tareas .- Definir los mecanismos para la representación y acumulación de experiencias .- Definir los mecanismos de razonamiento .- Definir los mecanismos de aprendizaje y adaptación.	.- Revisión de las características generales del agente .- Revisión de las tareas definidas en el modelo de tareas .- Entrevistas con los expertos .- Uso de técnicas de Ingeniería de Conocimiento	.- Diagrama del modelo de intelegencia (ver figura 2)
Modelado de coordinación	.- Definir las conversaciones necesarias entre agentes .- Identificar los actos de habla .- Especificar las conversaciones definidas .- Especificar los mecanismos de coordinación y de comunicación	.- Revisión de los objetivos y servicios definidos en el modelo de agente .- Revisión de las tareas definidas en el modelo de tareas .- Diagramas de secuencia en UML (ver figura 3) .- Uso de plantillas para la especificación de las conversaciones (ver tabla 7)	.- Modelado de diagramas de secuencia para cada conversación .- Tablas de especificación de conversaciones .- Diagrama de Coordinación (ver figura 4)
Modelado de la comunicación	.- Especificar los actos de habla de las conversaciones	.- Revisión de los actos de habla definidos .- Uso de plantillas para la especificación de los actos de habla (ver tabla 8)	.- Tablas de especificación de actos de habla

Tabla 4: Plantilla del Modelo de Agente

Agente		
Nombre		Nombre único para el agente
Posición		Ubicación del agente dentro del sistema multiagente
Componentes		Agentes que lo componen (si es un SMA)
Marco de referencia		Marco de referencia para el modelado de agentes
Descripción del agente		Lo que hace el agente
Objetivos del Agente		
Nombre		Nombre del objetivo
Descripción		Detalles del objetivo
Parámetro de entrada		Parámetros necesarios para el cumplimiento del objetivo
Parámetro de salida		Parámetros que se esperan obtener una vez cumplido el objetivo
Condición de activación		Condiciones que activan las tareas asociadas al cumplimiento del objetivo
Condición de finalización		Condiciones que indican la terminación de las tareas asociadas al cumplimiento del objetivo
Condición de éxito		Condiciones que indican el cumplimiento del objetivo
Condición de fracaso		Condiciones que indican el no cumplimiento del objetivo
Ontología		Descripción de la ontología
Servicios del Agente		
Nombre		Nombre del servicio ofrecido por el agente
Descripción del Servicio		Detalles del servicio
Tipo de Servicio		Clasificación (Interno, Externo o ambos: servicio dual)
Parámetros de entrada		Parámetros necesarios para el cumplimiento del servicio
Parámetros de salida		Parámetros obtenidos al finalizar el servicio
Propiedades del Servicio		
Nombre	Valor	Descripción
Calidad		Valor de la calidad del servicio
Auditable		Valor del nivel de auditabilidad del servicio
Garantía		Valor del nivel de garantía de recibir la solicitud del servicio
Capacidad		Valor de la capacidad del agente de cumplir con el servicio (capacidad de respuesta)
Confiabilidad		Valor de confiabilidad del servicio
Capacidad del Agente		
Habilidades del agente		Descripción de las habilidades generales del agente
Representación del Conocimiento		Lenguaje de representación del conocimiento
Lenguaje de Comunicación		Lenguaje de comunicación que usa el agente
Restricción del Agente		
Normas		Descripción de las normas del agente para el cumplimiento del servicio
Preferencias		Preferencias del agente en el momento de atender las solicitudes de servicio
Permisos		Accesos a información permitidos para el agente para el cumplimiento de su objetivo

3.2.1. Objetivo de la fase

Especificar los modelos de agente, tareas, inteligencia, coordinación y comunicación del sistema multiagente propuesto.

3.2.2. Producto principal

El producto principal de esta fase es un documento de análisis contentivo de los modelos de agentes, tareas, inteligencia, coordinación y comunicación del SMA.

3.2.3. Flujo de trabajo

En la figura 6 se presenta el diagrama de actividades que muestra los pasos que se desarrollan en esta fase. De la misma manera que se presentó en la fase de conceptualización, la tabla 3 contiene la descripción detallada de las actividades para la fase de análisis, las técnicas a utilizar y los productos obtenidos en cada paso.

La tabla 4 muestra la plantilla que se usa para realizar la descripción del modelo de agente. En esta descripción se incluyen: Objetivos, Servicios, Capacidad y Restricciones del agente.

La tabla 5 muestra la plantilla que se usa para describir la relación entre los servicios que presta el agente y las tareas que debe ejecutar para cumplir con esos servicios. Para cada tarea del

agente, se construye una plantilla que sirve para especificar la tarea (ver tabla 6).

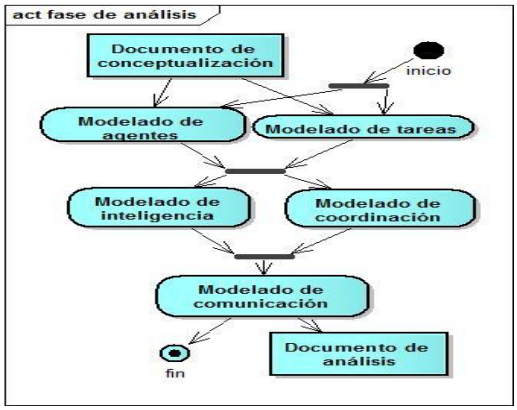

Figura 6: Flujo de trabajo de la fase de Análisis

Tabla 5: Relación Servicios-Tareas del Agente

Servicios	**Tareas**
Nombre Servicio 1	Nombre Tarea $S_1 - T_1$ ⋮ NombreTarea $S_1 - T_n$
Nombre Servicio L	Nombre Tarea $S_L - T_1$ ⋮ NombreTarea $S_L - T_k$

Tabla 6: Plantilla del Modelo de Tareas

Nombre de la Tarea	
Nombre	Nombre de la tarea
Objetivo	Objetivo de la tarea
Descripción	Detalles del objetivo de la tarea
Servicios asociados	Servicios asociado al objetivo
Precondición	Condiciones necesarias para la activación
Sub-tareas	Macro-procedimiento para hacer la tarea
Ingredientes-Nombre de la Tarea	
Nombre Ingrediente 1	Descripción del parámetro 1
⋮	⋮
Nombre Ingrediente n	Descripción del parámetro n

Por otro lado, es necesario especificar las interacciones que se dan en el SMA. Para ello se utilizan los modelos de coordinación y comunicación, en donde se describen, entre otras cosas, las conversaciones y los actos de habla. Las tablas 7 y 8 muestran las plantillas para la especificación de las conversaciones y de los actos de habla, respectivamente.

