ULA

UNIVERSIDAD
DE LOS ANDES
VENEZUELA

Sistemas MultiAgentes y sus Aplicaciones en Automatización Industrial

Jose AGUILAR
Addison RIOS BOLIVAR
Francisco HIDROBO
Mariela CERRADA

Universidad de Los Andes
Facultad de Ingeniería
Facultad de Ciencias
Mérida - Venezuela

Segunda Edición, 2013

Titulo de la obra:

- **Sistemas MultiAgentes y sus Aplicaciones en Automatización Industrial**

Autores:

- Jose AGUILAR, Dr.
- Addison RIOS BOLIVAR, Dr.
- Francisco HIDROBO, Dr.
- Mariela CERRADA, Dra.

Diseñadoras gráficas:

- Ing. Paola Rivero
- T.S.U. Nadia González

HECHO EL DEPÓSITO DE LEY

- Depósito Legal: lf23720120042635
- ISBN: 978-980-12-5942-8

Autores

Jose Aguilar
Profesor Titular, Departamento de Computación
Facultad de Ingeniería, Universidad de Los Andes
La Hechicera, Mérida-Venezuela.
Doctor en Informática de la Université Paris V "René Descartes", París, Francia, 1995.
Post-Doctorado en Ciencias Computacionales en University of Houston, Texas, Estados Unidos, 2000.
Areas de investigación: Computación Inteligente, Sistemas Distribuidos/Paralelos, Inteligencia Artificial Distribuida, Sistemas Emergentes.
e-mail: aguilar@ula.ve

Addison Ríos Bolívar
Profesor Titular, Departamento de Sistemas de Control
Facultad de Ingeniería, Universidad de Los Andes
La Hechicera, Mérida-Venezuela.
Doctor en Ciencias Aplicadas de la Universidad de Los Andes, 2003.
Doctor en Sistemas Automáticos de la Université Toulouse III "Paul Sabatier", Toulouse, Francia, 2001.
Areas de investigación: Control Automático, Detección y Diagnóstico de Fallas, Control Robusto, Automatización.
e-mail: ilich@ula.ve

Francisco Hidrobo
Profesor Titular, Departamento de Física
Facultad de Ciencias, Universidad de Los Andes
La Hechicera, Mérida-Venezuela.
Doctor en Informática de la Universidad Politécnica de Catalunya, Barcelona, España, 2004.
Areas de investigación: Computación Inteligente, Computación de Alto Rendimiento, Sistemas Distribuidos/Paralelos.
e-mail: hidrobo@ula.ve

Mariela Cerrada
Profesora Titular, Departamento de Sistemas de Control
Facultad de Ingeniería, Universidad de Los Andes
La Hechicera, Mérida-Venezuela.
Doctora en Sistemas Automáticos del Instituto Nacional de Ciencias Aplicadas (INSA), Toulouse, Francia, 2003.
Areas de investigación: Control Automático, Sistemas Inteligentes, Detección de Fallas en Sistemas a Eventos Discretos.
e-mail: cerradam@ula.ve

A NUESTRAS FAMILIAS, EN ESPECIAL
LA ACADÉMICA

Prefacio

Un prefacio, como indica la palabra, es un texto inicial que precede "lo hecho". Por esta razón, no tengo ningún reparo en comenzar hablando del "*big bang*". Según esa teoría, actualmente plebiscitada por la mayoría de los cosmólogos, todo empezó por una impresionante explosión, las partículas que se dispersaron en aquel corto instante han proporcionado el material con el cual todos estamos hechos. El juego de los campos de fuerzas que rigen el universo desde su creación, o tal vez desde antes, ha ido provocando asociaciones entre los trozos de aquella explosión. El polvo sideral, sin esas agregaciones hubiera permanecido simplemente polvo inerte, esperando recaer en su forma inicial. Toda la historia del universo está ritmada por esas agregaciones, sin las cuales ni las estrellas, ni los mundos, ni los elementos, ni la vida, ni la inteligencia habrían podido existir.

De la primera dispersión de energía y materia salieron partículas ínfimas, de las cuales sólo tenemos el recuerdo gracias a la "radiación fósil" (*Cosmic Microwave Background* o CMB), o sea el polvo primordial. Ese polvo fue inerte antes de que se formaran núcleos, con o sin cargas eléctricas: éste fue el primer paso. La formación de átomos, luego de moléculas, luego de grandes moléculas, tipo ADN, finalmente de la célula viva, de los animales y del cerebro inteligente, han sido los escalones por los que ha evolucionado el universo.

Hay que notar que, a cada etapa, las propiedades del nuevo objeto difieren y sobrepasan las de sus componentes. ¿Mediante qué mecanismo han podido haber ocurrido estas transformaciones? La respuesta es al mismo tiempo simple y compleja. Ha ocurrido simplemente que, a cada agregación, en los intersticios de sus componentes se ha escondido "información", es decir ha habido diálogo entre los componentes. El diálogo es una condición necesaria, aunque no suficiente, para la cooperación entre agentes.

El concepto de "*Agente*", como se menciona con mucha razón en este libro, no tiene una definición aceptada por todo el mundo, sin embargo, el tema mismo de esta publicación me indica hacia donde hay que dirigir la mirada: se habla principalmente de Agentes Inteligentes. Es intencionalmente que en la enumeración de los escalones o etapas de la formación del Universo tal como lo conocemos, he citado el cerebro inteligente como etapa final, o al menos actual.

En el desarrollo histórico de los artefactos, es decir de los objetos construidos gracias al "arte" del cerebro humano, encontramos también unas etapas, no sólo en la complejidad de estos objetos, si no en su "autonomía". Mientras el "ingeniero", es decir el hombre ingenioso y creador, se ha limitado a desarrollar herramientas, sus objetos eran absolutamente inertes; sólo la energía del hombre, o del animal, les daba vida. El principal obstáculo para salir de esta inmovilidad era la falta de energía propia. A partir de la famosa invención de la máquina de vapor, este obstáculo fue vencido;

entonces se planteó el problema de la autonomía de los artefactos. El regulador de Watt fue la primera realización que permitió controlar la energía indómita del vapor y proporcionar la modesta autonomía de la máquina, que le permite continuar funcionando en un régimen estacionario, sin requerir introducción de mandos externos, a pesar de las perturbaciones a la que el sistema pudiera estar sometido.

A pesar de la filosofía original que los griegos habían aplicado a la sociedad ateniense, formada de hombres libres y autónomos, la estructura, reconocida por su eficacia en la guerra, era la jerarquía, en la cual la voluntad se transmite en una sola dirección. No es de extrañar que hasta muy reciente no se haya considerado la posibilidad de dar autonomía a los objetos artificiales. Ha sido el antropomorfismo de que han estado circundados los robots desde su creación, el que ha traído a la mente de los científicos el concepto de autonomía de objetos artificiales. En este nuevo paradigma aparecen naturalmente la movilidad, la reactividad, la comunicación, y como corolario, la adaptatividad y la capacidad de evaluación. Una palabra que reúne todos estos conceptos es la "Inteligencia": se trata de dotar los artefactos de Inteligencia.

De forma similar a lo que hemos dicho de las etapas de formación del universo, el dar a nuestros artefactos inteligencia, y llamarlos "agentes", ha abierto una nueva etapa en la ingeniería. Como he dicho antes, la infancia de la nueva etapa empezó con la teoría de los sistemas con retroalimentación o *feedback*. Actualmente, la tarea del estudio y del desarrollo de sistemas consiste en proponer, gracias a la comunicación, metodologías para realizar una agregación inteligente de subsistemas, o agentes, para satisfacer funcionalidades altamente elaboradas.

Uno de los desafíos permanentes del ingeniero, tanto en el diseño como en la conducta de mecanismos, es la lucha permanente contra la incerteza, ya sea causada por falta de conocimiento, o por perturbaciones impredecibles. El estudio de las sociedades humanas nos muestra que la autonomía y la libre decisión de cada uno de los agentes involucrados en una tarea es la manera más eficaz de vencer las incertezas y las perturbaciones a medida que se presentan.

Ya sé que el mundo de los agentes se ha desarrollado enormemente de forma virtual, dentro del mundo artificial, me atrevería a llamarlo paralelo, de la informática de la comunicación, es decir de la "tela de araña", o "*web*" de Internet. Sin embargo, como ya he mencionado, fue la Automática, con el estudio de sistemas en anillo cerrado, "feedback", la que permitió el salto de la técnica a un nuevo escalón. De forma similar, creo yo, el campo de la Automatización de procesos industriales, es el lugar adecuado para que los nuevos conceptos adquieran la consagración que han de obtener gracias a la confrontación con sistemas materiales.

Hace unos años impartí algunas conferencias que dieron lugar a la publicación por parte de la Universidad de Los Andes en Mérida, Venezuela, la ULA, de un librito donde se trataba de proponer técnicas de Inteligencia

Artificial en el contexto del control de procesos; dicho libro parece haber tenido una buena aceptación en centros de investigación de habla hispana. El presente libro tiene además el privilegio de tratar de un tema mas novedoso y actual. Gracias a haber sido invitado a seguir algunas tesis de esta Universidad, en las cuales estos temas tomaban apoyo en la industria petrolera, he podido apreciar que los grupos de investigación, y en particular los autores de este libro, están entre las personas mejor acreditadas para hablar de la teoría, y también de la práctica, de los Agentes Inteligentes.

El entorno de los procesos industriales proporciona al estudio de los sistemas basados en agentes varios matices que no presentan la misma importancia cuando uno se limita al mundo virtual e informático. La integración en el campo de la ingeniería empresarial obliga a considerar arquitecturas, flujos y funciones específicas. Una de las cualidades del presente libro es de haber sabido explotar las experiencias de sus autores para introducir estos conceptos, tanto en sus aspectos teóricos como en los ejemplos de aplicaciones reales a la supervisión.

La realización y programación (hardware y software) de las estructuras informáticas basadas en el concepto de agente no pueden estar basadas en una teoría única, depende de muchos factores, y solamente un buen análisis de los conceptos utilizados puede guiar su diseño. En este libro, este tema viene también respaldado por aplicaciones en el mundo de la empresa. La modelización, en muchos casos, mediante la simulación, puede permitir cortocircuitar la experimentación, sin embargo, en el caso de los sistemas basados en agentes inteligentes, es complementaria, pero en ningún caso suficiente. Esta situación también ha sido examinada aquí, teniendo en cuenta los niveles de abstracción y los modelos de más amplia difusión.

No me queda más que felicitar y apoyar la labor de mis colegas y amigos de la ULA por la iniciativa de compilar en este libro, no solo en el contexto del estado del arte y de los conceptos básicos de los Agentes Inteligentes, si no que, gracias a sus experiencias de tamaño real, han podido ilustrar su utilización práctica, y demostrar el bien fundado de la orientación de la investigación.

Toulouse,
Julio 2012

Joseph Aguilar-Martin
Directeur de Recherche, (Emérito)
Laboratoire dÁnalyse et dÁrchitecture des Systèmes
LAAS- CNRS, Toulouse, Francia
Catedrático de Universidad (Emérito)
Catalunya, España

Exordio

En este libro se explora el vínculo entre dos áreas, una del ámbito computacional, en específico el tema de *Inteligencia Artificial*, y la otra vinculada al área de *Automatización de Procesos*. En el primer caso, los aspectos considerados del área Inteligencia Artificial corresponden a lo que se ha conocido como Inteligencia Artificial Distribuida, y más precisamente como *Sistemas Multiagentes*. Concretamente, los Sistemas Multiagentes conjugan ideas de las áreas de los Sistemas Distribuidos y de la Inteligencia Artificial.

Esa exploración acontece por la necesidad de mejorar los procesos industriales a través de mecanismos automatizados que partan de conceptos computacionales innovadores, como los que se conjugan alrededor del concepto de Sistemas Multiagentes. El libro presenta los resultados de esa exploración: productos de trabajos de investigación y desarrollo de tecnologías que los autores han venido realizando en sus años como docentes e investigadores en la Universidad de los Andes. Durante la realización de dichos trabajos son muchos los colegas, ingenieros y estudiantes que han participado con los autores, a quienes se les agradece todos los aportes dados. Además, han sido varias las fuentes que han financiado los mismos, como el FONACIT y sus programas de apoyo a la investigación, el CDCHTA de la Universidad de los Andes, y PDVSA, la compañía petrolera venezolana. A esas instituciones también nuestro agradecimiento por la confianza brindada.

Así, en este libro revisamos muchos de los estudios, investigaciones, y resultados que hemos realizados en el ámbito de la automatización industrial basada en agentes, así como las bases teóricas que sustentaron esos trabajos. El libro trata de hacer una presentación coherente de los trabajos, dejando algunos otros pendientes para futuros libros. En ese sentido, esta segunda edición del libro, reestructurada, corregida y ampliada, se organiza en cuatro partes: una primera dedicada a la teoría de agentes; la segunda, correspondiente a un tema novedosa en esta edición, está consagrada al estudio de la implantación de Sistemas Multiagentes. La tercera parte está destinada al estudio de la automatización integrada de procesos. Finalmente, la cuarta parte está dedicada a la presentación de aplicaciones basadas en Sistemas Multiagentes para la automatización de procesos industriales.

En específico, en la primera parte presentamos los dos conceptos claves bases de los trabajos presentados en el libro tomados de la Inteligencia Artificial Distribuida: *los agentes* y *los sistemas multiagentes*. Existe mucha literatura sobre esos conceptos, así que el libro hace una presentación resumida, pero completa, de los aspectos más importantes alrededor de esos conceptos, fundamentales para lo presentado en la tercera parte del libro. Después, esta parte del libro presenta una *metodología para especificar agentes* en problemas de automatización desarrollada en la Universidad de los Andes.

Con respecto a la metodología, si bien es cierto que en la literatura existen varias para desarrollar agentes, nuestra metodología conjuga dos aspectos distintos ha aquellas que se encuentran en la literatura: es orientada a la especificación de agentes de automatización industria; y mezcla técnicas de modelado del ámbito de la programación orientada a objetos (TDSO y UML) con otra del área de agentes (MASCommonKAD), para proponer una metodología que permite especificar, en detalle, los aspectos que caracterizan a los agentes y a la comunidad de agentes.

En la segunda parte, partiendo de una evaluación de las plataformas y estándares dedicados a la implantación de agentes inteligentes como mecanismo de solución informático computacional a problemas de ingeniería en general, se analizan las características y especificaciones que permitan desarrollar herramientas de software bajo el paradigma de Sistemas Multiagentes, como soporte a la gestión de procesos. El estudio de los estándares y plataformas se realiza desde la perspectiva del estado del arte, así como de las contribuciones que cada orientación somete a consideración del tema. Luego, sobre la base de los requerimientos de un sistema operacional para el funcionamiento de los Sistemas Multiagentes, se presenta un Medio de Gestión de Servicios, el cual es fundamental para el desempeño operacional de los agentes. Así, se aborda el problema de los *medios computacionales de gestión de servicios* requeridos para dar soporte a comunidades de agentes, presentando una especificación del mismo. Dicha especificación sigue el estándar establecido por la FIPA (*Foundation for Intelligent Physical Agents*). Finalmente, se hace una presentación del desarrollo y contribución de los autores al tema, caracterizando y especificando un entorno de desarrollo integrado para la construcción de Sistemas Multiagentes.

En la tercera parte se hace un estudio de la *automatización industrial*. Para ello, inicialmente se presentan conceptos bases alrededor del tema de automatización. Después, se hace una presentación detallada de las *arquitecturas de automatización industrial*, desde las jerárquicas, pasando por las heterárquicas, culminando con los modelos híbridos. Finalmente, esta parte culmina con una revisión del problema de integración para la automatización de procesos industriales, generando algunas luces y alternativas de solución.

La cuarta parte de este libro, dedicada a la automatización de procesos, a partir de aplicaciones basadas en Sistemas Multiagentes, comienza con la presentación de diferentes arquitecturas de automatización fundamentadas por agentes inteligentes. Enseguida se presentan algunos de los trabajos desarrollados en la Universidad de los Andes alrededor del problema de automatización industrial, basados en la teoría de agentes. Luego, se presentan tres modelos computacionales basado en agentes, responsables de tres tareas vitales en cualquier proceso de automatización industrial: control, supervisión y planificación. Por consiguiente, se presentan tres *Sistemas Multiagentes* para esas tareas particulares de *automatización de procesos industriales*.

Este libro es de interés para personas que quieran conocer más sobre el ámbito de aplicación de los Sistemas Multiagentes, o aquellas interesadas en el estudio de los procesos de gestión industrial, y en particular de la Automatización de Procesos, y como los avances en Inteligencia Artificial pueden ser usados en ellos. Los posibles productos tecnológicos vinculados a los trabajos mostrados en este libro, o que se derivan de ellos, son enormes. Dejamos al lector el reto del desarrollo de los mismos.

Mérida, *Los Autores*
Octubre 2013 *ULA*

Agradecimientos

Durante la realización de este trabajo son muchos los colegas, ingenieros y estudiantes que han participado con los autores, a quienes se les agradece todos los aportes brindados.

Adicionalmente, han sido varias las fuentes que han financiado los trabajos reportados en este libro, como el FONACIT y sus programas de apoyo a la investigación, a través de los proyectos: *2005000170: "Técnicas Emergentes para la Automatización Integrada de Procesos de Producción"*, *PCP-"Automatización Integrada de Procesos de Producción"*, *Agenda Petróleo (97003817)*. El CDCHTA de la Universidad de los Andes, por medio del proyecto No. *I-1237-10-02-AA*: "*Diseño y Construcción de una Plataforma para el Desarrollo e Implantación de Sistemas Multiagentes. Caso de Estudio: Automatización de Procesos Industriales*". La compañía petrolera venezolana PDVSA, a través de varios proyectos de investigación y desarrollo. El Vicerrectorado Administrativo de la Universidad de Los Andes. A esas instituciones, también nuestro agradecimiento por la confianza brindada.

Esta segunda edición, corregida y ampliada, ha sido publicada por el soporte financiero de FONACIT, a través del proyecto No. *2013000437*, dándole continuidad a sus programas de divulgación de la ciencia, tecnología e innovación.

Finalmente, nuestras palabras de profundo agradecimiento a la **Lic. Luisa Díaz**, Secretaria del Postgrado de Computación de la Facultad de Ingeniería, ULA; y a la **Sra. Alis Becerra**, Secretaria de la *Fundación Centro Nacional de Desarrollo e Investigación en Tecnologías Libres*, CENDITEL, quienes con sus labores, el apoyo, aliento y estímulo que nos han brindado, nos han permitido completar esta obra.

ÍNDICE GENERAL

Índice de figuras

ÍNDICE DE TABLAS

Acrónimos

AARIA Autonomous Agents for Rock Island Arsenal
ACL Agent Communication Language
ADC Convertidor Analógico Digital
BDI Beliefs-Desires-Intentions
CDD Control Digital Directo
CIM Manufactura Integrada por Computadora
DAC Convertidor Digital Analógico
DCS Sistema de Control Distribuido
EDISMA Entorno de Desarrollo Integrado para la construcción de SMA
ERP Enterprise Resources Planning
FIPA Foundation for Intelligent Physical Agents
IAD Inteligencia Artificial Distribuida
IDE Entorno de Desarrollo Integrado
IGU Interfaz Gráfica de Usuario
IDE Integrated Development Environment
JADE Java Agent DEvelopment Framework
JVM Java Virtual Machine
KQML Knowledge Query and Manipulation Language
MASINA MultiAgent Systems for INtegrated Automation
MES Manufacturing Execution Systems
MGS Medio de Gestión de Servicios
OMT Object-oriented Modeling Technique
PABADIS Plant Automation Based on Distributed Systems
PID Proporcional-Integral-Derivativo
PLC Controlador Lógico Programable
PROHA Product Resource Order Heterarchical Architecture
RDP Resolución Distribuida de Problemas
RPC Remote Procedure Call
RTU Unidad de Transmisión Remota
SADIA Sistema Automatizado Distribuido Inteligente basado en Agentes
SCADA Supervisory Control and Data Acquisition
SCDIA Sistemas de Control Distribuido Inteligente
SMA Sistema Multi-Agente
SMF Sistema de Manejo de Fallas
SMSA Sistema de Manejo de Situaciones Anormales
TDA Tipos de Datos Abstractos
TIC Tecnologías de Información y Comunicación
TDSO Técnica de Desarrollo de Sistemas de Objetos
UDP User Datagram Protocol
UML Unified Modeling Language

Parte I
Teoría de Agentes

Preámbulo

La puesta en operación de sistemas informáticos en procesos de alta complejidad requiere un esfuerzo significativo, de manera especial en las fases de modelado y construcción. La evolución y el desarrollo de la ingeniería de software ha contribuido, de manera fundamental en ello, para la puesta en operación de sistemas informáticos de alta complejidad.

Las técnicas de modelado y las herramientas de programación han ido evolucionando y acercándose cada vez más. Así, se ha evolucionado desde el enfoque de modelado donde se pensaba en un sistema como un "TODO" indivisible, y se usaba la programación monolítica; pasando por un esquema de división funcional orientado a módulos, que se programan independientemente, luego se desarrolló el modelado orientado a "objetos" que representan elementos propios del sistema; hasta llegar a enfoques donde el sistema se piensa como una agregación de elementos autónomos, que se modelan y programan como agentes.

La complejidad de los sistemas, y de sus componentes, requiere contar con mejores técnicas de modelado; y con altas capacidades de cómputo que permitan programar y ejecutar, de manera eficiente, los complejos diseños que pueden ser obtenidos para tales sistemas. Es posible afirmar que, gracias al importante crecimiento en las capacidades de cómputo y de almacenamiento, tales diseños pueden ser llevados a la práctica, surgiendo técnicas y herramientas que permiten que las abstracciones incorporadas en dichos diseños pueden ser programadas.

Por otro lado, se han incorporado a las técnicas de modelado de sistemas informáticos otras técnicas provenientes de diferentes áreas como la inteligencia artificial, sistemas complejos, base de datos, entre otras; lográndose una mayor capacidad de representación y de expresión formal/práctica, que permite capturar de mejor manera el comportamiento de un sistema. Es así como el uso de las abstracciones de agentes para diseñar y construir sistemas informáticos de alta complejidad ha ido creciendo y se ha convertido en una área de interés en diversos campos de aplicación.

En esta parte del libro se presenta una descripción de los agentes como técnica de implantación de sistemas computacionales. En el Capítulo 1 se presenta la conceptualización de los agentes, su caracterización y clasificación. Luego, el Capítulo 2 detalla los esquemas de organización de sistemas de múltiples agentes, describiendo algunos aspectos relacionados con la interacción en estos sistemas. Finalmente, en el Capítulo 3 se presenta una metodología para la especificación de sistemas basados en agentes; allí se describen las fases de dicha metodología y se muestra, a través de un ejemplo, como puede usarse esta metodología.

Capítulo 1
Agentes Inteligentes

Resumen: En este capítulo se presentan los conceptos y características que permiten entender que son agentes y como se relacionan en el contexto de sistemas.

1.1. Introducción

En este capítulo se presentan los conceptos y características que permiten entender qué son agentes y cómo están caracterizados. En este sentido, lo primero es definir el término "**agente**". Para dar una aproximación a tal definición, se toma como referencia la siguiente acepción de la real academia de la lengua española (www.rae.es): *Persona o cosa que produce un efecto y que obra con poder de otra*. Si bien esta definición se refiere al concepto amplio de agente, nos provee tres elementos fundamentales; el primero es que **produce un efecto**, el segundo es que **obra** (se ejecuta alguna acción) y el tercero es que lo hace en **función de otro**.

En la literatura existen diversas definiciones del término **agente**, desde distintos puntos de vista [7, 8, 1, 10, 6, 2], por lo cual aún no existe una definición universalmente aceptada. Entre estas definiciones podemos resaltar:

- Shoham: "Usualmente, cuando la gente usa el término *agente* se refiere a una entidad que funciona continua y autónomamente, en un entorno en el cual otros procesos ocurren y existen otros agentes" [8].
- Russel: "Un agente es una entidad que percibe su entorno y actúa bajo estas percepciones" [6].
- Franklin y Gasser: "Un agente autónomo es un sistema situado en, y como parte de, un entorno, que detecta dicho entorno y actúa en él, en búsqueda de sus propios objetivos ..." [2].

- Wooldridge y Jennings: “Es un sistema de hardware o, principalmente, software, que es autónomo, reactivo y social” [10].
- Ferber: “Un agente es un hardware o software que puede actuar sobre si mismo o sobre su ambiente. Además, tiene una representación parcial de su ambiente y puede comunicarse con otros agentes. Por otro lado, tiene objetivos individuales y su comportamiento es el resultado de las observaciones, conocimiento, habilidades e interrelaciones que él puede tener con otros agentes o con su ambiente” [1].
- Weiss: “Un agente es un sistema computacional que está situado en un ambiente, y que es capaz de tomar acciones autónomas en ese ambiente con el fin de cumplir sus objetivos de diseño” [9].

En [2] se puede encontrar otra amplia lista de definiciones para el término agente. Usando un combinación de estas definiciones, podemos decir que un agente es:

Definición 1.1. Un sistema informático situado en un entorno (ambiente), capaz de realizar acciones autónomas dentro de ese entorno para alcanzar sus objetivos.

La Figura 1.1 ilustra la noción más simple de un agente. En dicha figura se observa que el agente posee *sensores* para poder *percibir* su entorno, y *actuadores* para poder generar *acciones* sobre dicho entorno.

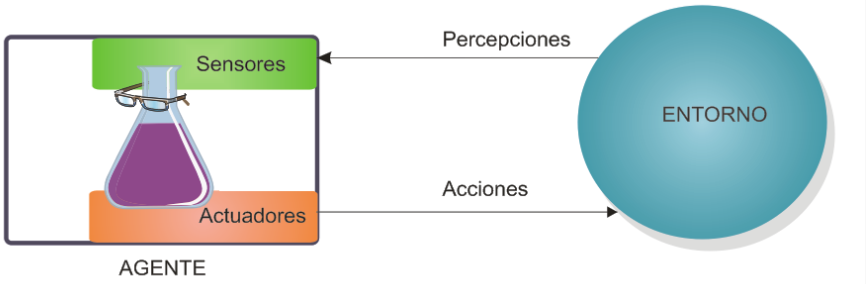

Figura 1.1: Esquema de interacción básica de un agente

1.2. Propiedades en los agentes

Dado que no existe una definición universalmente aceptada para los agentes, hoy día es más importante describirlos a través de sus propiedades o características. Así, un agente, para ser considerado como tal, debe tener una combinación del siguiente conjunto de propiedades [1, 2, 10]:

- **Reactividad**. Un sistema reactivo es aquel que mantiene interacción constante con su entorno, y responde a los cambios que ocurren en éste. Particularmente, la capacidad de emitir una acción inmediata al recibir una señal, o percibir un estado en el ambiente, es lo que caracteriza a los agentes reactivos. La reactividad en los agentes posibilita acciones rápidas, cruciales en sistemas de tiempo real, que no ameritan aplicar reglas complejas. Esta propiedad es fundamental en ambientes que cambian permanentemente. En general, construir aplicaciones para entornos dinámicos no es una tarea sencilla.
- **Proactividad**. La reactividad en un ambiente es relativamente simple (Estímulo → Respuesta) si se pueden describir todos los estímulos posibles, puesto que para cada estímulo se construiría la lógica que representa la respuesta. Pero como uno de los principales objetivos de los sistemas basados en agentes es que éstos actúen por nosotros, sus comportamientos, en algunos casos, deben estar basado en objetivos. En ese sentido, los agentes deben procurar tomar las acciones que permitan alcanzar las metas u objetivos del sistema. Para ello, sus acciones no pueden estar solamente dirigidas por estímulos, sino que ellos deben ser capaces de tomar iniciativas, planificar acciones, entre otras cosas, con el fin de alcanzar los objetivos del sistema. Este comportamiento es denominado proactivo.
- **Comunicación**. Es la capacidad de cada agente de conversar, intercambiar información, con otros agentes. La comunicación puede ser directa o indirecta. En el caso de comunicación directa se requiere de un lenguaje basado en ontologías (conjunto de conceptos, predicados, y relaciones entre ellos, que son entendibles por una comunidad de agentes) y realizar intervenciones, descritas como mensajes compuestos por dos partes: el *envoltorio* (indica los datos del agente emisor, de los agentes receptores, y el lenguaje, la ontología y el protocolo utilizado), y el *contenido* (compuesto por dos partes: una performativa que indica la acción general del mensaje, y la frase construida usando la ontología usada, que indica sobre que aspectos trata la performativa). La comunicación indirecta se basa en el uso de un espacio compartido por todos los agentes del ambiente (pizarra, cartelera), donde se deposita y se recupera la información. Estos últimos años, protocolos para este tipo de comunicación se han inspirado en investigaciones sobre las sociedades de insectos. Más adelante se retoma esta propiedad.
- **Sociabilidad**. La sociabilidad es la capacidad de obrar y cooperar recíprocamente con otros agentes (y posiblemente seres humanos), a través de un

lenguaje de comunicación de agentes. La razón de lo anterior es que en muchos casos, los agentes difícilmente alcanzarán sus metas sin tomar en cuenta a otros agentes. De hecho, algunas metas se pueden alcanzar solamente con la cooperación de otros agentes. En general, una sociedad de agentes es un grupo de agente que interactúan, se comunican, conversan, piensan y actúan en conjunto para lograr un objetivo común. En la literatura se han propuesto protocolos, basados en esquemas utilizados por las sociedades humanas, que llevan al logro de objetivos individuales o sociales. Algunos de ellos serán presentados más adelante en este libro.

- **Movilidad**. Es la habilidad del agente de moverse en el ambiente. Los agentes pueden migrar, trasladarse en una red de nodos de procesamiento, para ejecutar tareas específicas. Así, la movilidad tiene que ver con la capacidad de los agentes de moverse en un entorno dado. La movilidad implica contar con sistemas de nombramiento y localización, inherentes a los sistemas distribuidos.
- **Autonomía**. Según ciertos autores, la autonomía es la principal propiedad de los agentes. La autonomía es la capacidad de los agentes para actuar sin la intervención humana ni de otros sistemas externos; esto le permite tener un comportamiento propio, y reaccionar a los estímulos externos basándose en sus estados internos.
- **Veracidad**. Los agentes no comunican o devuelven información falsa a propósito, salvo que sean diseñados con esa intención; en cuyo caso estarían actuando solamente para cumplir sus objetivos. En ese sentido, en una comunidad de agentes se supone que todos los agentes proveen el resultado veraz de su actuación.
- **Benevolencia**. Siempre que no entre en conflicto con sus propios objetivos, los agentes ayudan a otros agentes a través de la prestación de servicios específicos. En ese sentido, en una comunidad de agentes todos los agentes son propensos a colaborar, a trabajar en conjunto.
- **Racionalidad**. Un agente actúa racionalmente en el sentido que no ejecuta acciones si éstas no lo llevan a cumplir sus objetivos. De manera general, los agentes tienen un conjunto de objetivos predefinidos y emprenden acciones para conseguirlos. La decisión de cual acción seguir, y en que momento hacerlo, es hecha según un principio de racionalidad: se prefiere la acción más prometedora o eficiente para conseguir sus metas.
- **Inteligencia**. Debido a que un agente debe analizar, ordenar ideas y conocimiento sobre el entorno, para llegar a una conclusión, y tomar una acción de forma autónoma, es necesario esta propiedad. Ella conjuga la capacidad de acumular conocimiento (aprender) y de usarlo adecuadamente (razonar a partir de él). Para implantar esta propiedad se pueden utilizar técnicas computacionales inteligentes como los sistemas expertos, las redes neuronales artificiales, la lógica difusa, etc.
- **Adaptativo**. Un agente es capaz de cambiar su comportamiento basado en las experiencias previas y las percepciones que ha hecho de su en-

torno. De esta manera, puede irse auto-reconfigurando para adaptarse a su entorno.

- **Asincronismo**: Un agente puede ejecutar acciones sin estar acoplado a un usuario o a otros agentes, es decir, sus acciones pueden llevarse a cabo por la ocurrencia de algún evento particular.

En la literatura se mencionan otras propiedades de los agentes como: la Persistencia y la Flexibilidad.

1.3. Caracterización de los agentes

Una vez descrito un conjunto de propiedades de los agentes, es posible realizar una caracterización de los agentes a través de un conjunto de elementos. Así, un agente se caracteriza como una entidad física o virtual que:

- es capaz de interactuar con su ambiente,
- puede comunicarse con otros agentes,
- tiene deseos (bajo la forma de objetivos individuales o funciones de satisfacción que busca optimizar),
- posee recursos propios,
- tiene una representación parcial de su ambiente,
- posee ciertas competencias y da servicios,
- puede reproducirse,
- su comportamiento trata de satisfacer sus objetivos.

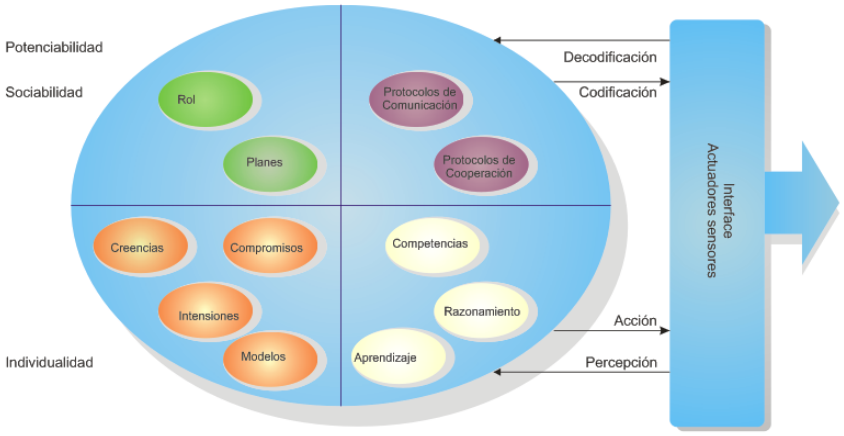

Figura 1.2: Arquitectura de un agente

La Figura 1.2 muestra la arquitectura de un agente. En esta figura se puede encontrar tanto las propiedades de los agentes como los elementos que permiten a un agente poseer tales propiedades. En la figura se distinguen cuatro cuadrantes:

- Cuadrante inferior izquierdo: este cuadrante está relacionado con los objetivos del agente. Incluye el modelo que el agente tiene de si mismo y del ambiente, sus intenciones, creencias y compromisos.
- Cuadrante superior izquierdo: está relacionado con el papel social del agente en la comunidad donde está inmerso; se incluye allí el rol del agente dentro de su entorno y los planes que el agente debe llevar a cabo para cumplir ese rol.
- Cuadrante superior derecho: describe las herramientas y mecanismos utilizados por el agente para comunicarse dentro de su entorno. Eso incluye tanto los estructuras y protocolos de comunicación, como las reglas que debe cumplir el agente para lograr la coordinación con otros agentes (ya sea para colaborar, no entrar en conflicto, entre otros).
- Cuadrante inferior derecho: cubre los elementos que permiten al agente poseer un comportamiento autónomo e inteligente (aprendizaje, conocimiento y competencias).

Los cuadrantes inferiores hablan de las capacidades *individuales* de un agente (sus conocimientos, competencias, objetivos, etc.). Los cuadrantes superiores describen las capacidades de *sociabilidad* de un agente (su rol, cómo comunicarse, sus planes, etc.) Los cuadrantes del lado izquierdo describen las *potencialidades* que el agente ofrece a su entorno. Finalmente, los cuadrantes derechos plasman los mecanismos, herramientas, etc., que posee el agente para actuar y comprender su entorno, en vias a *actualizarlo/modificarlo*.

1.3.1. Descripción práctica de un agente

Desde el punto de vista práctico, un agente puede ser caracterizado por:

- *sus Tareas:* el conjunto de actividades que ejecuta el agente, las cuales le permiten cumplir sus objetivos y/o funciones.
- *sus Conocimientos:* las reglas que sigue el agente para realizar sus tareas. Dicho conocimiento puede ser adquirido según alguna de las siguientes formas: especificadas por el diseñador (a través de algún formalismo), o incorporadas dinámicamente durante su evolución, ya sea porque provengan de otras fuentes de conocimiento (agentes, etc.) o se deriven de sus propios mecanismos de aprendizaje.
- *su Comunicación:* definen su forma de interacción con el ambiente y con los otros agentes, por consiguiente, se deben definir los lenguajes, protocolos de comunicación, entre otros, que utilizará.

Estos elementos están presentes en la arquitectura del agente que se ilustra en la Figura 1.2.

En general, los agentes realizan un ciclo continuo de observación y acción. Perciben la información a través de sus sentidos, y la transforman en conceptos que les permiten definir la situación en la cual se encuentran. De esta manera, comprueban si el objetivo perseguido ha sido alcanzado y en caso de una respuesta negativa deciden, según ciertos criterios, cuáles acciones deben ejecutarse para lograr la meta. A continuación, ejecutan dichas acciones.

Desde el punto de vista de implementación u operación, los agentes son objetos que describen los desarrolladores. Se asume que los agentes representan entidades del modelo, tales como procesadores, aplicaciones, controladores, etc. De esta manera, cuando se escribe un código para un agente, se deben considerar las siguientes partes:

- la estructura de datos, la cual tendrá las variables de estado del agente.
- las funciones (acciones) que manipularán (procesarán) la información que recibe el agente.

Es posible que el lector se haga la siguiente pregunta: **¿ Son los agentes y los objetos (de la programación orientada a objetos) la misma cosa?**. La respuesta a esta pregunta se pueda dar resumiendo los aspectos fundamentales de los objetos, y comparándolos con los aspectos fundamentales de los agentes. Básicamente, los objetos son abstracciones que:

- encapsulan estados,
- se comunican por pase de parámetros (mensajes),
- poseen métodos que se corresponden a acciones a ejecutar en algún estado.

Ahora bien, hemos dicho que los agentes tienen propiedades adicionales a las anteriores, de las cuales, para hacer la diferenciación con lo objetos, podemos destacar:

- **Autonomía**: los agentes incorporan una noción fuerte de autonomía. Particularmente, pueden decidir si ejecutar o no una acción, a petición de otro agente. También, pueden decidir proactivamente actuar, si para alcanzar sus objetivos lo requiere.
- **Flexibilidad**: los agentes pueden tener un comportamiento flexible (reactivo, proactivo y social). Esta característica no se contempla en un modelado orientado a objetos.
- **Actividad**: desde el punto de vista de implantación, cada agente es un elemento activo; por lo tanto, se asume que cada agente tiene al menos un hilo de ejecución.

1.3.2. Caracterización del ambiente

El ambiente del agente es un elemento fundamental, puesto que representa el "mundo" donde el agente vive. En este sentido, es muy importante lograr una descripción lo más precisa posible de las características del ambiente donde el agente se va a desempeñar. Esta descripción se logra a través de la identificación de las siguientes propiedades del ambiente:

- **Accesibilidad**. Un ambiente es accesible (observable) si el agente puede obtener información precisa, completa y actualizada de él.
- **Determinismo**. En una ambiente determinístico no hay incertidumbre acerca de los resultados que producirá una acción. De lo contrario, será estocástico. Los ambientes no determinísticos representan un problema más difícil para el diseñador de agentes.
- **Interdependencia de eventos**. Un ambiente se denomina episódico si el agente puede tomar acciones basado solamente en el evento actual, esto es, no se preocupa por la interrelación entre el evento actual y los eventos futuros. De lo contrario, será secuencial.
- **Dinamismo**. En un ambiente dinámico se pueden producir cambios en él, permanentemente, debido a procesos fuera del ámbito de la comunidad de agentes. De lo contrario, será estático.
- **Continuidad**. Un ambiente es discreto si hay un número fijo de acciones y percepciones en él, en el tiempo. De lo contrario, será continuo.

1.4. Clasificación de Agentes

En [4] se presenta un extenso análisis relacionado con la clasificación de agentes. Allí establece algunas dimensiones generales para definir los tipos de agentes. Estas dimensiones incluyen:

- **Movilidad**. Agentes móviles y agentes estáticos. Como se mencionó antes, un agente móvil es capaz de mover su código y su estado de un nodo a otro en una red. Un agente podría ser temporalmente móvil, y el resto del tiempo estático.
- **Proactividad**. Proactivos y reactivos. Cabe destacar que el comportamiento de un agente podría variar a través del tiempo. Incluso, bajo unas mismas condiciones de operación, un agente podría ser a veces reactivo y en otras proactivo (eso dependerá de las condiciones del ambiente y de la comunidad de agentes).
- **Propiedades mínimas (Autonomía, Aprendizaje y Cooperación)**. Basado en estas tres propiedades, Nwana [4] propone un esquema de intersección de propiedades que permite tipificar a los agentes de acuerdo a la existencia de ellas en él. La Figura 1.3 ilustra el esquema de intersección

de estas propiedades y los cuatro tipos de agentes que se producen: agente inteligente, agente interfaz, agente colaborativo y agente colaborativo con aprendizaje.

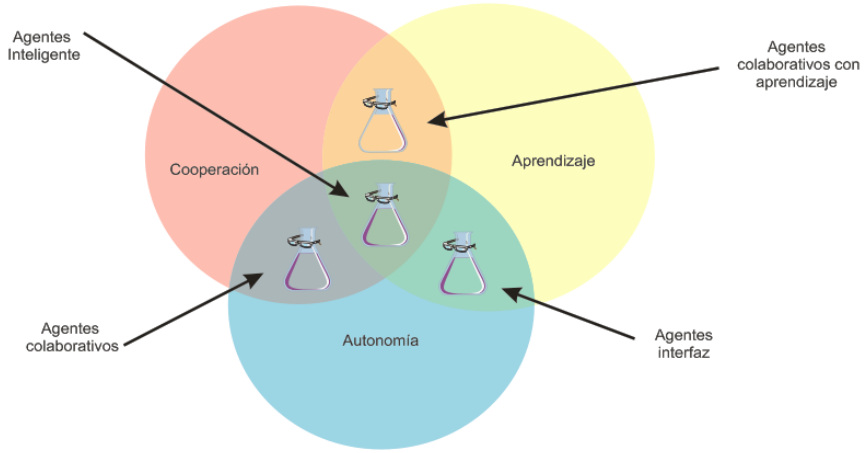

Figura 1.3: Tipos de agentes según sus propiedades

Esta tipología no contempla otra propiedades de los agentes. Ahora bien, esta clasificación deja claro que para que un agente se considere "inteligente" debe poseer al menos: autonomía, colaboración y aprendizaje.