Tabla 7: Plantilla del Modelo de Conversación

Nombre de la conversación	
Objetivo	Objetivo de la conversación
Agentes participantes	Agentes que participan en la conversación
Iniciador	Agente que inicia la conversación
Actos de habla	Actos de habla que se dan en la conversación
Precondición	Condiciones necesarias para que se inicie la conversación
Condición de terminación	Condiciones que se cumplen para dar por finalizada la conversación
Descripción	Descripción detallada de la conversación

Tabla 8: Plantilla del Modelo de Comunicación

Nombre del Acto de Habla	
Nombre	Nombre del acto de habla
Tipo	Indica la característica de la solicitud (requerimiento de información, de procesamiento, entre otros)
Objetivo	Objetivo de la interacción
Agentes participantes	Agente que participan en el acto de habla (emisor-receptor)
Iniciador	Agente inicia la interacción
Datos intercambiados	Indica cuales son los datos que se intercambian
Precondición	Especifica las condiciones que inician el acto de habla
Condición de terminación	Especifica las condiciones que determinan la finalización de la interacción
Conversaciones	Indica en cuáles conversaciones está presente el acto de habla descrito
Descripción	Detalla el fín del acto de habla específico

3.3. Fase 3: Diseño

En esta fase se obtiene el modelo de diseño bajo el paradigma de SMA, a partir de los modelos del SMA generados en la fase de análisis.

3.3.1. Objetivo de la fase

Elaborar un diseño detallado de los componentes y de la plataforma del SMA.

3.3.2. Producto principal

El producto principal de esta fase consiste en un documento de diseño que describe la estructura del SMA, la especificación detallada de los componentes de dicha estructura y de la plataforma

del SMA.

3.3.3. Flujo de trabajo

En la figura 7 se presenta el diagrama de actividades que muestra los pasos que se desarrollan en esta fase; estos pasos se detallan en la tabla 9.

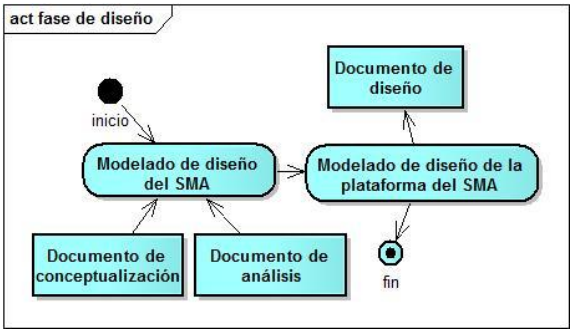

Figura 7: Flujo de trabajo de la fase de Diseño

Además, en la fase de diseño se generan las clases requeridas para la implantación del SMA. Para realizar la espcificación del sistemas se usa TDSO. En este caso se genera el universo de clases del sistema, especificado mediante la plantilla que se muestra en la tabla 10.

Para cada una de las clases descritas en el universo de clases se realiza la especificación formal, para lo cual se usa la plantilla que se muestra en la tabla 11. Así mismo, cada uno de los métodos que componen la clase son especificados mediante la plantilla mostrada en la tabla 12.

3.4. Fase 4: Codificación y Pruebas

Esta fase consiste en la implementación del SMA, para lo cual se escribe el código de los diferentes agentes que lo conforman y en la realización de las pruebas de software, lo que hará posible que el SMA implementado cumpla con los requisitos establecidos en la fase de análisis y corresponda al diseño del SMA producto de la fase anterior.

3.4.1. Objetivos de la fase

Esta fase contempla los siguientes objetivos:

- Implementar el SMA a partir del modelo de diseño generado en la fase de diseño.
- Diseñar y ejecutar el plan de pruebas a objeto de verificar que el comportamiento externo del SMA satisface los requisitos establecidos en la fase de análisis.

3.4.2. Producto principal

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

3.4.3. Flujo de trabajo

En la figura 8 se presenta el diagrama de actividades que muestra los pasos que se desarrollan en esta fase; estos pasos se detallan en la tabla 13.

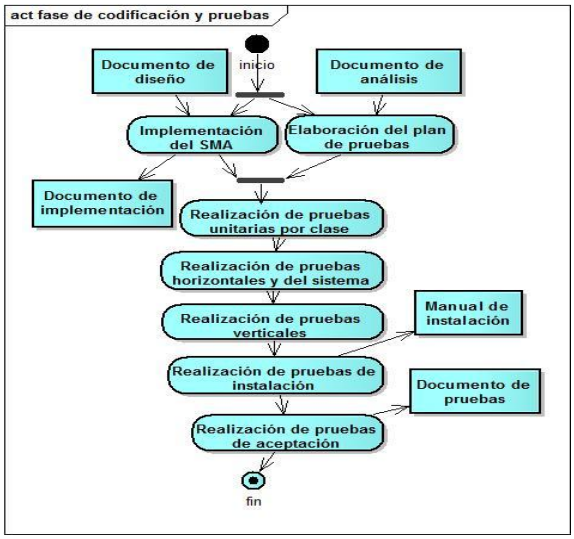

Figura 8: Flujo de trabajo de la fase de Codificación y Pruebas

4. Caso de Estudio

En aplicaciones de control y automatización, la visualización o despliegue de datos referentes a los procesos es de vital importancia para la obtención de información que permita una toma de decisiones adecuada. Ahora bien, el poder usar Agentes de Visualización que se puedan adaptar dinámicamente al perfil de los usuarios, seria de un gran aporte en los procesos de automatización, para darle mayor versatilidad a los mismos (en [2] hay una comparación entre un sistema de automatización convencional y otro basado en agentes).

Tabla 9: Especificación del flujo de trabajo de la fase de Diseño

Pasos	**Actividades**	**Técnicas y Notaciones**	**Productos**
Modelado de diseño del SMA	.- Definir las clases que representan a los agentes .- Definir la interacción entre agentes .- Definir los componentes del SMA .- Distribuir los componentes en una arquitectura .- Especificar detalladamente cada componente	.-Revisión del(los) diagrama(s) caso(s) de uso para cada agente .- Revisión de los modelos para cada agente .- Modelado de clases con el diagramas de clase en UML .- Estilos arquitectónicos .- Modelado de componentes con el diagramas de componentes en UML .- Uso de plantillas de TDSO para la especificación detallada de los agentes (ver tablas 10, 11 y 12)	.- Diagrama de clases .- Diagrama de componentes .- Descripción de la arquitectura del SMA .- Tabla de definición del universo de clases y TDAs de cada agente .- Tabla de definición formal de la clase que representa a cada agente .- Tablas de especificación formal de los métodos de cada clase
Modelado de diseño de la plataforma del SMA	.- Describir los elementos de la plataforma .- Describir cómo se integran los elementos .- Realizar el modelo de despliegue	.- Uso de diagramas de despliegue	.- Diagrama de despliegue