Mansilla propone una clasificación de acuerdo a diversos aspectos [3]:

- **Relación entre percepción y acciones**
  - **Agentes de reflejo simple**. Cada percepción activa una regla simple, que se corresponde con la situación actual, y se efectúa la acción que está asociada a esa regla.
  - **Agentes bien informados**. Con la ocurrencia de cada evento se produce una actualización que permite comparar los estados del "entorno" y su evolución. Además, el agente registra la manera en que sus propias acciones están afectando ese entorno; de esta forma puede acceder a variables del entorno.
  - **Agentes basados en metas**. Es cuando con una acción se alcanza la meta deseada. Sin embargo, el agente debe ser flexible con respecto a la posibilidad de alcanzar diferentes metas, ya que al marcar una nueva meta se crea en el agente una nueva conducta.
  - **Agentes basados en utilidad**. La utilidad puede ser medida con alguna métrica que correlacione los estados del agente con algún valor cuantificable que caracterice su grado de satisfacción.

- **Tipo de Aplicación**
  - **Agente de interfaz o usuario**. Puede verse como un asistente personal, y está basado en la autonomía y el aprendizaje. Es un mediador entre algún subsistema, basado en agentes o no, y el usuario. Puede aprender y almacenar conductas del usuario y respuestas de otros sistemas, que permitan mejorar su rendimiento.
  - **Agentes de Búsqueda**. Pueden almacenar e interpretar patrones de búsqueda. Además, deben ser capaces de crear información de utilidad para el usuario en cuanto a la búsqueda de información, objetos, etc., ensamblando trozos de información recolectada.
  - **Agentes de monitoreo**. Estos agentes son "guardianes" que constantemente revisan parcial, o totalmente, las variables de un entorno, y toman algunas acciones según los valores de esas variables: almacenamiento, envío para despliegue, alarma, etc.
  - **Agentes de filtrado**. Trabaja en base a algún perfil predefinido. Interactúan con otros agentes, como el de monitoreo, para detectar cambios significativos en el entorno y tomar las acciones necesarias para manejar dichos cambios.
- **Características especiales**. En este aspecto la clasificación de Mansilla [3] es similar a la que presenta Nwana [4] para los casos de reactividad y movilidad.

1.5. Arquitecura de Agentes

La arquitectura de un sistema describe, de forma explícita, los componentes y las relaciones entre los componentes de éste. En caso de los sistemas basados en agentes, la arquitectura determina la forma particular de descomposición del sistema de agentes, y el esquema de comunicación con el ambiente y entre ellos.

Existen diversas arquitecturas de agentes propuestas en la literatura, una de las mas conocidas es la propuesta por [9]:

- *Agentes puramente reactivos:* estos agentes basan su decisión en el presente, sin revisar los eventos pasados. Ellos simplemente responden directamente ante eventos en su ambiente. Para ello solo requieren de mecanismos de percepción y de actuación sobre ese ambiente.
- *Agentes con estados:* estos agentes tienen una estructura interna, la cual almacena información acerca de su estado actual y de sus estados pasados. El agente toma su próxima acción basado en lo que percibe del ambiente y en sus estados pasados. Así, existe un mecanismo de toma de decisión en el agente que toma ambas fuentes de información.

- *Agentes con representación del entorno:* esta arquitectura tiene una estructura interna que permite la representación simbólica del ambiente y de los agentes, así como su manipulación sintáctica. Las acciones de los agentes son producto de procesos de razonamiento (inducción, abducción o deducción) que se dan sobre las representaciones simbólicas y sus derivaciones (sentencias lógicas, teoremas, etc.). Para lograr la implantación de esta arquitectura es necesario:

  - Representar el ambiente de manera precisa usando una descripción lógica del mismo.
  - Establecer el esquema de razonamiento de los agentes a partir de la representación simbólica.

- *Arquitecturas deliberativas:* en este tipo de arquitectura, tanto el comportamiento como el conocimiento de los agentes están representados de manera explícita, haciendo uso de algún modelo simbólico. Es una extensión del caso anterior, y al igual que en dicho caso, los agentes toman sus decisiones a través de procesos de razonamiento lógico.
  El ejemplo más importante de una arquitectura deliberativa es el de los agentes BDI. Un agente BDI es un agente racional que presenta actitudes mentales: Creencias (*Beliefs*), Deseos (*Desires*) e Intenciones (*Intentions*).

  - **Creencias**. Están representadas por la información y conocimiento que tiene el agente sobre el entorno (sobre si mismo y sobre otros agentes). Es decir, representan el conocimiento del agente acerca del mundo: qué es lo que el agente cree que es cierto acerca del mundo. Dicho conocimiento es usualmente almacenado en una base de datos especial, conocida como "base de conocimientos". Debe tomarse en cuenta que el conocimiento de un agente puede variar con el tiempo, incluso puede llegar a ser falso.
  - **Deseos**. Los deseos representan los objetivos del agente. Éstos pueden ser estados a los que quiera llegar el agente, o situaciones que el agente quisiera lograr o causar. El uso de objetivos añade la restricción de que dicho conjunto debe ser consistente, evitando metas concurrentes o contradictorias.
  - **Intenciones**. Las intenciones representan el estado deliberativo del agente; lo que el agente ha elegido hacer. Las intenciones pueden verse como los deseos en los cuales el agente tiene mayor grado de confianza. En menos palabras, son las metas que se fija hacer el agente para cumplir sus deseos, dadas sus creencias. Una buena parte de los progresos importantes en ingeniería de software se basan en el uso de nuevas abstracciones (abstracciones procedimentales, tipos de datos abstractos, objetos, agentes, etc.). Así, el uso del concepto de intencionalidad se convierte en una herramienta de abstracción que permite, de manera conveniente, describir, explicar y especificar el comportamiento de sistemas complejos. Los agentes, y de manera más específica los

agentes como sistemas intencionales, representan una abstracción que facilita el diseño y la especificación de sistemas complejos.

- *Agentes con arquitectura por capas:* se hace una descomposición del agente en capas, para separar los diferentes tipos de comportamientos de un agente: reactivos, pro-activos, etc. La vista horizontal nos indica que todas las capas del agente están conectadas a los mecanismos de percepción y acción del agente, de tal manera que cada capa se puede activar según lo que perciba el agente del ambiente. La vista vertical nos indica que en ciertos casos, para poder determinar la acción que realizará el agente a partir de lo que percibió, se debe pasar por varias capas (reactiva, de planificación individual, o de modelado de los otros).

Otra clasificación muy conocida, es la presentada en [5]. Allí se proponen las siguientes arquitecturas de agentes:

- *Agente reflexivo:* básicamente, un agente reactivo que actúa según la regla que active lo que percibe.
- *Agente reflexivo con estado interno:* en este caso, la regla que se activará dependerá de lo que percibe y de su estado interno.
- *Agente con objetivos:* en este caso, el agente tiene objetivos que alcanzar, por lo cual, eventualmente deberá realizar planes y resolver problemas.
- *Agente con función de utilidad:* esta arquitectura permite que varias reglas puedan, en un momento dado, tener las condiciones para activarse. En ese sentido, el agente debe seleccionar cual regla activar, para lo cual usa una función de utilidad.
- *Agente que aprende:* un agente con capacidad de aprender nuevas reglas, para adaptarse a su entorno. Para ello requiere de mecanismos para evaluar el entorno, extraer la información que le interesa, generar conocimiento, etc.

1.6. Otros aspectos importantes a considerar en los agentes

En el momento que se está concibiendo un agente, es fundamental estudiar ciertos aspectos. Muchos de ellos son los que le confiere la característica de “agente inteligente”. A continuación nombramos algunos de ellos (algunos ya han sido señalados en esta sección):

- *Mecanismos para resolver un problema:* un agente debe tener la capacidad de plantearse un problema y buscar la forma de resolverlo. Por ejemplo, un agente puede plantearse el problema de cual camino tomar para ir entre dos ciudades, o buscar un objeto en una sala. Para ello, el agente debe formular el problema (establecer la situación problemática, describir el ambiente, el estado actual, qué puede hacer, etc.), definir los objetivos a

alcanzar, establecer el algoritmo de búsqueda de la solución, y ejecutarlo. Se pueden usar diferentes algoritmos de búsqueda:

  - búsqueda ciega: no utilizan información sobre el problema, como los algoritmos que siguen una estrategia por extensión, o por profundidad.
  - búsqueda dirigida: usan información sobre el problema, como los algoritmos heurísticos que parten de una solución y tratan de mejorarla (llamados iterativos, como los que usan el descenso de gradiente o el temple simulado), o los que tratan de construir una solución paseando por el espacio de soluciones posibles.

- *Mecanismo para planificar sus actividades:* es decir, un agente debe poder elaborar planes de acciones para alcanzar los objetivos que se propongan. Además, en ciertas situaciones, debe tener la capacidad de replanificarse, establecer planificaciones condicionales, entre otros aspectos.
- *Mecanismos para representar el conocimiento:* un agente debe tener formas de representar el conocimiento que requiere para sus procesos de tomas de decisión, para después almacenarlo. En ese sentido, en la literatura se han planteado diversos formalismos de representación del conocimiento, desde los más simples basados en lógica proposicional, pasando por el clásico basado en lógica de predicados, culminando con los más complejos, como los que consideran la noción de tiempo (por ejemplo, los basados en lógica modal o en crónicas), o la incertidumbre (por ejemplo, los basados en lógica difusa o redes bayesianas).
- *Mecanismo de razonamiento:* un agente requiere de este mecanismo para poder usar de manera eficiente todo el conocimiento que tiene almacenado. De esta manera, basado en lo que percibe y conoce, puede tomar decisiones sobre que hacer. Estos mecanismos son los que permiten procesos de inferencia en un agente, ya sean del tipo inductivo (obtiene conclusiones generales a partir de premisas generadas por datos particulares), abductivo (dada una conclusión conocida, la idea es buscar hipótesis que las expliquen), o deductivo (parte de ideas generales para hacer afirmaciones sobre cosas particulares).
- *Mecanismos de aprendizaje:* es la manera de adquirir conocimiento de su entorno, y es vital para sus procesos de toma de decisión inteligente. En general, se dice que un sistema aprende cuando es capaz de modificar sus estructuras y/o funciones, de acuerdo con su experiencia, en vista a realizar mejor sus actividades (se adapta). Existen muchos mecanismos de aprendizaje en la literatura, clasificados según diferentes criterios. Por ejemplo, según el proceso de razonamiento que se sigue para aprender, existen los esquemas de aprendizaje inductivos o deductivos. Según el enfoque de retro-alimentación que se considere durante el aprendizaje, el mismo puede ser supervisado, no supervisado o reforzado.
- *Mecanismos de percepción:* son fundamentales para que un agente pueda captar su entorno. El poder captar del exterior símbolos o señales, mar-

can pautas en los procesos de toma de decisión. En ese sentido, lo ideal es que un agente pudiera percibir su entorno a partir de los mismos sentidos sensoriales que ha desarrollado el ser humano, a saber: la vista, el oído, el tacto, el olfato y el gusto. En general, en los ámbitos donde mas se ha avanzado es en la vista (particularmente en el procesamiento de imágenes: captar escenas, filtrarlas (suavizarlas, quitarles ruidos), segmentarlas en objetos, extraer características de los objetos, y reconocerlos) y en el oído (particularmente, reconocimiento de la voz y procesamiento del lenguaje natural).

Referencias

1. Jacques Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Professional, 1999.
2. Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Third International Workshop on Agent Theories, Architectures, and Languages*, pages 21–35. Springer-Verlag, 1996.
3. Luis E. Mancilla. ¿ qué son los agentes inteligentes de software? *Gaceta Ide@s CONCYTEG*, 3(31):25–47, 2008.
4. Hyacinth S. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11:205–244, 1996.
5. P. Russell, S.; Norvig. *Inteligencia Artificial: Un Enfoque Moderno*. Prentice Hall Hispanoamericana, Madrid, 2004.
6. Stuart Russell. Rationality and intelligence. *Artificial Intelligence*, 94:57–77, 1995.
7. Y. Shoham and K. Leyton-Brown. *Multiagent systems: algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2009.
8. Yoav Shoham. Agent-oriented programming. *Artif. Intell.*, 60(1):51–92, 1993.
9. G. Weiss. *Multi- agent System: a modern approach to distributed artificial intelligence*. MIT Press, New York, 1999.
10. Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

Capítulo 2
Sistemas Multiagentes

Resumen: En este capítulo se presentan los conceptos y características de los Sistemas MultiAgentes (SMA) y su aplicación en la Resolución Distribuida de Problemas (RDP).

2.1. Introducción

En diferentes dominios se han hecho esfuerzos por plantearse formas para la solución de problemas cooperativamente. La Inteligencia Artificial Distribuida (IAD) es uno de los dominios que ha estudiado ese problema, en este caso, usando conceptos provenientes tanto de la Inteligencia Artificial (IA) como de los Sistemas Distribuidos. En general, los sistemas basados en IAD son cooperativos, compuestos por al menos dos agentes, los cuales interactúan para la resolución de un problema dado [10, 17, 18]. Los agentes son frecuentemente heterogéneos, deben tener cierto grado de autonomía, con ciertas capacidades de razonamiento, planificación, aprendizaje, entre otros aspectos [5, 7, 11]. Además, los agentes deben ser capaces de interactuar, negociar, coordinarse, y hasta competir con otros, para llevar a cabo sus tareas, las cuales pueden ser individuales o colectivas [4, 17, 18].

Así, la IAD puede verse como un sub-campo de la IA, que tiene que ver con la resolución de problemas cooperativamente, tal que los agentes resuelven sub-problemas del mismo. Sus áreas principales son los Sistemas MultiAgentes (SMA) y la Resolución Distribuida de Problemas (RDP). Veamos la diferencia entre los dos [4, 10, 7]:

- Una RDP se basa en la idea de que para resolver un problema en particular, se debe descomponer el mismo en módulos (división de tareas), que serán resueltos cooperativamente por un grupo de agentes. En este caso, normalmente el diseño es *centralizado*, ya que los agentes son diseñados

para su integración en un sistema interoperado capaz de resolver el problema. El diseñador debe asegurar que los agentes utilicen el mismo lenguaje, que cada agente influya en la consecución del objetivo global, etc. En general, el procedimiento que sigue un sistema RDP consiste en cuatro etapas:

- *Descomposición del problema en sub-tareas.* En esta etapa, el problema que se solucionará se descompone en sub-problemas más pequeños. La descomposición será típicamente jerárquica, y los sub-problemas pueden seguir descomponiéndose en sub-problemas más pequeños sucesivamente, hasta que los sub-problemas estén en una granularidad apropiada para ser solucionados por los agentes individualmente. Ahora bien, no existe una única forma de descomposición (en sí, esta tarea es compleja), y puede ser planteada para ser realizada cooperativamente entre quienes conozcan las características del problema a resolver.
- *Asignación de las tareas y recursos entre los agentes.* Una vez culminada la fase anterior, en esta fase se asignan a los diferentes agentes el sub-problema a resolver. Aquí hay varios aspectos a considerar, entre otros, las habilidades de los agentes para asignarle la sub-tarea apropiada a resolver y la asignación de los recursos para no entorpecerse entre ellos cuando estén resolviendo sus sub-problemas.
- *Resolución de los sub-problemas.* En esta etapa, los sub-problemas identificados durante la descomposición del problema, se solucionan individualmente. Esta etapa implica típicamente compartir información entre los agentes.
- *Integración de las soluciones.* En esta etapa, las soluciones a los sub-problemas individuales son integradas en una solución total. Como en la descomposición del problema, esta etapa puede ser jerárquica, con las soluciones parciales integradas en diversos niveles de abstracción.

- Un SMA tiene que ver con el comportamiento de una colección de agentes autónomos tratando de resolver un problema dado, por lo cual comparten conocimiento acerca del problema y su(s) solución(es). Un SMA está formado por un grupo (comunidad) de agentes que interactúan entre sí, utilizando protocolos y lenguajes de comunicación de alto nivel, para resolver problemas que están más allá de las capacidades o del conocimiento de cada uno. Algunas de las características de los SMA son:

  - Cada agente tiene capacidad para solucionar parcialmente el problema.
  - No hay un sistema global que los controla.
  - Los datos no están centralizados.
  - La computación es asíncrona.

  Los SMA, por lo tanto, se refieren a los problemas de diseño de sociedades de agentes autónomos. Los SMA buscan responder a preguntas, tales

como: ¿Por qué y cómo cooperan los agentes? ¿Cómo los agentes pueden reconocer y resolver conflictos? ¿Cómo los agentes pueden negociar? [4, 10, 18], entre otras.

Algunas de las diferencias entre los SMA y la RDP son indicadas en la Tabla 2.1.

SMA	RDP
Se enfocan primordialmente en la coordinación de acciones o negociación entre los agentes	Se enfocan en la descomposición de tareas y síntesis de las soluciones propuestas
Es asíncrono	Necesitan sincronizarse
Permiten la emergencia	Son deterministas

Tabla 2.1: Algunas diferencias entre los SMA y la RDP

Algunos aspectos que debe considerar la IAD son: las características heterogéneas que pueden tener los agentes, los métodos para distribuir el control entre los agentes y los protocolos de comunicación entre los agentes.

2.2. Ideas de base de los SMA

Los SMA tienen una relación muy íntima con el campo de la IA. De hecho, muchos se refieren a los SMA como un subcampo de la IA, aunque los investigadores de los SMA consideran que la IA podría verse como un subcampo de los SMA. El esfuerzo actual de la IA está enfocado en la construcción de agentes inteligentes [5, 11], lo cual consiste en desarrollar componentes que permitan proveerle habilidades inteligentes a los agentes, entre las cuales tenemos: aprender, planificar, entender, razonar, etc. En cambio, en los SMA se realizan esfuerzos por integrar esos agentes [7, 18]. Pareciera que para construir un SMA quizás haya que solucionar todos los problemas actuales estudiados en la IA, porque sus agentes necesitarían aprender, planificar, etc. Ahora bien, en muchas ocasiones eso no es el caso, ya que no es necesario que un agente tenga todas las capacidades estudiadas actualmente en la IA para actuar en un ámbito dado. Algunas veces, incluso capacidades tales como aprendizaje pueden ser indeseables. En fin, podemos concluir que no necesitamos de todos los aspectos estudiados en la IA actualmente para construir SMA, y que la IA clásica no ha estudiado los aspectos sociales de un agente. Esas capacidades de comunicarnos, cooperar, y alcanzar acuerdos con nuestros pares (capacidades de socializar y negociar), que utilizamos a diario en nuestras vidas, tan importantes como el comportamiento inteligente de un individuo, son los aspectos estudiados en los SMA [4, 7, 18].

Ahora bien, el elemento fundamental de un SMA, como es de esperarse, es el agente. Un SMA puede ser visto como un sistema distribuido compuesto por agentes, donde la conducta combinada de ellos produce un resultado, en conjunto, inteligente [4, 10, 7, 17, 18]. En general, en un SMA se pueden distinguir tres niveles de organización [4, 7, 18]:

- *El nivel microsocial*, que se interesa esencialmente por las interacciones y conexiones que existen entre los agentes.
- *El nivel de grupo*, que se interesa en las estructuras que se producen en la composición de una organización compleja. En este nivel se estudian los roles y actividades de los agentes, la emergencia de las estructuras organizacionales entre los agentes, y el problema general de agregación de agentes para la constitución de organizaciones.
- *El nivel de la sociedad global (o poblaciones)*, que se interesa en la dinámica de un gran número de agentes, así como en la estructura general del sistema y su evolución.

Existen clásicamente dos enfoques para construir SMA [4, 10, 7, 17, 18]:

- *El enfoque formal*, que consiste en dotar a los agentes de la mayor inteligencia posible, utilizando descripciones formales del problema a resolver, y de hacer reposar el funcionamiento del sistema en tales capacidades cognitivas. De esta manera, la inteligencia es definida utilizando un sistema formal (por ejemplo, sistemas de inferencia lógica), el cual es usado para que los agentes generen un comportamiento racional, deduzcan nuevo conocimiento, planifiquen las acciones a realizar, entre otras cosas.
- *El enfoque constructivista*, que persigue la idea de generar un comportamiento emergente, a través de la elaboración de mecanismos ingeniosos de interacción, tal que el mismo sistema genere comportamientos inteligentes no necesariamente planeados desde un principio, o definidos dentro de los mismos agentes. Los agentes pueden ser muy simples.

En general, los SMA son, por definición, una subclase de los sistemas concurrentes. Por consiguiente, deben preocuparse de aspectos tales como la exclusión mutua, los abrazos mortales, y la hambruna, problemas clásicos en los sistemas concurrentes. Sin embargo, hay tres aspectos importantes muy propios de los SMA [4, 18]: los agentes se asumen que son *autónomos* (capaces de tomar sus decisiones para satisfacer sus objetivos de diseño); se asume que las *técnicas de sincronización y de coordinación* requeridas por un SMA están implantadas en la plataforma computacional que les da soporte; y los encuentros que ocurren entre los componentes del SMA es porque ellos están *interesados* (no se asume implícitamente que todos los componentes comparten un objetivo común, como en los sistemas concurrentes). En ese sentido, aspectos propios a los SMA, que no son estudiados en los sistemas concurrentes, tienen que ver con las formas para alcanzar acuerdos (negociación), y cómo pueden los agentes coordinarse dinámicamente sin conocerse.

También, otro vínculo que tienen los SMA es con las ciencias sociales. Las ciencias sociales tratan de entender el comportamiento de sociedades humanas, de animales, etc. Para el diseño de SMA es fundamental estudiar cómo funcionan esas sociedades (aproximarse a esos fenómenos de la vida real, para estudiar cómo esas sociedades alcanzan una clase particular de capacidad inteligente, llamadas por algunos autores *inteligencia colectiva*, para después intentar modelarla). Los SMA se inspiran y hacen uso de analogías y metáforas de sociedades humanas y de animales, pero también se apoyan en otras herramientas (tales como la teoría del juego), para construir formas de interaccíón entre los agentes. De esta manera, los SMA proporcionan una herramienta para modelar, estudiar y entender a sociedades; luego, las ciencias sociales representan un depósito rico de conceptos para construir SMA.