Tabla 10: Definición del universo de clases <nombreSistema>

<fecha>	**Universo de clases y/o TDAs <nombreSistema>** {Comentario sobre el universo de clases y/o TDAs}	<Versión #>
# Clase o TDA	<nombreClase> ([<tipo_parámetros>])[: <claseBase>] <nombreTDA>	Documentación de las clases o TDA

Tabla 11: Definición formal de la clase <nombreClase o nombreTDA>

<fecha>	**# clase o TDA <nombreClase ([tipo_parámetros]) o nombreTDA> [: nombreClaseBase]** {Comentarios sobre la clase}	<Versión #>
# método u operación	**Especificación de atributos**: Comprende los elementos de datos requeridos para conformar la estructura de datos interna de la clase o del TDA. **Especificación sintáctica**: Está constituida por los métodos (de una clase) u operaciones primitivas (de un TDA). **Declaraciones**: Comprende las declaraciones de las instancias de la clase o TDA y de los atributos o variables que serán utilizadas en el escenario de pruebas de la especificación semántica. **Especificación semántica**: Presenta el escenario de cómo se deben utilizar los métodos u operaciones así como su comportamiento.	.- Documentación de las declaraciones. .- Documentación de los atributos de la clase. .- Documentación de los métodos u operaciones definidas en la especificación sintáctica.

Tabla 12: Especificación formal del <nombreMétodo> o de la operación <nombreOperación>

<fecha>	**# clase o TDA, # del método u operación (tipo del método u operación, tipo de acceso)** **nombreMétodo o nombreOperación (Tipo:parámetro,?)[:tipoResultado]** {Comentario sobre el método}	<Versión #>
{pre: <precondiciones>}		{pos: <poscondiciones>}
# Paso	Algoritmo	Documentación de las variables utilizadas
# Caso de prueba	Casos de prueba	Documentación de los casos de prueba

Tabla 13: Especificación del flujo de trabajo de la fase de Codificación y Pruebas

Pasos	**Actividades**	**Técnicas y Notaciones**	**Productos**
Implementación del SMA	.- Seleccionar la plataforma de desarrollo (hardware y software) .- Realizar el modelo de implementación (diagrama de componentes y diagrama de despliegue) .- Seleccionar componentes de software reutilizables, si aplica .- Adaptar componentes de software reutilizables, si aplica .- Desarrollar componentes .- Documentar el código fuente .- Listar los componentes implementados: reutilizados, adaptados y desarrollados	.- Estándares de codificación y estilos de programación .- Modelado de implementación .- Búsqueda de componentes reutilizables .- Búsqueda de componentes adaptables	.- Código fuente documentado de los componentes del SMA (reutilizados, adaptados, desarrollados) .- Documento de implementación
Elaboración del plan de pruebas	.- Definir los objetivos de las pruebas .- Definir los tipos de pruebas a realizar y las técnicas a utilizar .- Construir los formatos de los casos de prueba .- Diseñar el plan de pruebas	.- Estándares de documentación de pruebas	.- Documento del plan de pruebas
Realización de pruebas unitarias por clase (unidad de prueba)	.- Diseñar los casos de prueba para los métodos de cada clase .- Ejecutar los casos de prueba para los métodos de cada clase .- Diseñar los casos de prueba de instanciación: herencia, polimorfismo y encadenamiento dinámico .- Ejecutar los casos de prueba de instanciación .- Registrar los resultados de ejecución de los casos de pruebas .- Elaborar el resumen de incidentes de pruebas	.- Herramientas o ambientes de pruebas .- Estrategias de pruebas unitarias	.- Clases del SMA verificadas y validadas .- Documento de pruebas unitarias
Realización de pruebas horizontales de integración (por niveles de clases y agentes) y pruebas del sistema (funcionales y no funcionales)	.- Diseñar los casos de prueba .- Construir emuladores (salidas) para ejecutar pruebas internas (pruebas horizontales o a un mismo nivel) ante eventos externos .- Documentar el código fuente de cada emulador .- Ejecutar los casos de prueba .- Registrar los resultados de ejecución .- Elaborar el resumen de incidentes de pruebas	.- Herramientas o ambientes de pruebas	.- Documento de pruebas horizontales y pruebas del sistema .- Documento de los emuladores para pruebas horizontales
Realización de pruebas verticales de integración (por niveles del SMA)	.- Diseñar los casos de prueba .- Construir un emulador que genere los datos de los diferentes niveles del SMA .- Documentar el código fuente del emulador .- Ejecutar los casos de prueba .- Registrar los resultados de ejecución .- Elaborar el resumen de incidentes de pruebas	.- Herramientas o ambientes de pruebas	.- Documento de pruebas de integración .- Documento de los emuladores para pruebas verticales
Realización de pruebas de instalación	.- Diseñar los casos de prueba .- Ejecutar los casos de prueba en el ambiente real .- Registrar los resultados .- Elaborar el resumen de incidentes	.- Herramientas o ambientes de pruebas	.- Documento de Pruebas de instalación .- Manual de instalación
Realización de pruebas de aceptación	.- Diseñar los casos de prueba .- Ejecutar los casos de prueba en el ambiente real .- Registrar los resultados .- Elaborar el resumen de incidentes	.- Herramientas o ambientes de pruebas	.- Documento de pruebas de aceptación

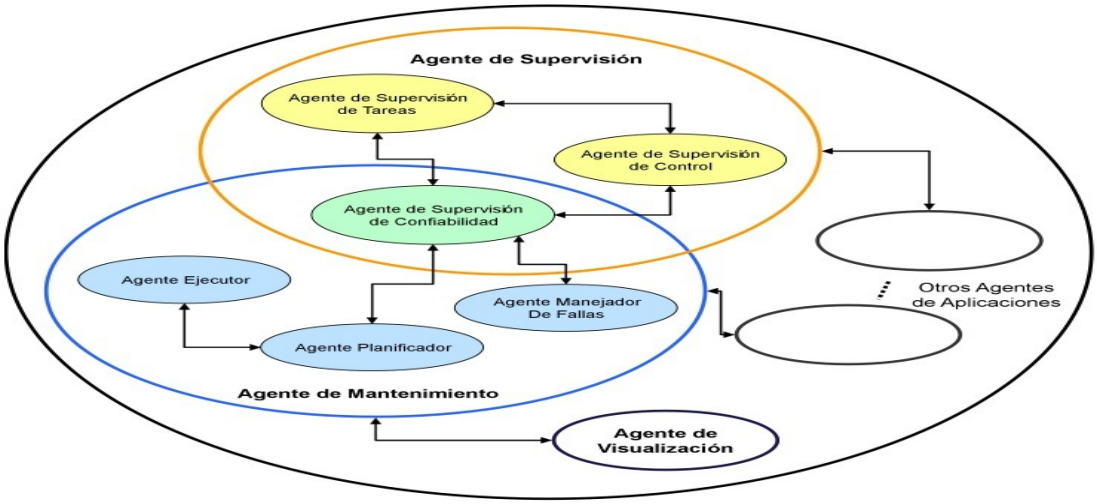

Figura 9: Ejemplo de un SMA para la automatización industrial

Para ilustrar el uso de la metodología propuesta, en sus fases de conceptualización, análisis y diseño, se presenta como ejemplo el desarrollo de un agente de visualización, el cual permite el despliegue de datos en diferentes dispositivos de salida, a solicitud de cualquier otro agente de un SMA del cual este agente forma parte (ver figura 9).