En los SMA aparecen un conjunto de conceptos de gran interés, que estudiaremos más adelante [4, 10, 7, 9, 11, 17, 18]: cooperación, resolución de conflictos, negociación, planificación, comunicación, entre otros. Según el acoplamiento de sus componentes, un SMA puede clasificarse en [4, 7, 18]:

- *Acoplamiento fijo:* cada componente posee un rol fijo (o eventualmente varios roles, pero todos fijos), al igual que las relaciones entre ellos son fijas. Así, la reorganización es imposible, por lo que es imposible adaptarse a su medio ambiente. Un ejemplo de ello son los clásicos sistemas computacionales.
- *Acoplamiento variable:* corresponden a estructuras organizativas fijas, cuya concretización (instanciación) puede variar. Las relaciones entre los agentes pueden cambiar, pero a través de mécanismos predefinidos bien precisos.
- *Acoplamiento evolutivo:* caracterizados por estructuras organizativas variables, pudiendo su concretización variar o no. Las relaciones abstractas entre los agentes pueden evolucionar (emerger), en función de los resultados de la organización (SMA).

Por otro lado, la estructura organizativa de un SMA puede ser vista de dos maneras [4, 18]:

- *Definida a priori por el diseñador*: se habla entonces de organizaciones predefinidas, lo que significa que las relaciones abstractas, sean estáticas o dinámicas, son predeterminadas. En el caso de organizaciones evolutivas, las relaciones abstractas posibles, y las transformaciones, son conocidas de antemano.
- *Definidas a posteriori*: se habla entonces de organizaciones emergentes. Éstas, que son muy comunes en comunidades de agentes reactivos, son caracterizadas por la ausencia de una estructura organizativa predefinida, ya que es resultado de las interacciones entre los agentes. En este caso, los roles y las relaciones no son determinadas de antemano, sino aparecen como el producto de la dinámica del SMA. Más precisamente, la repartición de funciones y tareas sigue un proceso de auto-organización, que

permite tener una organización que evoluciona y se adapta fácilmente a las modificaciones del medio ambiente y a las necesidades del grupo de agentes.

Finalmente, en cuanto a las relaciones de subordinación, existen dos tipos de SMA [4, 7, 18] (ver Figura 2.1):

- *Las estructuras jerárquicas*: suponen que las relaciones de subordinación forman una estructura piramidal. Una estructura jerárquica con acoplamiento fijo es un sistema centralizado (tipo militar), muy típico de los programas informáticos clásicos, expresándose como una llamada a un subprograma. Por otro lado, en el caso de un acoplamiento variable la estructura de subordinación generalmente produce una competición entre los componentes de bajo nivel, y los componentes de alto nivel se encargan de arbitrar esa competición (por ejemplo, seleccionar los vencedores).
- *Las estructuras igualitarias*: un agente puede pedirle a cualquier otro agente realizar una tarea, y este último eventualmente negarse. Estas estructuras son más características de organizaciones en las cuales todos los agentes intervienen uniformemente en la decisión final. Cuando el acoplamiento es fijo, los componentes actúan distribuidamente de manera predefinida en la toma de decisión. Con un acoplamiento variable, esta estructura genera organizaciones basadas en modelos económicos, en los que quienes presentan solicitudes y propuestas se organizan entre sí para encontrar un acoplamiento adecuado entre ellos.

El acoplamiento evolutivo puede seguir cualquiera de las estructuras de subordinación, la cual emergerá de su propia dinámica.

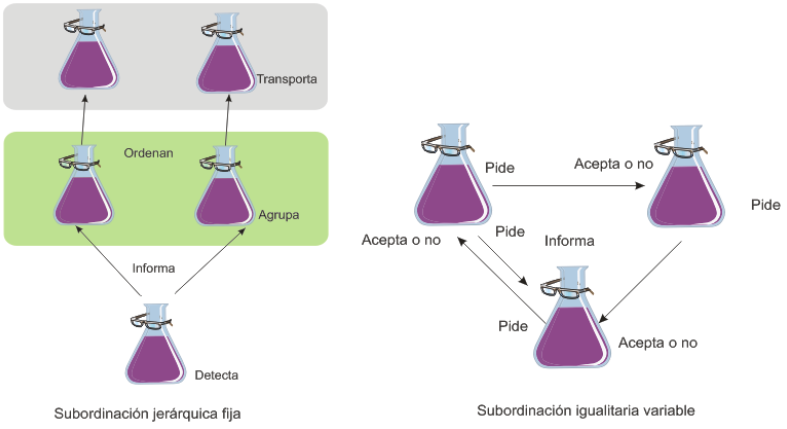

Figura 2.1: Ejemplos de estructuras de subordinación

2.3. La Interacción en los SMA

La cooperación entre los agentes es más que la suma de sus acciones, para lo cual se requiere coordinar éstas y resolver los conflictos que se les presenten. Todo ésto forma parte del problema de interacción en la teoría de agentes, en el que se estudian, analizan y conciben los diferentes mecanismos que les permiten obrar recíprocamente, para realizar sus tareas y satisfacer sus objetivos [4, 10, 7, 16, 17, 18]. La interacción le permite a los agentes, por medio de la comunicación, transmitir información, para eventualmente inducir en otro un comportamiento específico (es una acción particular que quizás pueda transformar el medio ambiente o modificar el estado mental del destinatario).

Así, el concepto de interacción es un problema central en los SMA; debemos entonces responder: ¿Cómo definimos formalmente una interacción?. En general, una interacción puede ser definida como el establecimiento de una relación dinámica entre dos o varios agentes, por medio de un conjunto de acciones reciprocas [4, 7, 18]. Normalmente, las interacciones se expresan a partir de una serie de acciones, cuyas consecuencias ejercen una influencia sobre el comportamiento futuro de los agentes. Los agentes obran recíprocamente a lo largo de un conjunto de eventos, durante los cuales, ellos entran en contacto, ya sea directo o por intermedio de otro agente o el medio ambiente. Las situaciones de interacciones son numerosas y distintas [4, 18]: compartir un recurso, intercambiar información, repartirse tareas, son ejemplos de situaciones de interacciones.

Generalmente, las interacciones son fundamentales para la constitución de organizaciones sociales. Es por los intercambios que mantienen, los compromisos que los vinculan, las influencias que se ejercen, los unos con los otros, que los agentes son entidades sociales, y que sus funcionalidades pueden emerger derivadas de las acciones mutuas. Esta es la razón por la cual no es posible analizar organizaciones sociales sin tener en cuenta las interacciones entre sus miembros. El concepto de interacción supone [4, 7, 17, 18]:

- La presencia de agentes capaces de actuar y/o de comunicarse.
- Situaciones susceptibles de servir de puntos de encuentro entre los agentes: colaboración, colisiones, utilización de recursos limitados, etc.
- Elementos dinámicos que permitan relaciones locales y temporales entre agentes: medios de comunicación, campos atractivos o repulsivos, etc.
- Un determinado "juego" en las relaciones entre los agentes, que les permite, a la vez, establecer una relación y separarse de la misma para actuar de manera autónoma.

Por otro lado, las interacciones le dan un aspecto plural a los SMA al aportar una dimensión suplementaria a cada individuo. Este último no es más el centro del universo, como en las concepciones clásicas de la IA, en las cuales, la inteligencia es caracterizada desde el individuo. Básicamente, de lo que estamos hablando es de comunidades de agentes, que a través de

interacciones entre si, o a través del ambiente, pueden agruparse dinámica y organizacionalmente para cumplir ciertos objetivos, ya sean individuales o comunitarios (ver Figura 2.2). Esta consideración de la interacción como elemento base de otras formas de inteligencia, es muy común en la *etología*, que estudia los comportamientos de animales y humanos, y las formas cognoscitivas producto de la interacción entre los individuos de su propia especie. Cuando este intercambio no existe, los individuos no pueden realizarse completamente y permanecen en un estado "primitivo". Ahora bien, a pesar de la importancia de las interacciones humanas, es poco lo que se ha hecho por tener en cuenta a las emociones en el diseño de las interacciones a nivel de agentes [18].

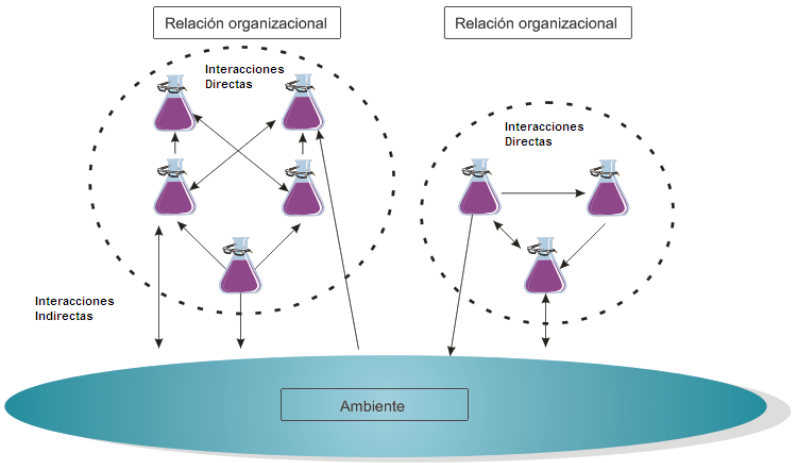

Figura 2.2: Estructura típica de un Sistema Multiagente

Las diferentes formas de interacción que estudiaremos son [4, 7, 17, 18]: colaboración, coordinación de acciones, resolución de conflictos, negociación, y cooperación. En general, ellas permiten determinar qué hace quién, cuándo, dónde, con qué medios, y de qué manera. La primera es de interés cuando se quiere repartir el trabajo entre varios agentes; la segunda analiza cómo las acciones de los diferentes agentes deben ser organizadas en tiempo y espacio, de manera que puedan alcanzar sus objetivos. Por otro lado, cuando aparecen conflictos, es importante poder limitar los efectos. Las técnicas de negociación sirven para satisfacer todas las partes del SMA en un momento dado, estableciendo compromisos y acuerdos. Finalmente, la cooperación es la forma más general de interacción en los SMA. Más adelante veremos que existen dos tipos de cooperación, la reactiva y la cognoscitiva.

Algunos aspectos que determinan las diferentes formas de interacción son los siguientes [4]:

- *Objetivos compatibles o incompatibles entre los agentes*: los objetivos y las interacciones de los agentes están interrelacionados a partir de la siguiente pregunta: ¿Los agentes tienen objetivos compatibles o contradictorios?. En general, la relación entre los objetivos y las interacciones nos da una primera clasificación de ellas: de cooperación (caso de objetivos compatibles, pero requieren ayuda entre ellos), de indiferencia (caso de objetivos compatibles, pero los agentes no requieren ayudarse), o de antagonismo (situación de competencia por tener objetivos incompatibles).
- *El uso de los recursos por parte de los agentes*: es otro componente importante para caracterizar las interacciones entre ellos. Se entienden por recursos los elementos ambientales o materiales (procesador, medio de almacenamiento, etc.) usados para realizar sus acciones los agentes. Los recursos, normalmente limitados, son el origen de conflictos. En este caso, las interacciones son perturbadas, desde el punto de vista de los recursos, si un agente entra en conflicto con otro en la realización de sus propias acciones. Esencialmente, dichos conflictos aparecen cuando varios agentes tienen necesidad de los mismos recursos al mismo tiempo, y en el mismo lugar. Estas situaciones pueden resolverse a través de mecanismos de coordinación de acción (dispositivos y reglas usadas para anticipar un posible conflicto, y procurar evitarlos antes que aparezcan), o resolución de conflictos (técnicas de negociación o arbitraje, etc.). Los conflictos pueden nacer, incluso, en situaciones donde todos los agentes tengan las mejores disposiciones para colaborar. Mas adelante volveremos de nuevo a este punto.
- *La capacidad de los agentes para realizar una tarea*: es otra fuente de caracterización de las interacciones. La pregunta clave es: ¿Un agente puede alcanzar solo un objetivo o necesita de otros para alcanzarlo?. En numerosas situaciones, un agente puede alcanzar su objetivo por si mismo. Objetivos complejos pueden ser alcanzados por un individuo, aunque su realización puede ser más fácil si participan otros agentes. Finalmente, hay otros objetivos que para alcanzarlos es necesaria la presencia de capacidades de varios agentes. En las dos últimas situaciones, las interacciones son beneficiosas para los agentes (las acciones de unos ayudan a los otros), y el producto que resulta es emergente.

Estos tres componentes de la interacción (naturaleza de los objetivos, el uso de los recursos, y las capacidades de los agentes) permiten establecer una tipología de las formas de interacción [4]:

- *Independencia*: objetivos compatibles, recursos suficientes, y capacidades suficientes. En este caso no hay necesidad de interacción entre los agentes (por ejemplo, las personas que caminan en la calle por aceras amplias).
- *Colaboración simple*: objetivos compatibles, recursos suficientes, y capacidades insuficientes. Aquí ocurre un tipo de interacción en el que se adicionan capacidades de los agentes, para lo cual se requieren de acciones

suplementarias de coordinación entre los agentes participantes (asignación de tareas, división de conocimientos, etc.).

- *Estorbo*: objetivos compatibles, insuficientes recursos, y capacidades suficientes. Esta interacción es el caso cuando los agentes se pertuban mutuamente cuando intentan realizar sus tareas (por ejemplo, usar un procesador por varios agentes). La administración de los recursos (horarios de uso, planificación de tareas, etc.) resuelve este tipo de problema.
- *Colaboración coordinada*: objetivos compatibles, insuficientes recursos, e insuficientes capacidades. En este caso, los agentes deben coordinar sus acciones para poder disponer de la sinergia del conjunto de sus capacidades. Los procesos industriales, en general, requieren de este tipo de colaboración. Es el caso más complejo de cooperación, puesto que añade al problema de asignación de tareas, los aspectos de coordinación debido a los recursos limitados.
- *Competición individual pura*: objetivos incompatibles, recursos suficientes, y capacidades suficientes. En general, cuando los objetivos son incompatibles, los agentes deben luchar o negociar para alcanzar sus objetivos. Se habla de competición pura cuando los recursos no son limitados (por ejemplo, un maratón). Aquí, el problema de interacción viene dado por el uso del recurso.
- *Competición colectiva pura*: objetivos incompatibles, recursos suficientes, y capacidades insuficientes. En este caso, los agentes deben agruparse en grupos o asociaciones para lograr sus objetivos, pero los grupos (eventualmente) no se molestan entre sí. Esta reagrupación ocurre en dos momentos: el primer momento es cuando se unen los agentes en grupos que interactúan de manera colaborativa coordinada, y el segundo momento es cuando eventualmente entran en competición los grupos entre sí (por ejemplo, la competición por equipo como las carreras de relevo).
- *Conflicto individual por recursos*: objetivos incompatibles, insuficientes recursos, y capacidades suficientes. Es el caso cuando los recursos no pueden ser compartidos, por lo que se da una situación de conflicto por el uso de los recursos, cada uno queriendo tener acceso exclusivamente a él (existen muchos ejemplos de este tipo de situación, como la defensa del territorio en los animales, la obtención de un cargo jerárquico superior en una organización humana, etc.).
- *Conflictos colectivos por recursos*: objetivos incompatibles, insuficientes recursos, e insuficientes capacidades. Este tipo de situación combina la necesidad de capacidades colectivas con la de conflictos individuales por recursos. Los grupos luchan entre sí para obtener el monopolio de un bien, de un territorio, o de una posición (por ejemplo, todas las formas de guerras, competición industrial, etc.).

Otra forma de entender las interacciones es a partir de las relaciones de dependencias entre los agentes. La idea de base es que una relación de dependencia existe entre dos agentes si uno de los agentes requiere del otro

para alcanzar una de sus metas. De esta manera, se podría clasificar las interacciones entre agentes como [4, 7, 18]:

- *Independiente*: no hay dependencia entre los agentes.
- *Unilateral*: un agente depende del otro, pero no viceversa.
- *Dependencia recíproca*: el primer agente depende del otro para una cierta meta, mientras que el segundo también depende del primero para otra cierta meta (las dos metas no son necesariamente iguales).
- *Mutuo*: ambos agentes dependen uno del otro para la realización de la misma meta. La dependencia mutua implica dependencia recíproca.

2.4. Comunicación

En esta era, la comunicación es uno de los aspectos más importantes en los sistemas computacionales. En el caso concreto de los SMA, la comunicación tiene que ver con la intención de intercambiar información entre los agentes, a través de la producción y percepción de signos en un sistema compartido de símbolos convencionales [1, 4, 10, 7, 11, 16, 18]. Las comunicaciones en los SMA, como en los humanos, son la base de las interacciones y de la organización social. Sin comunicación, el agente es un individuo aislado, sordo y mudo, encerrado en su propia percepción-razonamiento-actuación. La comunicación le confiere al agente un comportamiento social.

Así, la comunicación entre agentes permite la interacción entre ellos para sincronizar sus acciones; enviar y recibir conocimiento, datos o información; resolver conflictos en la resolución de una tarea; entre otros aspectos. Es a través de ella que los agentes pueden cooperar, coordinar sus acciones, realizar tareas conjuntamente, como verdaderos seres sociales. De esta manera, la comunicación es vista como un proceso social que integra múltiples formas de transmisión [4, 10, 18]: la palabra, el gesto, la mirada, la mímica, etc.

En específico, la comunicación es una forma de interacción en la cual la relación dinámica entre los agentes se expresa a través de mediadores, las señales, que, una vez interpretadas, van a producir efectos sobre ellos. Ese proceso interpretativo viene dado por el hecho de que en la comunicación de los agentes los mensajes, normalmente, tienen asociado un contenido semántico. De esta manera, la comunicación sirve para expresar, ordenar, solicitar, referenciar, verificar, entre otras consideraciones.

La comunicación se da en los SMA a través de actos de habla (acto de comunicación entre dos o más agentes en un momento dado), los cuales se agrupan en el marco de una actividad colectiva (de la comunidad de agentes) específica, para alcanzar un objetivo global o individual. A ese grupo de actos de habla del SMA que ocurre en la realización de una actividad dada, se le llama conversación [4, 18].

Algunos aspectos que se deben considerar para posibilitar la comunicación son [1, 4, 7, 11, 18]:

- *A nivel de la comprensión mutua:* debe haber formas de poderse entender, ya sea que exista una traducción entre lenguajes de representación, que compartan el contenido semántico, entre otras formas.
- *A nivel del medio de transporte:* deben haber acuerdos sobre cómo se envían y reciben los mensajes entre los agentes.
- *A nivel del lenguaje usado:* si no hay traductor, debe haber acuerdo en lo que significan los mensajes.
- *A nivel de la política de diálogo:* para poder establecer diálogos entre los agentes, deben haber acuerdos sobre cómo estructurar sus conversaciones.

Como dijimos antes, las comunicaciones se efectúan por medio de señales, cada una caracterizando un índice o signo. A nivel lingüístico y semiológico, se diferencian claramente los índices y los signos [4]. Un signo es el elemento más primitivo, es una marca, un rastro, una modificación del ambiente, eventualmente portador de una información para el que es capaz de percibirlo. Los índices son una forma de señal que *per se* no dice mucho, y deben ser usados en el marco de la elaboración de hipótesis sobre una situación o una cosa (son indicadores de algo). Por ejemplo, el rastro de animales constituye un índice del paso de los mismos, el humo puede ser el índice de fuego o de fábricas cercanas, los síntomas de un enfermo son los índices de la enfermedad que tiene. El proceso de razonamiento que seguiría un agente usando un índice para construir una hipótesis se llama abducción (la investigación y el diagnóstico siguen ese tipo de razonamiento) [5, 13, 11]. Esta concepción de la comunicación, que es muy poco usada en el mundo de la inteligencia artificial, es muy interesante, ya que descansa en mecanismos inferenciales y en la construcción de modelos del otro. Eso supone que el destinatario dispone de un conjunto de capacidades para realizar tal razonamiento.

En general, para que las señales se propaguen y signifiquen algo, necesitan dos condiciones [4, 9, 11, 16, 18]:

- Un agente con capacidad de percepción, para ser sensible a esa señal.
- Un sistema interpretativo capaz de transformar esa señal en un significado o comportamiento. Cuando la señal produce un comportamiento se dice que se trata de un estímulo. Cuando la señal requiere del sistema cognitivo para volverla inteligible, se dice que la señal es portadora de significado.

De esta manera, los agentes pueden razonar sobre las intenciones, deseos y objetivos de los otros agentes con los que se está comunicando.

Por otro lado, las señales se manifiestan en algo (una expresión concreta, un sonido, un movimiento, una figura, etc.), y llevan consigo algo más (una situación, un objeto físico, una acción, un procedimiento, otra expresión,

etc.) [4, 18]. Particularmente, las señales lingüísticas son esenciales para la comunicación entre agentes cognitivos, asociándoles significados, estados mentales, entre otras características, que deben ser interpretados por el receptor. Las señales lingüísticas son el resultado de un lenguaje, tienen una estructura de comunicación, siguen una fonética, sus sonidos se combinan para componer palabras, y dichas palabras se agrupan para construir frases.

A partir del concepto de comunicación en un SMA, pueden darse las siguientes situaciones [4, 18]:

- *No hay comunicación:* los agentes interactúan sin comunicarse, infiriendo las intenciones, los deseos, de otros agentes.
- *Comunicación primitiva:* está restringida a un número de señales fijas, con una interpretación establecida de antemano.
- *Arquitectura tipo pizarra:* es un espacio en el que los miembros del SMA escriben mensajes, dejan resultados, o encuentran información. A ese espacio todos los agentes tienen acceso.
- *Pase de mensajes:* es la forma clásica de comunicación que se usa en los sistemas distribuidos y paralelos, en los que simplemente se pasa información, sin ningún tipo de contenido semántico del mismo.
- *Comunicación de alto nivel:* la interacción entre los agentes se da a nivel de conocimiento (en vez del simbólico, como el anterior), lo que lo hace sólo entendible por aquellos que comparten dicho conocimiento (hay un contenido semántico compartido entre quienes participan en la comunicación).

La naturaleza del canal de comunicación (medio de transporte) juega un rol importante en la comunicación. Se pueden distinguir principalmente tres clases de mecanismos de transporte de mensaje [7, 1, 16, 18]:

- *El transporte directo* es el método más simple, cuando un agente decide enviar un mensaje, éste es tomado por el canal de comunicación y lo envía directamente a su(s) destinatario(s), sin tener en cuenta la distancia y sin darle la posibilidad a otros agentes de recibirlo.
- *El transporte por propagación de una señal* es un método de comunicación de los agentes reactivos. Un agente emite una señal que se difunde en el ambiente, cuya intensidad decrece en función de la distancia. Por lo tanto, entre más cerca se encuentre un agente de la fuente más fuerte será la señal recibida. Si esta señal es un estímulo, los agentes cercanos a la fuente que la encuentren tenderán más a tenerla en cuenta que los agentes situados más lejos. Esta característica es muy usada por los SMA reactivos, como mecanismo para generar diferenciación, regular la sociedad, entre otras cosas.
- *El transporte vía cartel* consiste en "pequeños anuncios". Un agente, si desea comunicarse, coloca su mensaje en un espacio común, visible por todos los agentes (o todos los de una clase particular).

La comunicación puede servir para diferentes funciones [4, 7, 18]:

- *Como función expresiva* que caracteriza la actitud del emisor: su estado, sus intenciones, sus objetivos, y cómo juzga y ve las cosas.
- *Como función de solicitud* de emprender o intentar algo, asociada a una orden. También se le asocian las respuestas de aceptación y denegación. Se trata de la función más importante en los SMA.
- *Como función referencial* en un contexto. Esta función permite la transmisión de información referente a hechos del mundo (estados del mundo o de una persona).
- Para *establecer, prolongar o parar una comunicación*.
- *La función poética* que sirve para valorizar el mensaje (ponerlo en valor). Es de poco uso en los SMA.
- *La función metalingüística* que se refiere a todo lo que permite clarificar un mensaje, su lengua y la situación de comunicación. Esta función es muy importante en los SMA, ya que permite caracterizar los mensajes en un discurso; los conceptos, el vocabulario y la sintaxis usada.

En general, los agentes no tienen la fuerza para obligar a otros agentes a realizar una cierta acción, ni pueden escribir sobre el estado interno de otros agentes. Pero un agente, a través de un acto comunicativo, puede intentar influenciar a otros agentes. Eso es lo más que se puede aspirar en un SMA, para garantizar la privacidad de cada agente, pero a su vez, para posibilitar que entre ellos se puedan influenciar.

2.4.1. Actos de Habla

Los actos de habla definen las acciones intencionales efectuadas en el transcurso de una comunicación (su conjunto constituyen una conversación) [4, 7, 16, 18]. La teoría de actos de habla establece la comunicación como una acción. Ella parte de la observación de que cierta clase de emisiones de frases desde el lenguaje natural - llamadas por nosotros actos de habla - tienen la característica de acciones, en el sentido que cambian el estado del mundo de una manera análoga a las acciones físicas. Puede parecer extraño esta afirmación, para ello veamos varios ejemplos: cuando se dice durante un matrimonio "los declaro marido y mujer", o cuando ciertas personas, en su discurso, dicen ciertas cosas (por ejemplo, un presidente define medidas económicas para su país, o le declara la guerra a otro país). Estos actos de habla, en las circunstancias apropiadas, cambian claramente el estado del mundo. La teoría de actos de habla establece un número de verbos, que llamaremos *performativas*, para darle esta connotación a los actos de habla. Ejemplos de tales verbos performativos son los usados para hacer una petición, para informar o prometer. Todo acto de habla debe considerar tres aspectos [4, 7, 18, 15]:

- El acto de hacer la emisión de la frase (cuando se pronuncia la frase).

- La acción que se realiza al decir algo (que se hizo después de pronunciarse la frase).
- El efecto del acto (el resultado de lo que se hizo).

La teoría de actos de habla también establece las condiciones requeridas para el procesamiento acertado de las performativas [4, 7, 18, 15]:

- Debe haber un procedimiento convencional aceptado, vinculado a cada performativa, tal que las circunstancias y las personas deben estar según lo especificado en el procedimiento.
- El procedimiento se debe ejecutar correcta y totalmente.

Una clasificación de los actos de habla es [4, 7, 16, 18, 15]:

- *Los descriptores:* sirven para dar información sobre el mundo, y en consecuencia describen los objetos o situaciones, afirmando algo de ellos (por ejemplo: Juan tiene 21 años, los rectángulos tienen cuatro ángulos rectos).
- *Los directivos o comandos:* son utilizados para dar directivas al destinatario (por ejemplo: lávate las manos). Buscan conseguir en el oyente que haga algo. Existen esencialmente dos actos de habla de este tipo: de solicitud de que se haga una acción, o de solicitud de suscripción. Estos actos de habla son iniciadores de la conversación. En el acto de habla de solicitud de una acción, o resolución de un problema, el agente receptor debe indicar si se compromete o no a realizarla. Si se compromete, entonces devuelve una notificación de final de la acción o el resultado (si se trata de resolver un problema). El acto de habla de solicitud de una suscripción tiene un principio muy simple: un agente A desea ser prevenido cuando un evento ocurre. La suscripción puede ser cíclica, por demanda, por eventos.
- *Los prometedores:* comprometen al orador a realizar algo en el futuro (ejemplo: vendré a la reunión, te prometo enviarte el archivo). Buscan crear confianza en el orador. Una forma de usar este acto de habla es cuando un agente propone sus servicios (ejemplo: realizar una tarea). Esta propuesta compromete al agente a realizar esa tarea, y debe ejecutar las solicitudes que correspondan a esa tarea.
- *Los expresivos:* sirven para dar al destinatario indicaciones relativas al estado mental, o psicológico, del orador (por ejemplo: soy feliz).
- *Los declarativos:* el mismo acto de habla es una acción al pronunciarse su enunciado (por ejemplo: declaro la sesión abierta, lo condeno a 2 años de prisión), generando cambios en una situación dada. Este caso lo comentamos antes, el cual está clásicamente vinculado al mecanismo de razonamiento abdutivo [5, 13].
- *Los interrogativos:* sirven para plantear preguntas, las cuales pueden ser sobre el mundo, sobre las creencias de otros agentes, sobre el estado mental del destinatario (cuáles son sus objetivos, sus compromisos, sus amistades, etc.) Plantear una pregunta es relativamente simple, y no existe un compromiso del destinatario por responder.

Existen otros actos de habla para expresar hipótesis, confirmar algo, indicar que se sabe hacer, etc. La lista anterior no es exhaustiva.

Finalmente, un acto de habla puede fallar por [1, 4, 7, 16, 18, 15]:

- El enunciado: se construye mal el acto de habla.
- La interpretación: no se entiende el acto de habla.
- La realización efectiva del acto de habla: no ocurre la comunicación como tal.

2.4.2. Conversaciones

Como ya se ha mencionado, las conversaciones son una secuencia de actos de habla que persiguen alcanzar un objetivo, o realizar una actividad del SMA [1, 4, 7, 16, 18, 15]. La teoría de actos de habla estudia dichos actos de manera aislada (el enunciado inicial, sus condiciones de aplicación, y los efectos locales entre los interlocutores), sin valorar la secuencia de interacciones que se establecen entre los interlocutores en sus comunicaciones, ni las posibles esperas que se dan (por ejemplo, un agente que plantea una pregunta espera una repuesta). Para ello, se requiere de una teoría de encadenamiento conversacional [4, 7, 16, 18, 15]. En ese contexto, los actos de habla son las unidades elementales que componen las conversaciones. Este enfoque se inspira en las conversaciones humanas, ya que no supone que existe a priori una única interpretación posible de un enunciado, al contrario, una frase sólo toma su sentido en el conjunto de reacciones que se suscitan en los interlocutores (su significado pragmático), y en consecuencia, en la acción que genera. Todo esto es construido a posteriori, como parte del contexto en que se da la conversación. En efecto, un acto de habla no se realiza prácticamente nunca aislado, a menudo es el origen de otros actos. De esta manera, todo acto de habla supone un determinado encadenamiento de enunciados, que genera modificaciones en los estados mentales de los interlocutores. Por ejemplo, una promesa tal como "te prometo venir mañana", constituye un compromiso del orador para efectuar un acto particular, el de venir, con un momento preciso, el día siguiente. Ahora, ese acto de habla puede ser el resultado de un conjunto de actos de habla previos que conllevaron a dicha promesa.

2.4.3. Lenguajes de comunicación para agentes

La teoría de actos de habla ha servido de base para numerosos lenguajes que se han desarrollado específicamente para la comunicación entre agentes. Nosotros sólo estudiaremos uno, el lenguaje mas conocido, el ACL

(Agent Communication Language) [4, 7, 9, 16, 18, 15]. Este es el lenguaje que ha venido estandarizando FIPA (Foundation for Intelligent Physical Agents) [6]. FIPA es una organización de la IEEE Computer Society que promueve la tecnología basada en agentes y la interoperabilidad de sus estándares con otras tecnologías. FIPA fue originalmente creada en Suiza, en 1996, para producir estándares de software para agentes y sistemas basados en agentes heterogéneos, y fue oficialmente aceptada por la IEEE en Junio del 2005. Una de sus tareas centrales es el desarrollo del lenguaje de comunicación ACL [6]. Este lenguaje está compuesto por un lenguaje "interno" (*KIF: Knowledge Interchange Format*), y un lenguaje "externo" (*KQML: Knowledge Query and Manipulation Language*). Así, un mensaje en ACL es una expresión KQML, en la que los argumentos son términos o sentencias KIF.

- *KIF:* es un lenguaje basado en la lógica de predicado de primer orden, con notación prefija, que permite representar conocimiento en los mensajes que se transmiten, con extensiones para soportar definiciones y razonamiento. Fue pensado para construir los contenidos de los mensajes de KQML como conocimiento. Así, usando KIF los agentes pueden expresar características de objetos (por ejemplo, José es vegetariano), relaciones entre objetos (por ejemplo, José y María son hermanos), características generales (por ejemplo, todos tienen una madre), entre otras. Para ello es que KIF usa la lógica de predicado de primer orden (sus conectores, cuantificador universal y existencial, etc.) Además, KIF proporciona un vocabulario básico de objetos: números, caracteres, y secuencias. También puede usarse para escribir un procedimiento. Un ejemplo de una expresión KIF es:
  $(> (sizechip1)(sizechip2))$
- *KQML:* es un lenguaje de comunicación de agentes que permite manipular conocimiento (KQML). Define un formato general para enviar mensajes, que le permite a un agente indicar explícitamente ideas. Los mensajes KQML no sólo comunican frases en algún lenguaje, sino que también comunican una actitud sobre el contenido. Para ello, cada mensaje tiene un primitiva (la clase del mensaje), y un conjunto de parámetros para clarificar el contenido enviado. Las primitivas del lenguaje reciben el nombre de "performativas" (definidas en la teoría de los actos de habla). Así, las primitivas son actos de comunicación con contenido semántico, y ellas pueden ser [4, 6, 7, 16, 18, 15]:

  - Las básicas: *evaluate*, *ask-if*, *ask-one*, etc.
  - Las de información general: *tell*, *achieve*, *cancel*, etc.
  - Las de preguntar: *stream-in*, *stream-all*.
  - Las de respuesta: *reply*, *sorry*.
  - Las de control de flujo: *standby*, *ready*, *next*, *rest*, *discard*, etc.
  - Las de definición de capacidades: *advertise*, *subscribe*, *monitor*, *import*,etc.
  - Las de gestión: *register*, *forward*, *broadcast*, etc.

La estructura de un mensaje KQML es la siguiente:

```
( Performativa
        : content (<Contenido a transmitir>)
        : receiver <destino>
        : language <lenguaje del contenido>
        : ontology <ontologia usada>
)
```

donde: *Performativa* es cualquiera de las listadas antes, *content* especifica el contenido del mensaje a enviar, el cual está escrito usando el lenguaje indicado en *language* (lenguaje en el que se expresa el contenido), dicho mensaje es enviado a quien se especifique en *receiver*, y todas las palabras usadas en content (la terminología usada en el mensaje) están definidas en el marco ontológico definido en *ontology*, el cual es conocido por los receptores y por quien envía. Diversas performativas requieren diversos parámetros. A pesar del gran aporte de KQML, hay algunos aspectos interesantes de nombrar [4, 7, 16, 18, 15]:

- Es ambiguo e impreciso. El significado de las performativas está formado de simples descripciones en lenguaje natural, lo que los vuelve a menudo confuso.
- Algunas de sus performativas son incoherentes.
- Faltan performativas, tales como las de promesa.
- Carece de formalización.

2.4.4. Ontologías para comunicación

En este libro, la idea de ontologías se presenta en los siguientes términos: Si dos agentes quieren comunicarse sobre un cierto dominio, entonces es necesario que estén de acuerdo con la terminología que utilizarán para describir ese dominio. Por ejemplo, imagínese que un agente está comprando un artículo particular de ingeniería (tuerca o perno) a otro agente: las necesidades del comprador para especificar al vendedor las características deseadas del artículo, tales como su tamaño, deben poder expresarse claramente. Para ello, los agentes necesitan poder estar de acuerdo en los términos a usar. Una ontología es usada con este fin, como una especificación de un sistema

de términos compartidos por los agentes (esto amplía la definición clásica de ontología para representar conocimiento).

Usando KIF se pueden desarrollar ontologías taxonómicas, ya que con él se pueden especificar las características de un dominio y las relaciones entre los componentes de ese dominio. De hecho, ya existen algunas, como Ontolingua [14]. Ontolingua es un servicio en Internet que proporciona una plataforma común en la cual las ontologías desarrolladas por diversos grupos se pueden compartir, y da una vista común de esas ontologías. El componente central de Ontolingua es una biblioteca de ontologías, expresada en el lenguaje de definición de ontologías de Ontolingua (basada en KIF). El servidor de Ontolingua es capaz, automáticamente, de transformar las ontologías expresadas en un formato a una variedad de otros. Además de KIF, se han desarrollado otros lenguajes para expresar ontologías, tales como WOL (*Web Ontology Language*), etc.

2.5. Coordinación

La coordinación de acciones puede describirse como el conjunto de actividades suplementarias necesarias a realizar en una comunidad de agentes para poder actuar colectivamente [1, 4, 7, 12, 16, 15, 18]. La coordinación trata de responder a preguntas del tipo ¿Cómo pueden dos actividades realizarse simultaneamente?. Los ejemplos de coordinación de acciones son múltiples, y se encuentran en todos los ámbitos de la vida humana: dos vehículos en un cruce deben coordinar sus acciones para no chocar; los acróbatas de un circo deben coordinar sus acciones con gran exactitud durante un acto; en un aeropuerto los aviones deben coordinarse para usar las pistas de aterrizaje y despegue, etc.

La actividad de coordinación se vuelve fundamental en el marco de la cooperación, y puede ser definida como la articulación de las acciones individuales realizadas por cada uno de los agentes, de manera que el conjunto tenga éxito [1, 4, 15]. Así, se trata de establecer en espacio y tiempo los comportamientos de los agentes, de tal manera que la acción del grupo mejore, o por lo menos disminuyan los conflictos. La coordinación de las acciones, es pues una de los principales métodos para garantizar la cooperación entre agentes autónomos.

La coordinación de acciones es necesaria por cuatro razones principales [4, 7, 16, 15, 18]:

- Los agentes necesitan información y resultados que sólo otros agentes pueden proporcionar.
- Los recursos son limitados, y muchas veces varios agentes quieren utilizarlos a la vez. Para realizar las acciones se necesitan recursos (tiempo, espacio, energía, dinero o herramientas), que no existen en cantidades infinitas. Es necesario compartir los recursos de manera de optimizar las

acciones que deben efectuarse (mejorar tiempos de respuesta, disminuir los costos, etc.), intentando al mismo tiempo evitar los potenciales conflictos.
- Se deben optimizar los costos, eliminando las acciones inútiles o evitando la redundancia de acciones. Por ejemplo, si dos personas deben ir al mismo lugar podrían utilizar un solo vehículo (eso mejoraría el tráfico).
- Se debe permitir que agentes con objetivos distintos, satisfagan sus objetivos.

El problema de la coordinación puede ser visto desde dos enfoques:

- *Coordinación global:* el sistema se planifica globalmente, existiendo un agente que realiza dicha tarea, verificando todos los conflictos presentes. Este es un enfoque centralizado, donde un agente coordinador es el responsable de detectar las interdependencias entre las actividades de los agentes locales.
- *Coordinación individual:* cada agente tiene completa autonomía para decidir qué hacer y resolver los conflictos que detecte localmente. Es descentralizado, ya que no existe un agente coordinador. Los agentes interactúan entre sí, y poseen el conocimiento para descubrir inconsistencias en sus acciones y adaptar sus decisiones locales.

Clásicamente, hay cuatro formas principales de coordinación de acciones [4, 7, 16, 15, 18]:

- *Coordinación por sincronización*. Se trata de la forma más básica de coordinación, ya que en general, la coordinación describe como debe ser el encadenamiento de las acciones, lo que induce necesariamente a la sincronización entre ellas. Se habla de sincronización cuando se trata de gestionar simultáneamente varias acciones, garantizando que el resultado de las acciones sea coherente. Los principales algoritmos de sincronización han sido estudiados en el ámbito de los sistemas distribuidos, y son utilizados de la misma forma en el contexto de los SMA.
- *Coordinación por planificación*. Esta técnica es la más tradicional en IA, se basa en descomponer la coordinación en dos fases. En la primera se reflexiona sobre el conjunto de acciones que se deben efectuar para alcanzar un objetivo, produciendo un conjunto de planes. En la segunda se elige uno de esos planes para ejecutar. Debido a que el medio ambiente puede cambiar, los planes pueden tener que revisarse durante su ejecución, lo que conlleva entonces a la replanificación dinámica. Además, en un SMA los diferentes planes elaborados por los distintos agentes pueden generar conflictos en los objetivos, o en el acceso a los recursos. Es necesario, entonces, coordinar los planes de manera de resolver dichos conflictos, y satisfacer así los objetivos de los diferentes agentes. En general, las técnicas de planificación garantizan una buena coordinación, pero son incapaces de tener en cuenta situaciones imprevistas o demasiado complejas.

Un caso de especial interés es la coordinación a través de la *planificación parcial global*, cuyo principio fundamental es que los agentes cooperan intercambiando información para alcanzar conclusiones comunes sobre el proceso de solución a un problema dado [8, 9, 12, 15, 17]. La planificación es parcial porque no se genera un plan para el problema entero, y es global porque los agentes forman planes intercambiando planes locales, y cooperando para alcanzar un propósito no local, la solución del problema. La planificación global parcial implica tres etapas iterativas.

- Cada agente decide cuáles son sus propias metas, y genera planes a corto plazo para alcanzarlas.
- Los agentes intercambian la información para determinar dónde obrar recíprocamente.
- Los agentes alteran sus planes locales para mejorar y coordinar sus actividades.