Este agente tiene la capacidad de configurar la interfaz de usuario de un SMA en función de los diferentes roles y perfiles de usuario o de aquellos agentes que soliciten su servicio, entendiéndose por configuración la capacidad del agente de desplegar los datos según algún formato especifíco, por lo que la interfaz de usuario generada por el agente de visualización se dice que es configurable.

En este ejemplo se presentarán los productos generados en cada una de las fases de la metodología para caracterizar a los agentes y sus interacciones, omitiendo la fase de codificación y pruebas, la cual no presenta características muy diferentes a las de cualquier metodología de Ingeniería de Software para realizar esas tareas. Se prefiere mostrar el uso de nuestra metodología en la parte del modelado del sistema de ingeniería orientado a agentes para este caso de estudio, que es la parte más innovadora de ella.

4.1. Fase I. Conceptualización

El Agente de visualización (AV) procesa solicitudes para el despliegue de valores de variables a petición de otros agentes del sistema. La gestión de visualización permite manejar perfiles y configuracines de usuarios en función de sus roles y desplegar los datos.

La figura 10 muestra el diagrama del único caso de uso para el AV. Los actores en este caso de uso son los agentes de aplicaciones, que solicitan una visualización específica, los agentes de gestión de datos que permiten ubicar los datos y los agente especializados (repositorios, tablas, bases de datos) que almacenan los datos a desplegar. Además, los usuarios que interactúan con el AV, para configurar sus formatos de visualización, también son considerados como actores. Este caso de uso es especificado en la tabla 14.

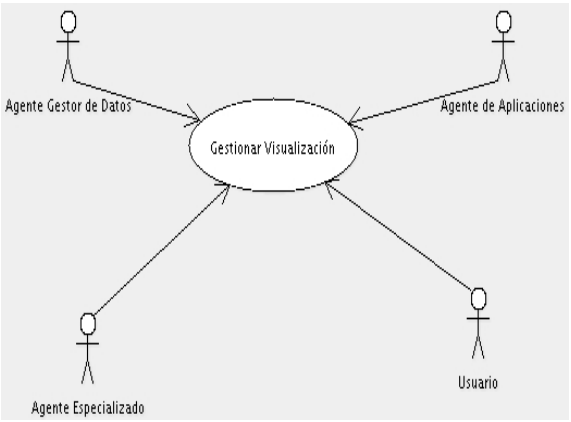

Figura 10: Diagrama de caso de uso para el AV

Para este agente se define un único servicio llamado *Desplegar Datos* y para ello realizan las siguientes actividades: consultar configuración, actualizar configuración, consultar perfil de usuario,

actualizar perfil de usuario y desplegar datos. La figura 11 muestra el diagrama de actividades del AV.

Tabla 14: Descripción del caso de uso para el AV

Caso de uso	Gestionar visualización
Descripción	Procesa solicitudes para el despliegue de valores de variables a petición de otros agentes del sistema.
Pre-condición	Existencia de solicitudes de despliegue de datos, de actualización de configuración y perfiles de usuarios.
Actores	Usuario, Agentes de Aplicación, Agente Especializado y Agente Gestor de datos.
Condición de fracaso	Error de comunicación, en la recepción de la solicitud o en el despliegue.
Condición de éxito	Datos presentados o actualización de perfiles y configuración de usuarios realizadas con éxito.

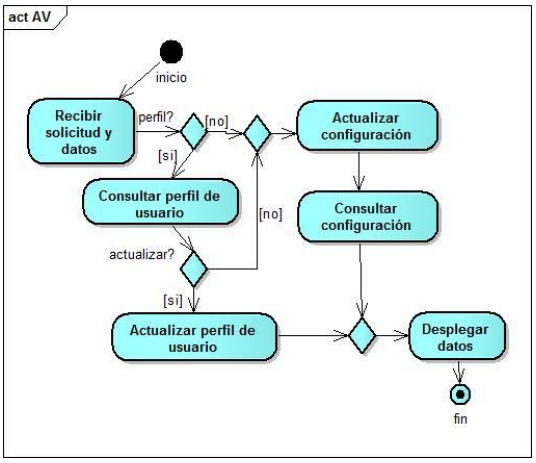

Figura 11: Diagrama de actividades para el AV

4.2. Fase II. Análisis

En esta fase se presenta el modelo de especificación de las comunicaciones y coordinación del SMA.

4.2.1. Modelo de Agente

La tabla 15 muestra un resumen del modelo de agente para el Agente de Visualización. Por razones de espacio, esta tabla no contiene la descripción de cada objetivo y cada uno de los servicios del agente.

4.2.2. Modelo de tareas

En este caso, las tareas que ejecuta el AV, ver tabla 16, se asocian al único servicio definido.

Una vez definidas las tareas, éstas se especifican según la tabla indicada en la descripción de la fase de análisis. La tabla 17 muestra la especificación de la tarea **Realizar Consulta de Datos**.

Cabe destacar que los ingredientes especificados en la tabla 17, no son únicos y dependerán de las funcionalidades que se le especifiquen al agente. Pueden aparecer ingredientes asociados a la configuración de usuario, en caso de cambiar la configuración conocida por el agente.

Tabla 16: Relación Servicio-Tareas para el AV

SERVICIOS-TAREAS	
S1. Desplegar datos	T1. Recibir solicitud y datos T2. Realizar consulta de perfil del usuario T3. Realizar inserción de perfil del usuario T4. Realizar modificación del perfil del usuario T5. Realizar eliminación del perfil del usuario T6. Realizar consulta de datos T7. Realizar consulta de configuración del perfil del usuario T8. Realizar inserción de configuración del perfil del usuario T9. Realizar modificación de configuración del perfil del usuario T10. Realizar eliminación de configuración del perfil del usuario T11. Realizar despliegue de datos

4.2.3. Modelo de Coordinación

En función del objetivo y servicios del AV, se define una conversación que permite todas las interacciones necesarias para el cumplimiento de los mismos. En la tabla 18 se presenta la conversación definida para el AV y en la figura 12 se presenta el diagrama de interacción con los actos de habla de la conversación definida.

4.2.4. Modelo de Comunicación

A fin de ilustrar la especificación de los actos de habla, en la tabla 19 se caracteriza uno de los actos de habla usado en la conversación definida en el modelo de coordinación.