- *Coordinación reactiva*. Esta técnica considera que es más fácil aplicar mecanismos de coordinación en agentes reactivos que planear el conjunto de acciones y sus interacciones antes de actuar. Se basa en la idea de "percepción-acción" de un agente, bajo el esquema de definir acciones *in situ*, y no a priori, como las precedentes. Se pueden definir varios métodos de coordinación reactiva, según como usan el campo de acción (ambiente), si usan o no una representación del otro, entre otras cosas. Los enfoques reactivos no garantizan el máximo aprovechamiento de los recursos y resultados disponibles, pero son muy buenos en contextos cambiantes y en situaciones difíciles de anticipar. En este enfoque de coordinación emerge de forma natural la *división del trabajo* y la *asignación de tareas*.
- *Coordinación por reglamentación, normas y leyes sociales*. El principio consiste en establecer reglas o normas de comportamiento que permiten eliminar conflictos, o establecer comportamientos por defectos o misión (necesarios en tiempo real). Por ejemplo, se podrían asignar reglas para definir la prioridad entre vehículos en un cruce para evitar conflictos, o los tiempos de respuesta máximos permitidos para un agente dado (sino, hay una respuesta por omisión).
  Una norma es un patrón de comportamiento preestablecido. Los ejemplos de normas en sociedades humanas abundan. Por ejemplo, en el Reino Unido es una norma formar una cola para esperar un autobús. Esa norma no se hace cumplir de ninguna manera, es simplemente un comportamiento previsto, la divergencia en esa norma (generalmente) causa malas miradas. Sin embargo, esa norma proporciona una plantilla que es utilizada por todos para regular sus comportamientos. Así, las normas proveen a los agentes de plantillas para estructurar su repertorio de acciones. Ellas representan el equilibrio entre la libertad individual, por una parte, y la meta de la sociedad, en la otra. Como tal, dictan líneas de conducta que se seguirán en ciertas situaciones. Es importante acentuar el papel que juegan las normas en nuestras vida cotidiana. Las normas que to-

dos reconocemos como tal (por ejemplo, conducir en el lado derecho del camino), tienen una gran importancia en cada aspecto de nuestra vida social. Algunas cosas a considerar en las normas son:

- *Normas y leyes sociales emergentes.* Una cuestión clave es cómo una norma o una ley social puede emerger en una sociedad de agentes. Particularmente, la cuestión es cómo los agentes pueden alcanzar un acuerdo global en el uso de convenciones sociales, usando solamente la información local disponible. La convención debe ser global en el sentido que todos los agentes la utilizan, pero cada agente debe decidir sobre qué convención adoptar, según su estado interno (sin estructuras de poder ni relaciones predefinidas de autoridad). Note que no hay una opinión global posible, y un agente debe, por lo tanto, basar su decisión considerando dicha opinión. La idea es que cada estrategia de decisión local funciona tal que, cuando es utilizada por cada agente dentro de la sociedad, llevará a la sociedad a un acuerdo global tan eficientemente como sea posible. Un agente toma la decisión de que norma usar en un momento dado, usando diferentes estrategias, algunas de ellas pueden ser:
  - *Mayoría simple.* Los agentes cambiarán a una norma alternativa si han observado hasta ahora el uso de ella en muchos otros agentes, más que la norma que actualmente usan.
  - *Mayoría simple con comunicación sobre el éxito.* Consiste en una forma de comunicación basada en un umbral de éxito. Cuando un agente alcanza cierto nivel de éxito con una norma en particular, él lo comunica al resto de los agentes.
  - *Recompensa acumulativa más alta.* En este caso, un agente valora la norma particular que esta usando según su rentabilidad. La regla acumulativa más alta dice que un agente utilizará la norma, la cual considere le dé más rentabilidad acumulativa a la fecha. Cada cierto tiempo se debe recomenzar la memoria, es decir, el agente debe periódicamente "olvidar" todo lo visto hasta la fecha, se vacía su memoria, y comienza desde cero. Esto le permite a un agente "poder partir de nuevas ideas".
- *Diseño fuera de línea de normas y de leyes sociales.* La alternativa a permitir que las convenciones emerjan dentro de una sociedad es diseñarlas fuera de línea, antes de que el SMA comience a ejecutarse. El diseño fuera de línea de leyes sociales esta estrechamente vinculado al diseño de protocolos de comunicación para SMA.
  La pregunta clave, en este caso, consiste en como diseñar leyes sociales útiles. La respuesta es definir en un sistema estados focales, es decir, estados que son siempre legales, y que un agente debe siempre poder transitar. Una ley social útil entonces, es una que obliga a acciones de agentes para hacer eso posible. Ese problema es NP-completo.

Existe otra caracterización de la coordinación en función de si ella podía ser positiva o negativa [16, 15, 18]. Las positivas son las que existen entre dos planes cuando una cierta ventaja se puede derivar, para uno o ambos planes de los agentes, combinándolos. Tales situaciones pueden ser solicitadas explícitamente o no. Se pueden caracterizar tres tipos de relación no-solicitadas:

- *La relación de la igualdad de la acción*. Planeamos realizar una acción idéntica, y reconociendo esto, uno de nosotros puede realizar la acción solamente y así ahorrarle al otro el esfuerzo.
- *La relación de la consecuencia.* Las acciones en mi plan tienen el efecto secundario de realizar una de sus metas.
- *La relación del favor.* Una cierta parte de mi plan tiene el efecto secundario de contribuir al logro de una de sus metas, quizás haciéndolo más fácil.

Ahora bien, la coordinación en SMA va más allá de lo que se hace en computación clásica. En los SMA se considera al otro. Esto nos lleva a otra clasificación de la coordinación [4, 7, 16, 15, 18]:

- *Coordinación con intenciones comunes.* Las intenciones pueden desempeñar un papel crítico en la coordinación: proporcionan la estabilidad y la previsibilidad que es necesaria para la interacción social, y la flexibilidad y la reactividad que es necesaria en un ambiente cambiante. Por ejemplo, si usted sabe que estoy planeando estudiar para prepararme a un concurso próximamente, usted tiene la información necesaria para coordinar sus actividades con las mías. Con esa información, usted evitará invitarme a un viaje en los próximos días porque perturba mis estudios.

  Es importante distinguir la acción coordinada que no es cooperativa de la acción cooperativa coordinada. El ser parte de un equipo implica una cierta clase de responsabilidad hacia los otros miembros del equipo. Para ilustrar eso suponga que usted y yo estamos levantando un objeto pesado como parte de una actividad del equipo, entonces, ambos tenemos la intención de levantar el objeto. Suponga que creo que no va a ser posible levantarlo por alguna razón. Lo racional de hacer es tener la intención de comunicarlo a los demás. ¿Pero estaríamos inclinados a decir que cooperábamos?. El ser parte de un equipo implica que demuestro una cierta responsabilidad hacia el otro: si descubro que el esfuerzo del equipo no va a trabajar, yo debo por lo menos intentar comunicarlo a los otros.
- *Coordinación usando un modelo del otro.* Otro acercamiento a la coordinación, estrechamente vinculada a los modelos del trabajo en equipo humano, es el de la coordinación por el modelado mutuo. La idea es como sigue: recuerde el ejemplo de los vehículos que quieren atravesar al mismo tiempo un cruce - una colisión es inminente, ¿Qué debemos hacer?. Una opción es que se paren los dos, lo que garantiza que no ocurrirá ninguna colisión, pero esto es subóptimo: el recurso (el cruce) no se usa adecuadamente, pudiendo haber sido utilizado por uno de los vehículos. Otra

posibilidad es ponerse en el lugar del otro, es decir, construir un modelo del otro agente,(sus creencias, intenciones, etc.), y coordinar nuestras actividades alrededor de las predicciones que este modelo hace.

Los mecanismos de coordinación tienen las siguientes características:

- Temporales: rapidez, adaptabilidad, predictividad.
- Organizacionales: centralizada, distribuida.
- De calidad y eficacia: mejorando el rendimiento de los agentes, o eliminando conflictos entre ellos.
- De implementación: cantidad de información requerida, grado de representación de los otros, dificultad del mecanismo, etc.
- De generalización: manejo de la heterogeneidad.

2.5.1. Cooperación

Es un proceso por el cual un grupo de agentes generan deberes, mutuamente dependientes, para realizar actividades conjuntas (planes). En este caso, cada agente resuelve localmente aquello que es posible, y recurre a otros agentes para el resto de las tareas. Así, es una clase de coordinación que consiste en que varios agentes no antagonistas (acciones de uno no perjudiquen al otro) interactúen entre sí con el objetivo de conseguir un fin común [1, 4, 16, 15, 18].

La cooperación para alcanzar un objetivo O se logra, sí y solo sí los agentes comparten O como meta común. La cooperación es algo más que una coordinación accidental, los agentes deben tener el deseo y la preferencia por alcanzar O, y suponer que cooperar es interesante. Si para alcanzar O se deben alcanzar los sub-objetivos O_1, O_2..., O_n, entonces los agentes los convierten en sus objetivos inmediatos.

Algunas razones por las que es interesante cooperar son [4, 15, 18]:

- *Mejorar Rendimiento.* Para ello se requieren de índices de rendimiento (criterios de rendimiento). Los índices de rendimiento traducen la capacidad de un agente, o de un grupo de agentes, para realizar ciertas tareas. Los índices de rendimiento son variados, pero deben traducir las características intrínsecas de un colectivo (por ejemplo, para un grupo de robots recolectores de minerales, se podrá usar como índice de rendimiento la cantidad de mineral recogida por unidad de tiempo, o el número de acciones (movimientos) para recoger una cantidad de mineral dada). Algunas formas de mejora de rendimiento son:

  - Realizar tareas imposibles de realizar solo. En este tipo de situación, solamente la colaboración permite realizar la acción deseada.

  - Mejorar la productividad de cada uno de los agentes (basada en la idea de "economía de escala": la producción aumenta en función del número de agentes)
  - Aumentar el número de tareas realizadas en un tiempo asignado, o disminuir el tiempo para hacer una tarea.
  - Mejorar la utilización de los recursos.

- *Posibilitar la sobrevivencia.* Los índices de supervivencia reflejan la capacidad de un individuo, o de un grupo, de mantenerse mientras se confronta a fuerzas que tienden a destruirlo. Así, se pueden distinguir dos tipos de supervivencia: la supervivencia del individuo (la capacidad del agente de persistir), y la supervivencia colectiva (corresponde al mantenimiento del grupo). En el primer caso, la situación es simple, ya que se basa en el análisis de la supervivencia de un individuo; mientras que el segundo caso es más difícil, ya que se trata de determinar en que momento la organización colectiva desaparece. Se deben definir criterios para indicar que una organización colectiva dejó de existir (por ejemplo, una empresa sobrevive mientras tenga balances positivos anuales).
- *Resolver conflictos.* los conflictos surgen cuando al resolver un problema se da una o varias de las siguientes circunstancias: coexisten metas diferentes y divergentes en algún momento, hay diferentes criterios de evaluación de soluciones, los recursos son limitados, entre otros. A causa de su autonomía, un agente tiene la capacidad de determiner su propio comportamiento, por lo que los agentes pueden llegar a situaciones donde sus intereses pueden ser contradictorios con los de otros, como las listadas antes. En esos casos, es necesario utilizar técnicas de resolución de conflictos. Por otro lado, las situaciones de conflicto requieren de interacciones suplementarias para salir de los mismos: técnicas de negociación, de arbitraje, uso de reglamentos, o incluso de formas de competición.

Existen varios enfoques para el problema de la cooperación, según si se considera que la cooperación es una actitud de los agentes para trabajar conjuntamente, o que lo vea un observador que interpreta a posteriori los comportamientos, calificándolos de cooperativos o no [4, 15, 18].

- *La cooperación como actitud intencional.* En este caso, la cooperación es caracterizada como una actitud (postura) de los agentes. Se dice que los agentes cooperan si se comprometen en una acción común después de haber adoptado un objetivo común. Los ejemplos de asociación, como una asociación contra la violencia en el hogar corresponden a este esquema. El problema clave es el compromiso en un objetivo colectivo.
  En el caso de los agentes reactivos (que no tienen intención explícita, y aún menos modelos de los otros) el poder cooperar no existe en estos términos, pero si existe un tipo de cooperación sin ser conscientes de ello (los insectos sociales, como las hormigas, trabajan juntas para buscar comida, cooperando sin ser consciente de ello, al dejar sus trazas). Esto nos

dice que la cooperación no es solamente una actitud, sino que es el resultado de comportamientos. En ese caso, la cooperación no es ya el producto directamente de una intención de cooperación, sino del beneficio obtenido gracias a las interacciones de los agentes.

- *La cooperación desde el punto de vista del observador.* Es el caso cuando un observador califica las actividades de un conjunto de agentes. Por ejemplo, si se califica el comportamiento de las hormigas de cooperativas, es porque, como observador, se perciben una serie de fenómenos que se utiliza como índices de una actividad de cooperación. La idea de índices de cooperación es interesante, ya que permite liberarse de las características internas de los agentes, y de ocuparse de sus comportamientos observables. Algunos ejemplos de indices son [4]:

  - La división de los recursos, que se refiere a la utilización organizada de los recursos.
  - La robustez, que se refiere a la tolerancia a fallas en los agentes.
  - La no redundancia de las acciones.
  - La no persistencia de los conflictos.

  Estos índices deben poder ser medidos, para caracterizar el grado de cooperación que existe en grupos de agentes. Pero su aplicación no es evidente: ¿Cómo cuantificar la robustez de un sistema?. Además, este conjunto de índices debe ser coherente; por ejemplo, la medida de la robustez es una característica de la capacidad de un sistema para adaptarse, por lo que puede implicar cierta redundancia. Finalmente, estos índices pueden tener una relación jerárquica entre ellos, la división de los recursos es la consecuencia de evitar conflictos.

¿Cómo cooperar?, clásicamente, los métodos de cooperación son [1, 4, 16, 15, 18]:

- *La reagrupación y la multiplicación*. Consiste en que los agentes se acerquen físicamente, para constituir un bloque más o menos homogéneo en el espacio, o una red de comunicación. Por ejemplo, el mundo animal utiliza la reagrupación para garantizar un gran número de necesidades, y en particular, para mejorar la seguridad de cada uno de los miembros (defensa en grupo, una masa es más sólida, más difícil de atacar). Vivir en comunidad facilita muchas cosas: búsqueda de alimentos (puesto que basta que uno de los miembros encuentre algo para que todos los otros lo aprovechen), su reproducción, la adquisición de comportamientos complejos (por ejemplo, organizarse para trasladar un objeto muy pesado), la simplificación en la resolución de problemas complejos (como los de navegación), actuar como un organismo distribuido (unos se especializan en unas cosas y otros en otras). En general, todos los miembros actúan a la vez para su bien y para el del grupo. Por otra parte, la multiplicación es un aumento cuantitativo de los individuos, lo que representa ventajas

considerables, tanto desde el punto de vista del aumento de los resultados como del de la robustez del sistema.

- *La comunicación.* El sistema de comunicación que vincula a un conjunto de agentes actúa como un sistema nervioso que pone en contacto individuos. La comunicación les permite a los agentes beneficiarse de la información y de los conocimientos de los otros agentes. Las comunicaciones son indispensables para cooperar, lo que les permite a los agentes intercambiar información. En los sistemas cognoscitivos, las comunicaciones habitualmente se hace por envíos de mensajes, mientras que en los sistemas reactivos es el resultado de la difusión de una señal a través del ambiente.
- *La especialización.* Es el proceso por el cual agentes pasan a ser cada vez más adaptados a hacer ciertas tareas. Es a menudo difícil hacer que los agentes estén especializados en todas las tareas. La realización de una tarea supone características estructurales y comportamentales que no pueden permitir efectuar otras tareas con eficacia. La especialización no es necesariamente el fruto de una elección a priori. Agentes inicialmente totipotentes se pueden especializar progresivamente. Esa especialización normalmente es beneficiosa para el colectivo, aumentando la capacidad del grupo para resolver rápidamente un problema similar. Una sociedad especializada es más eficaz que un sociedad de agentes totipotentes, a causa de la disminución de los errores debido a una mejor sensibilidad para ir resolviendo los tipos particulares de problemas por parte de cada uno de los agentes especializados.
- *La colaboración por división de tareas y recursos.* La colaboración consiste en trabajar varios sobre un proyecto. En específico, la colaboración es el conjunto de técnicas que permitirán a los agentes repartirse las tareas, la información y los recursos, de manera de actuar en conjunto para resolver algo en común. Resolver un problema de colaboración consiste en responder a la pregunta, ¿Quién hace qué?. Existen numerosas maneras de repartirse las tareas y recursos. En los sistemas de agentes cognoscitivos, el proceso de asignación de tareas puede usar mecanismos de oferta y demanda. Existen los mecanismos de repartición centralizados, en los cuales un agente coordinador centraliza las ofertas y las solicitudes, y los reparte a continuación de la mejor manera posible; y los distribuidos, en los cuales todo agente puede ser a la vez oferente o solicitante, sin que exista un órgano centralizador. En este último caso, pueden haber dos formas: las que se basan en redes de amistades, y las que se basan en el concepto de mercado (redes de contrato). En el caso de los agentes reactivos, también es posible pensar en la asignación de tareas, a través de mecanismos donde los agentes se van especializando en la realización de una tarea. Un ejemplo de ello es el modelo de umbral de respuesta, donde cada agente va ajustando su umbral para responder a cada estímulo, siendo más alto el umbral para los estímulos de hacer tareas no deseadas, y más débiles en los otros casos.

- *La coordinación de acciones.* La sección anterior introdujo este problema. Básicamente, consiste en la definición del orden en que las acciones deben efectuarse.
- *Resolución de conflicto por arbitraje y negociación.* El arbitraje y la negociación son dos de los medios usados por los SMA para resolver conflictos. El arbitraje conduce a la definición de reglas de comportamiento que actúan sobre el conjunto de los agentes, que tiene como efecto limitar los conflictos y preservar a los individuos, y en particular a la sociedad de agentes. En la sociedad de humanos el comportamiento de los individuos es regulado por un conjunto de leyes y reglamentaciones que regulan las actividades sociales para el bien de todos. Existe un órgano de arbitraje, la justicia, cuya función es hacer respetar esas reglas colectivas, y velar por su aplicación. Este órgano actúa también para determinar las responsabilidades en los conflictos, y decide los castigos que las personas juzgadas culpables deben sufrir. En los SMA, el conjunto de reglas sociales actúa como un conjunto de comportamientos más o menos internalizados por los agentes. Obviamente, estas "leyes" no pueden aplicarse sino son agentes cognoscitivos de alto nivel, para que puedan ser capaces de pensar sobre los pro y contra de sus acciones. Otra forma de resolución de conflicto en los agentes cognoscitivos es no recurrir a un árbitro, sino buscar un acuerdo con los otros, conocido esto como un proceso de negociación (más adelante se estudia este caso).
- *Protocolo de votación.* Cada agente tiene una relación de preferencia sobre un conjunto de posibles acuerdos, y se elige una alternativa según un protocolo de votación. El alcanzar un acuerdo por votación se puede conseguir por medio de un protocolo de pluralidad (el mayor número de votos gana), un protocolo binario (series de votos con 2 opciones cada una), o un protocolo de "Borda" (En este caso se escoge aquella preferencia de los agentes que tenga el valor más alto, por lo cual cada agente debe revelar sus preferencias. Para esto, existe una función de bienestar social que asigna N puntos a la elección de todos los individuos, $N-1$ a la siguiente elección, y así sucesivamente, siendo N el número de preferencias posibles). En la lógica de votación se permite la formación de coaliciones: agentes pueden formar coaliciones (subgrupos) que cooperan con los miembros de su coalición, y compiten con los demás.

2.5.2. Negociación

La negociación consiste en poner de acuerdo a los agentes de un sistema, cuando cada uno defiende sus propios intereses, llevándolos a una situación que los beneficie a todos [4, 7, 16, 15, 18]. La negociación se resuelve con un plan común. El objetivo es determinar (las condiciones de) un acuerdo entre, al menos, dos agentes. Algunos tipos de negociación son [4, 7, 15, 18]:

- *Subastas:* consiste en adjudicar productos y tareas a través de un "mercado" con N participantes. Es un mecanismo estructurado para forjar acuerdos. Es un protocolo que requiere de diferentes roles: 1 subastador y N licitadores/subasteros. La estrategia consiste en las "pujas" de los subasteros a partir de un precio inicial dado por el subastador. Mas adelante presentamos en detalle este tipo de negociación.
- *Argumentación:* consiste en resolver (supuestos) conflictos a través del debate. Muchas veces un conflicto de interés es sólo aparente porque a un agente le falta información, o porque un agente ha sacado una conclusión equivocada. La argumentación es el proceso de intentar convencer a los demás de la veracidad de un hecho. Algunos modos de argumentación son: i) Modo lógico: "Si aceptas A y A implica B, entonces debes aceptar B", ii) Modo emocional: "¿Cómo te sentirías si eso te sucediera a ti?", iii) Modo visceral: "¡Cretino!".
- *Licitación:* a diferencia de la subasta, parte de un pliego de condiciones que construye un agente (quien licita, denominado cliente), donde especifica las condiciones en las que se requiere un servicio. Un grupo de agentes, llamados licitadores, hacen una oferta de realización del servicio según el pliego de licitación. En ese sentido, cada licitador ofrece su mejor propuesta para intentar ganar el visto bueno del cliente. Dicha oferta es evaluada por el cliente, y le asignará el servicio al agente licitador cuya oferta mejor responda al pliego de licitación. La evaluación consiste en revisar el cumplimiento del pliego de licitación establecido por parte de cada oferta, y en que condiciones son cumplidos (en la sección 2.5.3 volveremos a hablar de ella).
- *El regateo:* forja acuerdos globales ("creíbles") entre N agentes, tal que todos los agentes puedan beneficiarse de los acuerdos. Los elementos de un escenario de regateo consiste en un conjunto (espacio) de todos los posibles acuerdos a los que se pueden llegar (ejemplo: todos los precios entre las expectativas iniciales de un comprador y un vendedor). El protocolo de negociación se basa en reglas que determinan el proceso de negociación: ¿Cómo, cuándo, y qué ofertas se pueden hacer? ¿Cuándo termina la negociación y cuál es el resultado? (ejemplo: "No se puede empeorar una oferta ya hecha"). La estrategia de negociación consiste en cómo elegir entre las diferentes acciones que permite el protocolo (por ejemplo: "Mejorar mi última oferta en 10").

Cualquier técnica de negociación tendrá generalmente cuatro componentes:

- *Un sistema de negociación*, que representa el espacio de ofertas posibles que los agentes pueden hacer.
- *Un protocolo*, que define las interacciones, y su orden, que los agentes pueden hacer.
- *Una colección de estrategias*, una para cada agente, que determinan qué ofertas harán los agentes. Generalmente, la estrategia que un agente juega es

privada: el hecho de que un agente esté utilizando una estrategia particular no es generalmente visible a otros participantes de la negociación.
- *Una regla*, que determina cuando se ha alcanzado un acuerdo.