Tabla 15: Modelo de Agente para el AV

AGENTE		
Nombre		Agente de Visualización
Posición		SMA
Componentes		No aplica
Marco de referencia		No aplica
Descripción del agente		Se encarga del despliegue o visualización de datos solicitados por cualquier agente autorizado del sistema. Estos datos deben ser visualizados en un formato preestablecido en la configuración de la interfaz de usuario, de acuerdo a las necesidades de cada usuario. Las actividades principales de este agente son: recibir solicitudes de visualización de datos, actualizar los perfiles y configuración de los usuarios de acuerdo al rol dentro del sistema y administrar las distintas formas de interfaz de usuario que puede soportar el agente (Web, sistemas de ventanas con menú, comandos, etc.)
Objetivos del Agente		
Nombre		Gestionar visualización de datos
Descripción		Recibe la solicitud de visualización de datos identifica el formato de despliegue del o los datos de acuerdo al perfil y configuración del usuario y genera la visualización solicitada
Parámetro de entrada		Solicitud y datos relativos a la solicitud (identificador del solicitante, punto de despliegue, identificador de los datos a visualizar, entre otros.)
Parámetro de salida		Notificación de Visualización del o los datos solicitados
Condición de activación		Recepción de solicitud de visualización de datos
Condición de finalización		Datos visualizados
Condición de éxito		Se visualizaron los datos en el formato especificado
Condición de fracaso		No se visualizaron los datos solicitados o el formato es incorrecto o se produjo un error en la comunicación entre los actores, o los datos no existen
Ontología		Ontología de visualización de datos
Servicios del Agente		
Nombre		Desplegar datos
Descripción del Servicio		Recibe la solicitud de visualización y los datos relativos a la solicitud (identificador de las variable, rangos de valores a desplegar, entre otros). El agente verifica el perfil de usuario (permisos, entre otros), consulta (los) repositorio(s) donde se encuentran los datos, consulta la configuración del usuario y envia los datos al dispositivo o punto de despliegue. Este servicio puede ser ejecutado bajo la modalidad de peticiones o se inicia una vez y luego siempre se estará ejecutando cuando la aplicación así lo requiera.
Tipo de Servicio		Dual
Parámetros de entrada		Solicitud con datos relativos a la solicitud (variables o rangos a desplegar, identificador del usuario, localización del repositorio donde se encuentra el (los) valor(es) de la(s) variable(s), dato(s) o rango(s) a desplegar, punto de despliegue).
Parámetros de salida		Notificación de visualización de los datos
Propiedades del Servicio		
Nombre	Valor	Descripción
Calidad	0-100	Porcentaje de la calidad del servicio en función del tiempo de respuesta
Auditable	1	Capacidad de diagnosticar la calidad del servicio
Confiabilidad	0-100	Certificación de respuesta
Capacidad General		
Habilidades del agente		Gestionar los medios de almacenamiento, Gestionar Dispositivos de despliegues
Representación del Conocimiento		No aplica
Lenguaje de Comunicación		ACL
Restricción		
Normas		El acceso a los medios de almacenamiento estará supeditada a la existencia de conexión entre el agente gestor de datos y los medios de almacenamiento. La gestión de dispositivos de despliegues remotos estará sujeta a la conectividad entre el agente solicitante y el punto de despliegue de los resultados.
Preferencias		Gestión de las solicitudes por orden de llegada
Permisos		Los medios de almacenamiento son accedidos a través del Agente Gestor de Datos

Tabla 17: Modelo de Tareas

REALIZAR CONSULTA DE DATOS	
Nombre	Realizar consulta de datos
Objetivo	Realizar una consulta a un medio de almacenamiento de datos
Descripción	Realiza la consulta de datos al medio de almacenamiento de datos correspondiente, a través del agente Gestor de Datos.
Servicios asociados	Desplegar datos
Precondición	El perfil de usuario del agente solicitante permite la cosulta solicitada.
Sub-tareas	Verificar permisología del usuario Verificar existencia de comunicación Solicitar consulta al Agente Gestor de Datos. Manejar respuesta Manejar errores
INGREDIENTES-NOMBRE DE LA TAREA	
DatoId	Identificador de los datos asociados a la consulta
UsuarioId	Identificador del usuario
Rango	Rango de visualización de los datos, si son numéricos

Tabla 18: Modelo de Conversación

CONVERSACIÓN: **Solicitud de servicio al AV**	
Objetivo	Enviar al Agente de Visualización el listado de los datos, variables operacionales o rangos a desplegar, de manera permanente o puntual a solicitud de un agente autorizado del sistema
Agentes participantes	Agente de Aplicaciones, Agente especializado, Agente Gestor de Datos, Agente de Visualización
Iniciador	Cualquiera de los agentes autorizados del sistema
Actos de habla	Solicitar visualización Consultar datos Visualizar datos Respuesta
Precondición	Existir una solicitud de visualización de datos, variables operacionales o rangos, existir la comunicación entre los agentes involucrados, existir dispositivo de despliegue habilitado
Condición de terminación	Notificación al agente solicitante sobre la respuesta de visualización o actualización de los datos en el dispositivo solicitado
Descripción	Mediante esta conversación, el agente solicitante inicia la conversación que permite al Agente Visualización interactuar con los actores involucrados en en servicio, para lograr el despliegue de los datos, en el dispositivo de salida indicado por el agente solicitante del servicio.

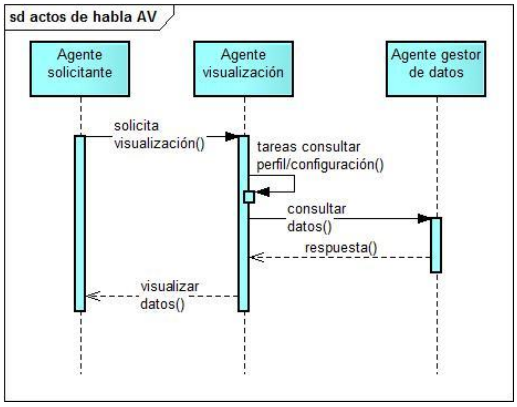

Figura 12: Diagrama de interacción del AV

Tabla 19: Modelo de Comunicación

ACTO DE HABLA: **Visualizar Datos**	
Nombre	Resultados
Tipo	Requerimiento de información
Objetivo	Enviar los datos solicitados en los puntos de despliegue indicados
Agentes participantes	Agente especializado, Agente de aplicación y Agente de visualización
Iniciador	Agente de aplicación
Datos intercambiados	Datos de entrada: Datos a ser visualizador en el formato especificado Datos de salida: Datos visualizados en el punto de despliegue o error
Precondición	Existencia de comunicación entre los agentes involucrados
Condición de terminación	Mensaje de error o de éxito del servicio
Conversaciones	Solicitud de Servicio al Agente Visualización
Descripción	El agente visualización envía los datos al punto de despliegue indicado en el formato indicado

4.3. Fase III. Diseño

En esta sección se muestran algunas de las tablas obtenidas en la fase de diseño usando TDSO. Estas tablas representan la especificación detallada de dicha fase para la implantación del Agente Visualización.

Las tablas 20 y 21 muestran la especificación básica y la especificación formal, respectivamente, para el universo de clases y tipos de datos abstractos de la clase AgenteVisualizacion.

Para cada uno de estos métodos se genera una especificación formal. Para ilustrar dicha especificación, la tabla 22 muestra la descripción detallada del método *desplegarDatosSuministrados*.

Tabla 20: Universo de clases del AV

AgenteVisualizacion
ArchivoHash<Perfil>:perfilAgente ArchivoHash<Configuracion>:confDespliegue Perfil:perfilActual Configuracion:confActual IdAgente: idAgente IdConf:idConfig
+contruirAgenteVisualizacion(): AgenteVisualizacion +desplegarDatosSuministrados(): Lógico +desplegardatosConsultados(): Lógico +insertarPerfilAgente(): Lógico +consultarPerfilAgente(): Perfil +modificarPerfilAgente():Lógico +eliminarPerfilAgente(): Lógico +insertarConfiguracionAgente(): Lógico +consultarConfiguracionAgente(): Lista<Configuracion> +modificarConfiguracionAgente(): Lógico +eliminarConfiguracionAgente(): Lógico +destruirAgenteVisualizacion()

5. Conclusiones

En este artículo se presenta una metodología para especificar SMA en ambientes computacionales. La metodología permite el modelado de un sistema de ingeniería orientado a agentes. Para esto, usa el Lenguaje de Modelado Unificado (UML), ampliamente usado para modelar sistemas de software, y la Técnica de Desarrollo de Sistemas de Objetos (TDSO) la cual es una herramienta para la especificación formal de modelos orientados a objetos. La metodología abarca los lineamientos fundamentales para la especificación de sistemas de ingeniería, iniciando con la fase de conceptualización hasta la fase de codificación y pruebas. La fase de conceptualización permite identificar los actores del sistema y sus casos de uso, lo cual sirve de base para proponer la arquitectura del SMA y definir sus servicios. La fase de análisis permite describir todo el modelo multiagente, desde sus objetivos y tareas, capacidades inteligentes, hasta sus interacciones con el resto del sistema. La fase de diseño permite establecer las bases para la futura implantación del SMA como un objeto de software. Para ello se usan las plantillas descriptivas de desarrollos de objetos planteadas en TDSO. En esta fase se llega al diseño de clases y a la definición y especificación formal de dichas clases. Finalmente, en la fase de codificación y prueba se definen claramente los elementos que permiten verificar la codificación hecha a partir de la fase de diseño, teniendo especial importancia las pruebas de integración del sistema.

El caso de estudio presentado muestra las bondades de la metodología para llevar a cabo el diseño de un agente que cumple un servicio específico dentro de un SMA.

Agradecimientos

Este trabajo ha sido financiado por el FONACIT bajo el proyecto No. 2005000170, y por el CDCHT-ULA a través del proyecto No. I-820-05-02-AA, instituciones a quienes los autores expresan sus agradecimientos.

Referencias

[1] D. Tegarden A. Dennis, B. H. Wixom. *Systems analysis and design with UML version 2.0: an object-oriented approach.* John Wiley & Sons, 2005.

[2] J. Aguilar, I. Besembel, O. Terán, F. Narciso, M. Cerrada, F. Hidrobo, A. Ríos, and L. León. Descripción del Marco Referencial: Desarrollo de Ingeniería Conceptual de Software para la definición de una Arquitectura Sistémica que permita implementar las funcionalidades y tecnologías identificadas en las mesas corporativas de Arquitectura y de Aplicaciones sobre la Plataforma Net-DAS 2.0. Informe Técnico, FUNDACITE-ULA, Venezuela, Noviembre 2004.

[3] J. Aguilar, C. Bravo, E. Colina, and F. Rivas. Sistema Multiagentes para Tratamiento de Situaciones Anormales en Procesos Industriales. In *V Congreso Nacional de la Asociación Colombiana de Automática*, pages 24–28, Medellín-Colombia, Julio 2003.

[4] J. Aguilar, C. Bravo, and F. Rivas. Diseño de una Arquitectura de Automatización Industrial basada en Sistemas Multiagentes. *Revista Ciencia e Ingeniería, Facultad de Ingeniería*, 25(2):75–88, Julio 2004.

[5] J. Aguilar, V. Bravo, M. Cerrada, F. Hidrobo, G. Mousalli, and F. Rivas. Especificación detallada de los agentes del SCDIA. Informe Técnico del 3er año, Proyecto Agenda Petróleo, ULA-FONACIT, Venezuela, Diciembre 2003.