Una fuente de complejidad en la negociación es el número de agentes implicados en el proceso, y la manera como esos agentes obran recíprocamente. Hay tres posibilidades: negociación uno a uno, en la cual el agente negocia con otro agente; muchos a uno, en este caso un solo agente negocia con otros agentes (la subasta es un ejemplo); y negociación múltiple, en este ajuste muchos agentes negocian con muchos otros simultáneamente.

La capacidad de alcanzar acuerdos es una capacidad fundamental de agentes autónomos inteligentes, sin esta capacidad, seguramente encontraríamos imposible funcionar en sociedad. A partir de este hecho se llega a la noción de compromisos. Dicha noción es fundamental en la negociación, ya que los compromisos son el conjunto de acciones y actitudes que cada agente acuerda hacer/tener como parte del acuerdo. Son el fundamento, no sólo de la negociación, sino también de la cooperación, de planes y metas conjuntas; pudiéndose ver al compromiso como una elección individual, temporal y local, en un entorno distribuido o social.

2.5.2.1. Las subastas

Las subastas suelen establecer un contrato entre dos agentes (el subastador y el que gana la subasta) [4, 7, 15, 18]. Las subastas son técnicas muy útiles para asignar tareas a los agentes. En abstracto, una subasta ocurre entre un agente conocido como el subastador y una colección de agentes conocidos como los licitadores. La meta de la subasta es que el subastador asigne el producto a uno de los licitadores. En la mayoría de los casos, los deseos del subastador es de maximizar el precio en el cual se asigna el producto, mientras que los licitadores desean reducir al mínimo el precio. el subastador intentará alcanzar su deseo con el diseño de un apropiado mecanismo de subasta, mientras que los licitadores intentan alcanzar sus deseos usando estrategias. Los protocolos comunes de subasta son [4, 15, 18]:

- *Subasta inglesa:* el subastador ofrece un producto a un precio inicial (usualmente por debajo de un precio mínimo privado), y los subasteros van ofertando precios (ninguna, una, o varias veces). Cada oferta tiene que superar todas las anteriores, y el ciclo de apuestas termina cuando no hay más ofertas. La adjudicación se hace si la última oferta alcanza el precio mínimo (privado) del subastador, el producto es adjudicado al licitador con la oferta más alta, de lo contrario no se vende el producto (el subastador tiene la última palabra). La subasta inglesa es la más común, siendo del tipo primer-precio. La estrategia dominante es que un agente sucesivamente haga una oferta un poco más alta que la oferta actual, ver Figura 2.3. Allí, el agente 3 hace la oferta ganadora.

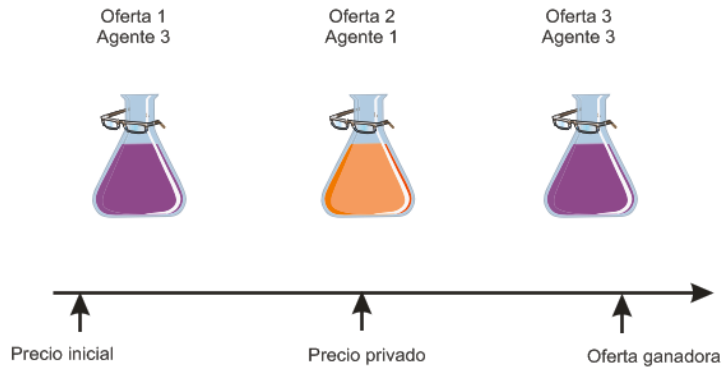

Figura 2.3: Ejemplo de Subasta inglesa.

- *Subasta holandesa:* al inicio, el subastador ofrece una cantidad de un producto a un precio inicial (por encima del precio mínimo privado), y lo ofrece por un cierto tiempo (Dt), después del cual disminuye el precio en una cantidad (D). De esta manera, el subastador va bajando el precio continuamente hasta que el comprador acepte la oferta actual. Cada oferta especifica la cantidad del producto a comprar al precio actual. El subastador determina el final de la subasta ya sea porque toda la cantidad ha sido adjudicada, o porque se alcanza el precio mínimo privado. La adjudicación de cada oferta a los compradores es directa. La subasta holandesa es un ejemplo de subasta descendente: el subastador comienza a ofrecer el bien a un cierto precio elevado; el subastador entonces baja continuamente el precio de oferta del bien en algún pequeño valor, hasta que algún agente haga una oferta para el bien en ese precio; el bien entonces se asigna al agente que hizo la oferta, ver Figura 2.4.

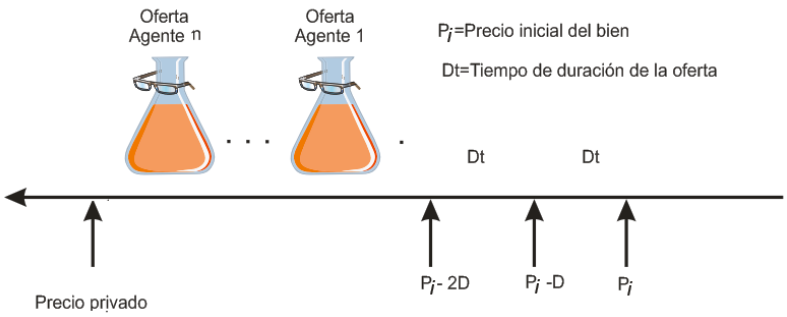

Figura 2.4: Subasta holandesa

- *Subasta sellada:* cada subastador realiza una oferta, sin conocer las ofertas de los demás. La mejor estrategia es ofrecer menos que el valor real.
- *Subasta de Vickrey:* subasta sellada de segundo precio, tal que gana el mejor postor, pagando el precio de la segunda mejor oferta. Esto significa que hay una sola ronda de negociación, durante la cual cada subastador hace una sola oferta; los subastadores no consiguen ver las ofertas hechas por otros agentes.
  Esta subasta es de las menos usuales, y quizás la más anti-intuitiva de todos los tipos de subasta que hemos presentado. El bien se concede al agente que hizo la oferta más alta; sin embargo, el precio que este agente paga no es el precio de la oferta más alta, sino el precio de la segunda oferta más alta.

Un agente apuesta con veracidad, independientemente de los demás postores. No se pierde ningún esfuerzo en contra-especular con otros agentes, esto permite una toma de decisión globalmente eficiente.

A pesar de su simplicidad, las subastas son una herramienta de gran alcance en los SMA, que puede ser usada para asignar mercancías, tareas y recursos. Las grandes preguntas que intentan responder los protocolos de la subasta son: ¿Cómo determinan al ganador?; ¿El tipo de subasta (subastas de primero-precio, subastas de segundo-precio, etc.)?; ¿Las ofertas hechas por los agentes las saben los otros?; y ¿Cuál es el mecanismo por el que se hace una oferta (por turnos, una sola rueda, el precio más bajo)?.

2.5.3. Asignación de tareas y recursos

La asignación de tareas y recursos es importante en los SMA. Al hacer hincapié en la idea de *trabajo distribuido*, con el concepto de contrato o compromiso, los SMA definen el problema en términos a la vez sociales y computacionales [4, 7, 16, 18], lo que lo distingue de las formas clásicas utilizadas en computación distribuida y paralela para resolver este problema (por ejemplo, asignación o repartición de procesos sobre procesadores) [2]. Repartir tareas, recursos e información implica responder a las preguntas: ¿quién debe hacer qué? y ¿con qué medios?, en función de los objetivos, roles y competencias de los agentes, y restricciones del ambiente.

El problema de repartición de tareas puede resolverse centralizadamente o distribuidamente. En cada caso se requiere de diferentes roles en la comunidad de agentes: proveedores, servidores, mediadores. En el caso centralizado, los mediadores son los responsables de la asignación. En un método de asignación distribuida, cada agente se ocupa individualmente de obtener los servicios de los proveedores que le pueden ser útiles para la realización de sus proyectos. A continuación presentamos algunos mecanismos de asignación [2, 4, 7, 15, 18]:

- *Asignación centralizada clásica.* La estructura de subordinación es jerárquica, y el nivel superior le indica a los subordinados las tareas a realizar. Se habla de asignación rígida o definida, como las llamada a procedimiento en la programación clásica.
- *Asignación centralizada con mediador.* Si la estructura es igualitaria, la asignación pasa por la definición de agentes especiales, los mediadores, quienes manejan el proceso de asignación, centralizando las solicitudes de los clientes y las ofertas de servicios de los servidores, con el fin de poner en correspondencia estas dos categorías de agentes.
  Un ejemplo muy simple es donde no existe más que un único mediador que dispone de una tabla que le indica el conjunto de agentes capaces de realizar una tarea particular, ver Figura 2.5. Esta tabla puede directamente actualizarla el proveedor que entra al SMA, indicando sus competencias. Cuando el agente A tiene necesidad de efectuar una tarea T que no puede (o no quiere) hacer, le pide al mediador encontrar a alguien para realizarla. Entonces, el mediador se dirige a los agentes que él sabe que poseen la capacidad de hacer T. Si uno de ellos acepta, el mediador envía la aceptación al cliente, de lo contrario le informa de que no tiene a nadie para realizar la tarea. Se supone que un proveedor que acepta realizar una tarea, se compromete efectivamente a hacerla. Se supone también que el mediador posee una tabla de las competencias de los agentes que es completa y coherente (todos los agentes que poseen una competencia están bien referenciados en la tabla, y toda referencia en esa tabla es buena). En fin, este sistema es sensible a fallas. En efecto, si el mediador falla todo el sistema se cae. Es necesario prever mecanismos de tolerancia a fallas complejos para intentar atenuar estos problemas. Por ejemplo, utilizando varios mediadores. Pero eso conlleva a otro problema, ya que es necesario poder gestionar la coherencia entre los diferentes mediadores.

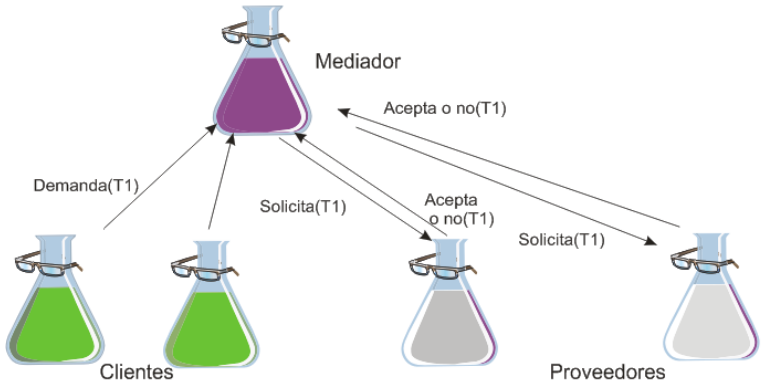

Figura 2.5: Esquema de un simple Mediador

- *Asignación por redes de amistades*. En este caso se supone que cada agente dispone de una tabla de competencias de los agentes que él conoce, en forma de un diccionario; es decir, cada agente posee una representación de algunos agentes (sus amigos) y sus capacidades.
  No se supone que cada agente conoce el conjunto de las competencias de todos los agentes, ya que eso violaría la hipótesis de representación parcial de los SMA, y su actualización seria muy pesada. Solamente es necesario que la red de amistades sea conexa (esto es, que exista siempre un camino que va de cualquier agente hacia otro, no importa que otro agente sea). Por otro lado, se supone que las tablas de amistades son correctas.
  Basado en lo anterior, existen esencialmente dos métodos de asignación por redes de amistades, ya sea que se autorice a los agentes a delegar sus solicitudes a otros agentes o no. Se hablará entonces de asignación directa ó de asignación por delegación:

  - En la *asignación directa*, el agente que quiere realizar una tarea intenta pedirle a cada uno de los agentes que él conoce, que pueden hacerla, que la haga, hasta que uno de ellos acepte. Si ninguno acepta, se alcanza un estado especial que es necesario tratar, aunque normalmente se trata de garantizar que en todos los casos habrá al menos un agente que aceptará efectuar la tarea. Para este estado especial se puede usar un comportamiento por omisión, como por ejemplo un mecanismo de concurso que permite elegir el agente que tiene menos razones para rechazar la tarea.

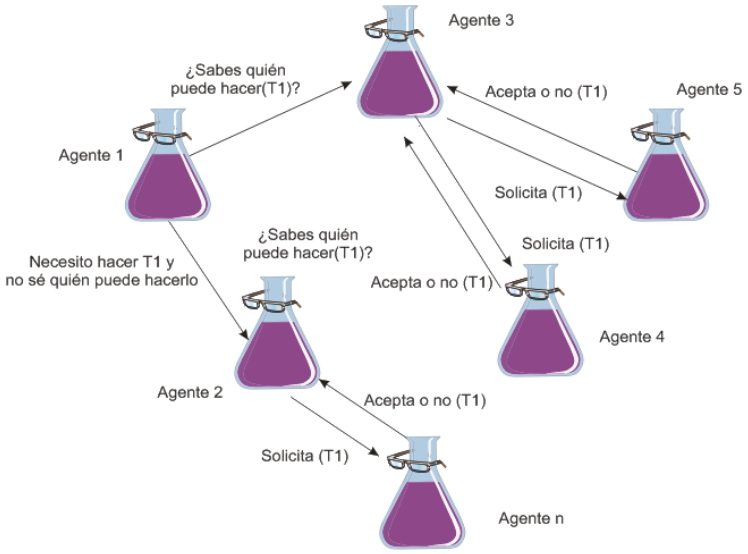

Figura 2.6: Asignación por Delegación

- La técnica de *asignación por delegación* permite conectar a clientes y a proveedores que no se conocen directamente. Cuando ninguno de los que conoce un agente son capaces de realizar una tarea que él solicita, este método de asignación permite pedirle a todos esos agentes que él conoce que busquen entre sus amistades si alguno puede efectuar dicha tarea (ver Figura 2.6). Este proceso se repite hasta que se encuentre a un agente que acepte realizar esa tarea, o que toda la red se haya recorrido. Para ello se requiere de algoritmos de recorrido paralelo de grafos, es decir, algoritmos de difusión que propagan recursivamente las solicitudes. Sin embargo, es necesario tener en cuenta dos dificultades: por una parte, verificar que se pide una única vez a un agente de realizar una tarea y, por otra parte, garantizar que el proceso termine, para pasar a otro método como el de concurso. El mecanismo de asignación por delegación requiere que cada agente tenga cuatro módulos:
  - Un módulo de delegación, que se encarga de enviar un mensaje de demanda a todas sus amistades, y de recibir sus propuestas.
  - Un módulo de evaluación de las propuestas, que se encarga de aceptar el primero que ofrezca hacer la tarea, y de rechazar al resto. Para esto, el agente informa su decisión a todos los agentes.
  - Un módulo de recepción de demandas, y de construcción de la propuesta. La misma es enviada al agente que la solicita.
  - Un módulo de recepción de la decisión, para saber si se le contrató para realizar la tarea.

- *Asignación por concurso:* es un mecanismo de asignación basado en el concepto de concurso (se inspira en el esquema de negociación por licitación). Es una estructura de control simple, pero su simplicidad oculta algunas dificultades. Se basa en la idea de la red de contratos, a partir del protocolo de elaboración de contratos en los mercados públicos. La relación entre el cliente o administrador, y los proveedores o licitadores, pasa por el intermedio de un concurso y de una evaluación de las propuestas enviadas por los proveedores. El concurso se efectúa en cuatro etapas (ver Figura 2.7):

  1. El administrador envía una descripción de la tarea a efectuar (pliego de licitación).
  2. A partir de esa descripción, los licitadores elaboran una propuesta (oferta) que envían al administrador.
  3. El administrador recibe y evalúa las propuestas, asignando la tarea a la mejor oferta.
  4. Finalmente, el licitador, que recibe la tarea, se convierte en el contratante, enviando un mensaje al administrador indicándole que está todavía de acuerdo en realizar la tarea requerida, y que se compromete, por lo tanto, a realizarla, o que él no puede aceptar el contrato, lo que reactivaría al protocolo.

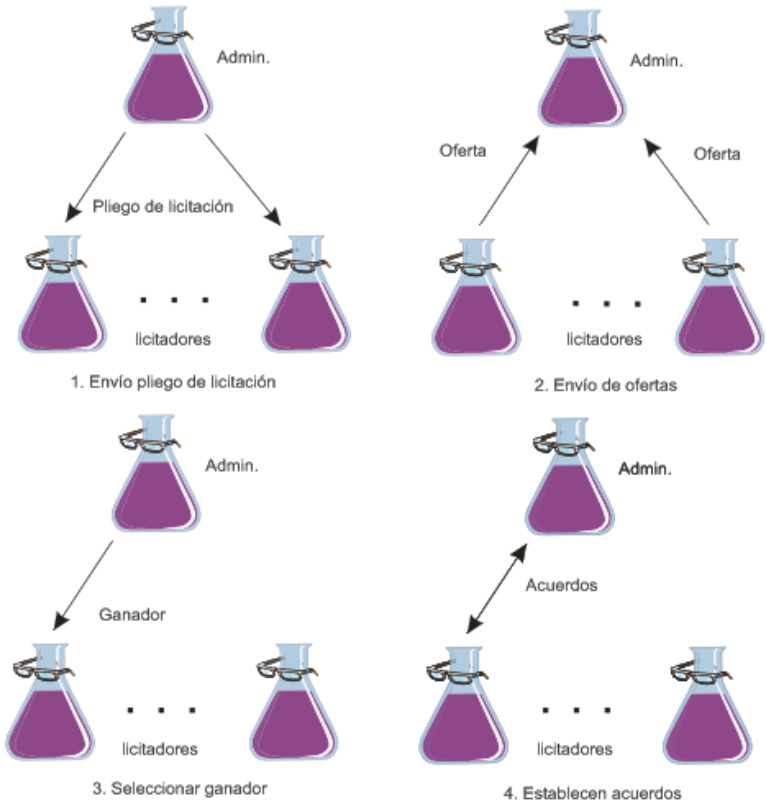

Figura 2.7: Asignación por Concurso

Esta asignación también es conocida como asignación por licitación. Este mecanismo debe prestar atención a los problemas de ignorancia temporal (distancia de los proveedores, lo que retarda la llegada de sus ofertas), e ignorancia espacial (los proveedores pueden estar interesados en ofrecer sus servicios a clientes que aún no conocen). También, deben preveerse momentos de rechazo, reservación, y los compromisos que acontecen en cada uno de ellos. Dichos compromisos pueden adquirirse rápidamente o más tarde, con las respectivas consecuencias. Finalmente, existe la posibilidad de subcontratar los servicios que vienen de ser asignados a un agente, tal que el proveedor vela por el servicio prestado por el subcontratado, de manera transparente para el cliente.

- *Asignación emergente:* es el menos conocido, característico de los sistemas reactivos, en los cuales las comunicaciones se efectúan normalmente por propagación de estímulos. En los casos anteriores, la asignación de tareas es hecha por agentes cognoscitivos, capaces de comunicarse intencionalmente, en forma de diálogos, hasta llegar a determinar que agente puede

efectuar un trabajo. La asignación reactiva recurre al concepto de señal, y no al mensaje. Las señales son formas no intencionales de comunicación, transmitidas generalmente por difusión y propagación en el medio ambiente. Las señales son formas de comunicación para las cuales no existe una semántica única: una misma señal podrá inducir dos comportamientos diferentes en dos agentes diferentes. El ejemplo más característico es el grito de un animal joven: en sus padres podrá iniciar un comportamiento de protección, mientras que en su depredador da lugar a un comportamiento de intensificar la caza. Así, este mecanismo define un arranque diferencial de un comportamiento en función de la señal que recibe un agente.
Un aspecto importante de las señales en la asignación de tareas, es que su intensidad decrece según la distancia a la fuente de emisión de la señal. Esa degradación de la intensidad de las señales en función de la distancia, introduce una diferencia en los agentes que perciben la señal. Los que se encuentran próximos a la fuente, para los cuales la señal es fuerte, y los que se encontrarán más lejos, y que por lo tanto sólo recibirán una señal atenuada. El principio de la asignación de tareas se basa, entonces, en la realización diferenciada de un comportamiento en función de la intensidad de la señal que un agente recibe. Por ejemplo, si un agente puede efectuar dos tareas T_1 ó T_2, y que estas tareas tienen asociadas respectivamente las señales S_1 y S_2, él ejecutara T_2 sí $S_2 > S_1$. Cuando las intensidades son idénticas, se elige aleatoriamente entre las dos acciones. La forma de comportamiento diferencial introduce dos aspectos a considerar en los SMA:

- La competición intra-agente, que se refiere al conjunto de tareas que un agente puede realizar. La toma de decisión de un agente reactivo depende de varios factores: la intensidad inicial, la tendencia a realizar una tarea u otra, la distancia a la fuente, y la capacidad de percepción del agente. Estos cuatro factores se combinan para la elección de la tarea a realizar. El modelo de umbral de respuesta es un ejemplo de ello [3].
- La relación de los agentes con la fuente de la señal. Como la intensidad relativa es más fuerte cuando uno se encuentra cerca de la fuente, los agentes más cercanos tenderán a realizar las tareas vinculadas a ese estímulo que los que están más lejanos. Pero, ¿Qué pasa cuando dos agentes deciden responder al mismo estímulo y que un único agente basta?. La solución normalmente consiste en que la haga el primero que la inicie. Normalmente, el más cercano a la fuente llegará primero (tendrá antes la fuente de datos), por lo que deberá ejecutar la tarea.