Tabla 21: Definición Formal de la clase AgenteVisualizacion

10 de Mayo de 2007		Versión 1.0
	AgenteVisualizacion {Colección de clases y TDAs requerida para implantar el AV}	
	Especificación de atributos:	
1	ArchivoHash<Perfil>:perfilAgente	*Perfil*: TDA que permite almacenar los elementos estructurales del perfil del Agente solicitante. Posee los campos: IdAgente y DescripciónPerfil *perfilAgente*: Archivo hash donde se almacenan los perfiles de los distintos agentes conocidos por AV
2	ArchivoHash<Configuracion>:confDespliegue	*Configuracion*: TDA que permite almacenar los elementos estructurales de la configuración del Agente solicitante. Esta estructura posee los campos: IdPerfil,IdConf, ConfiguracionDespliegue, Salida. Donde ConfiguracionDespliegue es una estructura que describe la información referente al despliegue *confDespliegue*: Archivo hash donde se almacenan las distintas configuraciones relacionadas con un perfil
3	Perfil:perfilActual	*perfilActual*: Variable interna que se usa para cargar un registro del archivo *perfilAgente* en memoria
4	Configuracion:confActual	*confActual*: Variable interna que se usa para cargar un registro del archivo *confDespliegue* en memoria
5	IdAgente: idAgente	*ideAgente*: Identificación del agente
6	IdConf:idConfig	*idConf*: Identificación de la configuración
	Especificación sintáctica:	
1	construirAgenteVisualizacion(ArchSec< $Perfil$ > : perfilInicial, ArchivoSec< $Configuracion$ > : confInicial)	*construirAgenteVisualizacion()*: Crea un agente del tipo AgenteVisualizacion
2	desplegarDatosSuministrados(IdAgente : idA, IdConf : conf, Lista<InformacionVar> : listaValores) : Logico	*desplegarDatosSuministrados()*: Método que permite el despliegue de los datos suministrado como parámetro siguiendo la configuración de despliegue.
3	desplegarDatosConsultados(IdAgente : idA, IdConf : conf, Lista<InformacionVar> : listaVar) : Logico	*desplegarDatosConsultados()*: Método que permite el despliegue de los datos asociados a la configuración de visualización del agente. Este método permite generar gráficos de tendencias, hacer consultas automáticas a los repositorios de datos y hacer despliegues
4	insertarPerfilAgente(IdAgente : idA, Perfil : perfiAgente) : Logico	*insertarPerfilAgente()*: Inserta un nuevo perfil del agente para que sea conocido por el AV.
5	consultarPerfilAgente(IdAgente : idA) : Perfil	*consultarPerfilAgente()*: Devuelve el perfil del agente.
6	modificarPerfilAgente(IdAgente : idA, Perfil : nuevoperfiAgente, Configuracion: nuevaConfiguracion) : Logico	*modificarPerfilAgente()* : Modifica el perfil actual del agente que invoca agente de visualización.
7	eliminarPerfilAgente(IdAgente : idA) : Logico	*eliminarPerfilAgente()*: Elimina el perfil actual del agente que invoca agente de visualización.
8	insertarConfiguracionAgente(IdAgente : idA, Configuracion : nuevaConfiguracion) : Logico	*insertarConfiguracionAgente()* : Inserta una nueva configuración asociada al agente cliente. Un agente puede tener varias configuraciones de despliegue.
9	consultarConfiguracionAgente(IdAgente : idA) :Lista< Configuración >	*consultarConfiguracionAgente()*: Devuelve la configuración que esta activa del agente cliente.
10	modificarConfiguracionAgente(IdAgente : idA, IdConf: idc, Configuracion : nuevaConfiAgente) : Logico	*modificarConfiguracionAgente()*: Modifica la configuración de despliegue de un agente cliente
11	eliminarConfiguracionAgente(IdAgente : idA, IdConf: idc) : Logico	*eliminarConfiguracionAgente()*: Elimina una configuración del agente cliente.
12	destruirAgenteVisualizacion()	*destruirAgenteVisualizacion()*: Elimina un agente del tipo AgenteVisualizacion.
	Declaraciones AgenteVisualizacion **x** Logico **ban**	***x***: Instancia (objeto) de la clase AgenteVisualizacion. ***ban***: cierto si hay éxito, falso si no.
	Especificacación Semántica AgenteVisualizacion() → AgenteVisualizacion x.desplegarDatosSuminstrados() → ban x.desplegarDatosConsultados → ban x.destruirAgenteVisualizacion()	

Tabla 22: Especificación formal del método desplegarDatosSuministrados

28 de junio de 2008		Versión 1.0
	1,2 (Observador, Público)	
	desplegarDatosSuministrados(IdAgente: idA, IdConf: conf, Lista<InformacionVar>: listaValores): Logico	
	{Despliega los valores suministrados como parámetros siguiendo la configuración asociada al agente que solicitante}	
	{**Pre:** Debe existir Archivo hash de perfil y de configuraciones}	{**Pos:** <Plista ≠ nulo ∨ Plista = Nulo>}
1	Si (Existe perfilAgente y confDespliegue) entonces perfilActual=perfilAgente.busca(idA) confActual=confDespliegue(idA, conf) desplegar(perfilActual,confActual,listaValores) Devolver(cierto) sino Devolver(falso) fin_si	*busca()*: Función del archivo hash que busca el parámetro. *desplegar()*: Método del dispositivo de despliegue.
2	crearAgenteVisualizacion **x**	
3	**x**.desplegarDatosSuministrados() → Cierto o Falso	Despliega los datos según un perfil y una configuración.

[6] J. Aguilar, M. Cerrada, F. Hidrobo, G. Mousalli, and F. Rivas. Aplicación de Sistemas Multiagentes en Problemas del Mundo Real. In *XXVII Conferencia Latinoamericana de Informática (CLEI)*, Sep 2001.

[7] J. Aguilar, Ferrer E, N. Perozo, and J. Vizcarrondo. Arquitectura de un Sistema Operativo Web. *Revista Gerencia en Tecnología Informática*, 2(1):16–29, 2003.

[8] J. Aguilar, F. Hidrobo, and M. Cerrada. A Methodology to Specify Multiagent Systems. *Lecture Notes in Artificial Intelligence*, 4496:92–101, 2007.

[9] J. Aguilar, G. Mousalli, V. Bravo, and H. Díaz. Agentes de control y de gestión de servicio para el modelo de referencia SCDIA. In *II Simposio Internacional de Automatización y Nuevas Tecnologías, TECNO2002*, pages 45–50, Mérida, Venezuela, Noviembre 2002.

[10] J. Aguilar, N. Perozo, and J. Vizcarrondo. Definition of a Verification Method for the MASINA Methodology. *International Journal of Information Technology*, 12(3):121–131, 2006.

[11] J. Aguilar and J. Vizcarrondo. Descripción del Subsistema Manejador de Objetos Web. In *XXX Conferencia Latinoamericana de Informática*, pages 440–450, Arequipa, Perú, Octubre 2004.

[12] F. Alonso, S. Frutos, L. Martinez, and C. Montes. SONIA: A Methodology for Natural Agent Development. In *Fifth International Workshop Engineering Societies in the Agents World*, Toulouse, France, October 2004.

[13] I. Besembel. Técnica de Desarrollo de Sistemas de Objetos (TDSO). Guía Técnica, Grupo de Investigación en Ingeniería de Datos y Conocimiento. Universidad de Los Andes, Mérida, Venezuela, 2003.

[14] C. Bravo, J. Aguilar, M. Cerrada, and F. Rivas. Design of an industrial automation architecture based on multi-agents systems. In *Proceedings of 16th IFAC World Congress*, Prague, Czech Republic, July 2004.

[15] V. Bravo, J. Aguilar, F. Rivas, and M. Cerrada. Diseño de un Medio de Gestión de Servicios para Sistemas Multiagentes. In *XXX Conferencia Latinoamericana de Informática*, pages 431–439, Arequipa, Perú, Octubre 2004.

[16] P. Bresciani, A.Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[17] B. Burmeister. Models and methodology for agent-oriented analysis and design. In K. Fisher, editor, *Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems*, Eindhoven, The Netherlands, 1996.

[18] M. Cerrada, J. Cardillo, J. Aguilar, and R. Faneite. Agent-Based Maintenance Management System For the Distributed Fault Tolerance. In *Proceedings of the*

IFAC SAFEPROCESS 2006, pages 997–1002, Beijing-China, Agosto 2006.

[19] M. Cerrada, J. Cardillo, J. Aguilar, and R. Faneite. Agents-based reference design for fault management systems in industrial processes. *Computers in Industry*, 58(4):313–328, 2007.

[20] A. Collinot, P. Carle, and K. Zeghal. Cassiopeia: A Method for Designing Computational Organizations. In *The First International Workshop: Decentralized Multi-Agent Systems DIMAS'95*, pages 124–131, Crakow, Poland, Nov 1995.

[21] S. A. DeLoach. Modeling Organizational Rules in the Multi-agent Systems Engineering Methodology. In *Canadian Conference on AI*, pages 1–15, 2002.

[22] J. DiLeo, T. Jacobs, and S. A. DeLoach. Integrating Ontologies into Multiagent Systems Engineering. In *AOIS@CAiSE*, 2002.

[23] M. Elammari and W. Lalonde. An Agent-Oriented Methodology: High-level and intermediate models. In G. Wagner and E. Yu, editors, *The First International Bi-Conference Workshop on Agent-Oriented Information Systems*, Seattle, USA and Heidelberg, Germany, May, June 1999.

[24] M. Fowler and K. Scott. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley, 2000.

[25] A. Giret and V. Botti. Holons and agents. *Journal of Intelligent Manufacturing*, 15:645–659, 2004.

[26] N. Glaser. The CoMoMAS Methodology and Environment for Multi-Agent System Development. In C. Zhang and D. Lukose, editors, *Multi-Agent Systems - Methodology and Applications*, pages 1–16. Springer, LNAI 1286, 1997.

[27] J. Gómez and J. Pavón. Methodologies for developing multi-agent systems. *Journal of Universal Computational Science*, 10(4):359–374, 2004.

[28] B. Henderson-Sellers and P. Giorgini (eds.). *Agent-Oriented Methodologies*. Idea Group Publishing, 2005.

[29] C. A. Iglesias, M. Garijo, and J. Centeno-González. A Survey of Agent-Oriented Methodologies. In *ATAL '98: Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages*, pages 317–330, London, UK, 1999. Springer-Verlag.

[30] C.A. Iglesias. Definición de Metodología para el Desarrollo de Sistemas Multiagentes. Master's thesis, Universidad Politécnica de Madrid, Departamento de Igeniería de Sistemas Telemáticos, Madrid, España, Enero 1998.

[31] C.A. Iglesias, M. Garijo, J:C González, and J.R. Velazco. Analysis and Design of Multi-Agent Systems using MAS-CommonKADS. In M.P. Singh, A.S. Rao, and M. Wooldridge, editors, *Intelligent Agents IV. Agents Theories, Architectures and Language*, pages 1–16. Springer, LNAI 1365, 1999.

[32] Aguilar J., Cerrada M., Mousalli G., Rivas F., and Hidrobo F. A Multiagent Model for Intelligent Distributed Control Systems. *Lecture Notes in Artificial Intelligence*, 3681:191–197, 2005.

[33] N. Jennings and S. Bussmann. Agent-based Control Systems. *IEEE Control Systems Magazine*, pages 61–73, 2003.

[34] D. Kinny, M. Georgeff, and A. Rao. A Methodology and Modelling Technique for Systems of BDI Agents. In Rudy van Hoe, editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

[35] V. Marik and D. McFarlane. Industrial Adoption of Agent-Based Technologies. *IEEE Intelligent Systms*, pages 27–35, 2005.

[36] J. Montilva. Desarrollo de Aplicaciones Empresariales. El Método Watch. Informe Técnico, Grupo de Investigación en Ingeniería de Datos y Conocimiento. Universidad de Los Andes, Mérida, Venezuela, 2004.

[37] Y-E. Nahm and H. Ishikawa. A hybrid multi-agent system architecture for enterprise integration using computer networks. *Robotics and Computer-Integrated Manufacturing*, 21:217–234, 2005.

[38] F. Narciso and I. Besembel. Técnicas de Desarrollo de Sistemas de Objetos (TDSO). In *XXII Seminario Interuniversitario de Investigación en Ciencias Matemáticas*, pages 440–450, Ponce, Puerto Rico, Febrero 2007.

[39] OMG. Object Management Group. http://www.omg.org.

[40] OMG. Unified Modeling Language Specification, 2003. Version 2.0, June 2003 via `http://www.uml.org`.

[41] A. Omicini. SODA: Societies and Infrastructures in the Analysis and Design of Agent-Based Systems. In P. Ciancarini and M.J. Wooldridge, editors, *Agent-Oriented Software Engineering*, pages 185–194. Springer, LNCS 1957, 2001.

[42] D. Ouelhadj, S. Petrovic, P.I. Cowling, and A. Meisels. Inter-agent cooperation and communication for agent-based robust dynamic scheduling in steel production. *Advanced Engineering Informatics*, 18:161–172, 2004.

[43] L. Padgham and M. Wimikoff. Prometheus: A Methodology for Developing Intelligent Agents. In P. Giunchiglia, J. Odell, and G. Weiss, editors, *Agent-Oriented Software Engineering III*, pages 174–185. Springer, LNCS 2585, 2003.

[44] D. Pilone and N. Pitman. *UML 2.0 in a Nutshell.* O'REALLY, 2005.

[45] A. Ríos, F. Hidrobo, J. Aguilar, and M. Cerrada. Control and Supervisión Sytem Development with Intelligent Agents. *WSEAS Transactions on Systems*, 6(1):141–148, 2007.

[46] M. Russ. *Learning UML 2.0.* Sebastopol O'Reilly Media, 2006.

[47] G. Schreiber, B. Wielinga, R. de Hoog, H. Akkermans, and W. Van de Velde. CommonKADS: A Comprehensive Methodology for KBS Development. *IEEE Expert: Intelligent Systems and Their Applications*, 9(6):28–37, 1994.

[48] A. L. Self and S. A. DeLoach. Designing and Specifying mobility within the multiagent systems engineering methodology. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 50–55, New York, NY, USA, 2003. ACM Press.

[49] S.W. Ambler. *The object primer: the application developer's guide to object orientation and the UML.* Cambridge University Press, New York, 2nd ed. edition, 2001.

[50] S. Vafadar, A. Barfouroush, and M. Reza A. Shirazi. Towards a More Expressive and Refinable Multiagent System Engineering Methodology. In *AOIS*, pages 126–141, 2003.

[51] T. Wagner. Applying Agents for Engineering of Industrial Automation Systems. In M. Schillo et al., editor, *Lecture Notes in Artificial Intelligence. MATES 2003*, volume 2831, pages 62–73. Springer Verlag, Berlin, 2003.

[52] M. F. Wood and S. A. DeLoach. An overview of the multiagent systems engineering methodology. In *First international workshop, AOSE 2000 on Agent-oriented software engineering*, pages 207–221, Secaucus, NJ, USA, 2001. Springer-Verlag New York, Inc.

[53] M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000. Times Cited: 3 Article English Cited References Count: 34 412fd.

[54] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.

[55] W. Zayas, J. Aguilar, M. Cerrada, F. Hidrobo, and F. Rivas. Development of a Code Generation System for Control Agents. *WSEA Transactions on Computers*, 5(10):2406–2411, 2006.