La asignación emergente tiene una gran capacidad de adaptabilidad. Esta técnica es independiente del número de agentes, además, entre más grande sea el número de agente más es la oportunidad de responder a las solicitudes de realización de tareas. Si se suprime a un agente, el SMA

sigue funcionando como si nada hubiera pasado. No se necesitan protocolos complejos, ni la verificación de culminación.
Ahora, existe un problema, y es que los agentes reactivos pueden ser encuadrados a zonas de señales procedente de solicitudes específicas, tal que los agentes son sordos a llamadas de otros agentes (a otros estímulos). Se puede resolver esta dificultad con un gran número de agentes (entre más agentes existan más es la probabilidad de que algunos agentes puedan responder a esas demandas), o aumentando continuamente las intensidades de las señales mientras sus solicitudes no sean satisfechas. Si se puede hacer crecer la intensidad, todas las zonas se cubrirán, tal que los agentes podrán percibir en un momento dado esos estímulos.

Pueden desarrollarse enfoques híbridos que intenten aprovechar las ventajas de los diferentes enfoques presentados anteriormente.

2.5.4. Planificación

Un problema en los SMA es el de planificar las actividades de un grupo de agentes. El planeamiento multiagentes es diferente al de un agente individual, ya que se debe tomar en consideración el hecho de que las actividades de los agentes pueden interferir entre ellas; sus actividades deben, por lo tanto, ser coordinadas [4, 8, 9, 12, 15, 17, 18]. Así, el problema de la planificación en los SMA no es simple, al contrario, si es difícil para un único agente determinar las consecuencias de sus acciones, la introducción de agentes suplementarios no hace sino aumentar la complejidad. En primer lugar, la multitud no puede sino empeorar el riesgo de modificaciones no prevista por el sistema de planificación, lo que puede necesitar de numerosas replanificaciones. Y en segundo lugar, la interdependencia entre las acciones se amplifican, lo que aumenta las dificultades de obtener un orden conveniente para la ejecución de las acciones.

La planificación en un SMA puede descomponerse en tres etapas: la construcción de planes, la sincronización/coordinación de los planes, y la ejecución de estos últimos. Como es posible utilizar uno o más agentes en cada una de estas etapas, es posible obtener un gran número de organizaciones diferentes. Un sistema de planificación es entonces compuesto de un conjunto de agentes que pueden planear, sincronizarse o ejecutar los planes, pudiendo un mismo agente realizar una sola o varias de esas tareas. Así, la planificación multiagente puede recurrir a varios planificadores, varios sincronizadores/coordinadores y, evidentemente, a varios ejecutores [4, 8, 12, 15, 18].

En el caso de la construcción de planes, se puede recurrir a un único agente (se habla entonces de planificación centralizada) o al contrario, distribuir la tarea de planificación en cada uno de los agentes (en ese caso, cada agente

construye un sub-plan que satisface sus objetivos locales, sin tener en cuenta a los otros agentes). En el último caso, los planes se deben integrar, por lo que es necesario coordinarlos. Esta tarea puede ser delegada a un único agente, quien centralizará todos estos sub-planes e intentará sintetizarlos en un plan global, o dejarlo en cada uno de los agentes, coordinándose entre ellos a través de técnicas como la negociación. Sólo para esa primera fase ya se ve la multiplicidad de posibles organizaciones de un sistema de planificación para un SMA. Veamos los métodos de organización clásicos del sistema de planificación en los SMA [4, 8, 12, 15, 17, 18]:

- *Planificación centralizado para planes distribuidos:* es un sistema de planificación centralizado que genera planes individuales para un grupo de agentes, en los cuales la división y el orden de trabajo ya están definidos. En este caso se realiza un plan general (el planificador) y se identifican los planes que se pueden realizar en paralelo, para asignarlos a los agentes respectivos según sus capacidades. Se supone que existe un sólo planificador, capaz de planear y organizar las acciones del conjunto de agentes. Este agente trata también de resolver la sincronización de los planes. El resto de agentes simplemente ejecutan los planes asignados (ver Figura2.8). Las técnicas de construcción centralizada de planes se efectúan generalmente en tres tiempos:

  1. Se busca un plan general, que pueda expresarse en forma de un grafo acíclico.
  2. Se determinan las ramas que pueden ser ejecutadas en paralelo, y se introducen puntos de sincronización cada vez que dos ramas se deban juntar.
  3. Se asigna la ejecución de las ramas a los diferentes agentes.

  El problema de coordinación se resume a una simple combinación de procesos de sincronización y de asignación de tareas. En el caso de la asignación, la misma puede realizarse de dos formas:

  - *Gestión dinámica de la asignación.* Una gestión dinámica de la asignación puede ser efectuada por un mecanismo de asignación de tareas como los vistos en la sección 2.5.3. Se debe manejar tanto la sincronización como la asignación de tareas de manera dinámica, considerando a los ejecutores como recursos quiénes están a priori de acuerdo para efectuar el trabajo.
  - *Gestión estática de la asignación.* la asignación a los agentes es efectuada al inicio de la ejecución de los planes, cada uno sabiendo lo que debe hacer y cuando. La coordinación se resume a una simple sincronización. Dicha asignación es hecha *a priori* por el planificador (o cualquier otro agente centralizado), quien precisa que agente debe hacer qué.

- *Coordinación centralizada para planes parciales.* Es posible también no centralizar más que la coordinación. En ese caso, cada agente construye in-

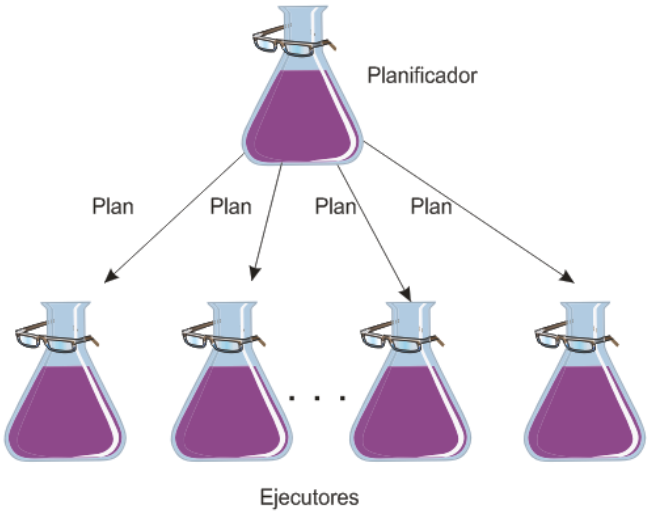

Figura 2.8: Planificación centralizado para planes distribuidos

dependientemente su propio plan parcial que envía al coordinador. Este último intenta entonces sintetizar todos estos planes parcial en un solo plan global coherente. Se trata, entonces, de resolver/suprimir los conflictos potenciales, o de determinar los puntos de sincronización necesarios. El coordinador debe fusionar los planes parciales procedente de los distintos agentes de manera coherente (ver Figura 2.9). En ese caso, aparecen varias posibilidades, según que las acciones de los planes sean independientes, positivas o negativas.

- *Acciones independientes*: No hay ningún lugar común entre los planes, por lo que es posible fusionarlos. Al ser independientes, la ejecución de sus acciones en paralelo no da problema.
- *Relaciones positivas entre las acciones*: Los planes implican acciones que están en relación positiva entre ellas. Es entonces posible fusionar los planes para obtener un único plan, más o tan potente como los dos planes separados. El problema simplemente es sincronizar la ejecución de los planes parciales, de manera de resolver los conflictos por acceso a recursos, sin modificar los planes.
- *Relaciones negativas y de conflicto por acceso a recursos*: La resolución de este problema consiste en buscar una intersección no vacía de los planes, donde cada agente pueda satisfacer sus objetivos.

- *Planificación distribuida de un plan centralizado:* un grupo de agentes coopera para formar un plan centralizado. Típicamente, los agentes serán especializados en el diseño de ciertos aspectos del plan total, y contri-

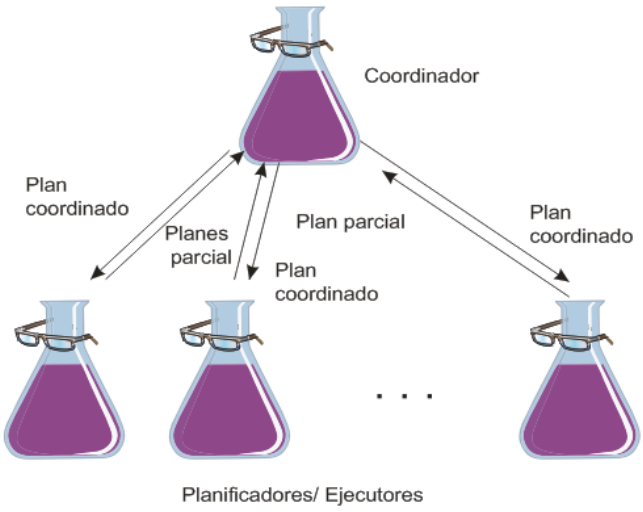

Figura 2.9: Coordinación centralizada para planes parciales

buirán en esa parte de él. Sin embargo, los agentes que forman el plan no necesariamente lo ejecutan, su papel es simplemente generar el plan.

- *Coordinación distribuida para planes parciales, o Planificación distribuida para planes distribuidos.* Un grupo de agentes posee planes de acción individuales, coordinando dinámicamente sus actividades, como se muestra en la Figura 2.10. Los agentes deben coordinarse, por lo que pueden necesitar mecanismos como los de negociación. No existe ningún agente centralizador, ni para planear planes globales, ni para coordinar planes parciales. En la planificación distribuida cada agente planea individualmente las acciones que piensa realizar en función de sus propios objetivos. La dificultad se refiere no sólo a resolver los conflictos potenciales que pueden presentarse en la ejecución de los planes (por ejemplo, por el uso de recursos), sino también, en reconocer las situaciones sinérgicas que se pueden presenter cuando las acciones de los unos pueden ser útiles a los otros. El problema consiste en cómo intercambiar información referente a sus planes y sus objetivos, para que cada uno pueda hacer los ajustes respectivos.
- *Jerarquización de los planes.* Se puede jerarquizar la ejecución de los planes, y suponer que la priorización entre ellos no conduce a conflictos. Si aparecen conflictos, se deberán tratar utilizando técnicas de resolución de conflictos.
- *Coordinación reactiva.* Ella considera que es a menudo más simple actuar directamente sin planear lo que se debe hacer. Para ello, toda la información para definir su comportamiento se encuentran en el medio ambiente, y sus reacciones dependen solamente de la percepción que ellos puedan

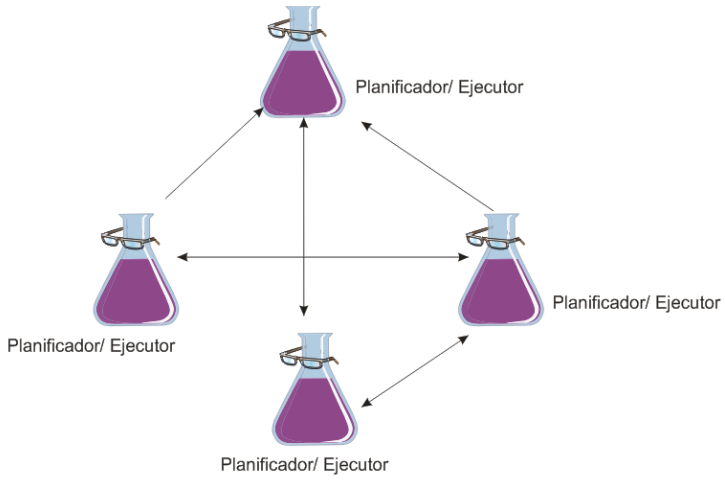

Figura 2.10: Planificación distribuida para planes distribuidos

tener. Estos agentes operan "aquí y ahora", sin memoria, y sin anticipar el futuro.
Casi todas las técnicas de coordinación reactivas se resumen a usar campos de potencias o marcas. Existen muchas técnicas en este ámbito, inspiradas en el comportamiento de las colonias de insectos. Básicamente, ellas se basan en:

- Mantener una distancia mínima entre los miembros de la colonia.
- Adaptar su comportamiento a la media de sus vecinos.
- Su comportamiento tenderá a ir hacia el centro de gravedad de sus vecinos.

Esas reglas son suficientes para que ese SMA adopte un comportamiento parecido al de, por ejemplo, el vuelo de un grupo de pájaros o el movimiento de bancos de peces [3, 15].

Referencias

1. H. Afsaneh. *Communication and Cooperation in Agent Systems: A Pragmatic Theory*. Springer Verlag, 1996.
2. J. Aguilar and L. Leiss. *Introducción a la Computación Paralela*. CDCHT-ULA, Gráficas Quintero, 2004.
3. E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence*. Oxford University Press, 1998.
4. J. Ferber. *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*. Addison Wesley, 1999.

5. S. Fernández, G. Pajares, and M. Santos. *Inteligencia Artificial e Ingeniería del Conocimiento*. Alfaomega, México, 2006.
6. Foundation for Intelligent Physical Agent. Specification. http://www.fipa.org, 2004.
7. A. Mas. *Agentes software y sistemas multiagente, Conceptos, arquitecturas y aplicaciones*. Pearson, 2004.
8. D. Mathijs and C. Brad. Introduction to Planning in Multiagent Systems. *Multiagent and Grid Systems Journal*, 5(4):345–355, 2009.
9. N. Nilsson. *Inteligencia Artificial: Nueva Síntesis*. McGraw-Hill, 1964.
10. G. OŕHare, N. Jennings, and N. (Editors) Jennings. *Foundations of Distributed Artificial Intelligence*. Wiley, 1996.
11. S. Russell and P. Norvig. *Inteligencia Artificial: Un Enfoque Moderno*. Prentice Hall Hispanoamericana, Madrid, 2 ed. edition, 2004.
12. I. Seilonen, T. Pirttioja, P. Appelqvist, A. Halme, and K. Koskinen. Distributed planning agents for intelligent process automation. In *IEEE InternationalSymposium on Computational Intelligence in Robotics and Automation*, pages 614 – 619, Finland, July 2003.
13. Y. Shoham and K. Leyton-Brown. *Multiagent Systems Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.
14. the Center for Integration Technology (CIT) Stanford University and the Knowledge Systems. Ontolingua. http://ksl.stanford.edu/software/ontolingua/, 2010.
15. Vidal, J. Fundamentals of multiagent systems with netlogo examples. http://www.damas.ift.ulaval.ca/c̄oursMAS/ComplementsH10/mas-Vidal.pdf, 2010.
16. N. Vlassis. A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence. In R. Brachman and T. Dietterich, editors, *Synthesis lectures on artificial intelligence and machine learning*. Morgan and Claypool Publishers, 2008.
17. G. Weiss. *Multi-agent System: a modern approach to distributed artificial intelligence*. MIT Press, 1999.
18. M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, second edition edition, 2009.

Capítulo 3
Metodología de Especificación de Agentes

Resumen: En este capítulo se detalla una metodología para la especificación de SMA en ambientes industriales denominada MASINA.

3.1. Introducción

Las metodologías para el desarrollo de 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 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 [20, 33, 34, 36, 52], la metodología *MultiAgent Systems for INtegrated Automation* (MASINA) [9] 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 [4, 5, 8, 10, 11, 15, 16, 19, 56]. 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 rápidos, precisos y eficientes; por otro lado, la *generación de conocimiento*, que debe ser incorporada 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 inteli-

gencia el proceso de inteligencia a nivel individual (mecanismos de aprendizaje, razonamiento, etc.) y colectivo (modelado de la inteligencia colectiva usando ontologías, aprendizaje reforzado mediado por el ambiente (por ejemplo: a través de feromonas, 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 características, para lo cual propone modelos de coordinación y comunicación precisos. También es posible representar aspectos como el uso de 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 consideraciones.

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 [41].

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 [14, 39].

A continuación se presenta una metodología que combina los aspectos considerados en MASINA [9], 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 metodología 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. En ella se resalta 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 permite especificar SMA de forma precisa, facilitando el proceso de implantación en ambientes reales complejos.

Esta metodología ha sido utilizada en los siguientes 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 [46].

3.2. Metodologías de especificación de agentes

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 [18, 22, 23, 30, 35, 49, 51, 53]; el segundo en metodologías provenientes de la ingeniería del conocimiento [27, 30, 31, 32, 48]; y el tercero, en el paradigma de agentes [13, 17, 21, 24, 42, 44, 54, 55]. En esta sección describiremos algunos aspectos relevantes de estas metodologías, en [28, 29] se presentan análisis exhaustivos de diferentes metodologías 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 [35], Burmeister [18] y Wood [53]. Kinny define una metodología para SMA que amplía OMT (Object Modelling Technique); por su lado, Burmeister describe una metodología para diseñar agentes, extendiendo metodologías orientadas a objetos, en la cual se distinguen tres modelos: modelo del agente, modelo de la 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 [27, 30, 31, 48], conocidas como CoMoMAS [27] y MAS-CommonKADS [32], 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 [21], Gaia [54, 55], HLIM [24], Prometheus [44], SODA [42] y Tropos [17]. 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 o estructura interna.

En [28, 29] 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 [34], y la complejidad de los ambientes de negocio y manufactura [36]. Por tanto, con el fin de integrar las múltiples perspectivas de los ambientes industriales (control, supervisión, mantenimiento, planificación, etc.), para alcanzar objetivos globales, diversos trabajos han sido orientados a la proposición de arquitecturas basadas en SMA [38, 43, 52]. Ahora bien, estos trabajos prestan más atención a los aspectos de comunicación.

Así pues, en vista de la potencialidad del uso de SMA en ambientes industriales, surge MASINA [9] a partir de los trabajos realizados en [5, 6, 7, 16, 33], 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 [26], 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 [9] y la técnica TD-SO. Se omite la descripción de UML por ser una herramienta ampliamente conocida [1, 25, 41, 45, 47, 50].

3.3. MASINA

La metodología MASINA [9] 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 3.1). 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.

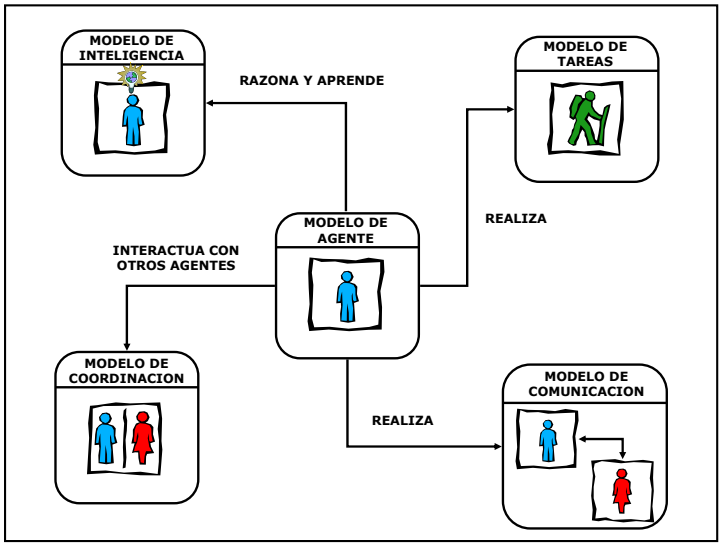

Figura 3.1: Modelos de MASINA para análisis.

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 existente.

En el *modelo de tareas*, MASINA agrega nuevos atributos, uno para especificar las tareas que requieren el uso de técnicas específicas (inteligentes, estadísticas, etc.), y otro para describir el macro-procedimiento (sub-tareas) que se debe seguir para la ejecución de dicha tarea. Por ejemplo, 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 3.2, 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 3.3). Estos actos de habla son detallados en el modelo de comunicación.

El modelo de coordinación de MASINA permite una descripción detallada de cada conversación, tal y como se muestra en la Figura 3.4. En este modelo se especifica el esquema de coordinación, de planificación, el mecanismo de comunicación directa o 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 amplia, y propone un modelo de comunicación que describe los actos de habla involucrados en las conversaciones entre los agentes del SMA, especificados en el modelo de coordinación.

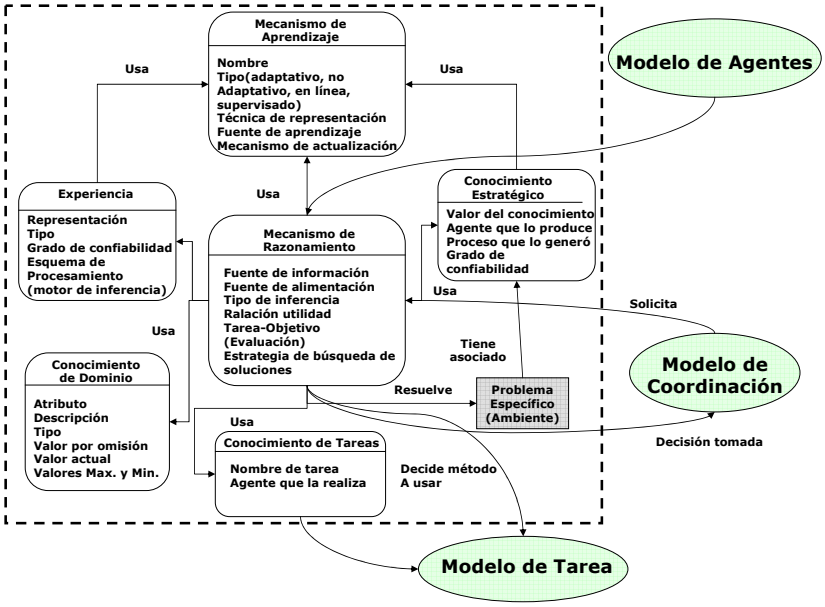

Figura 3.2: Modelo de inteligencia de MASINA

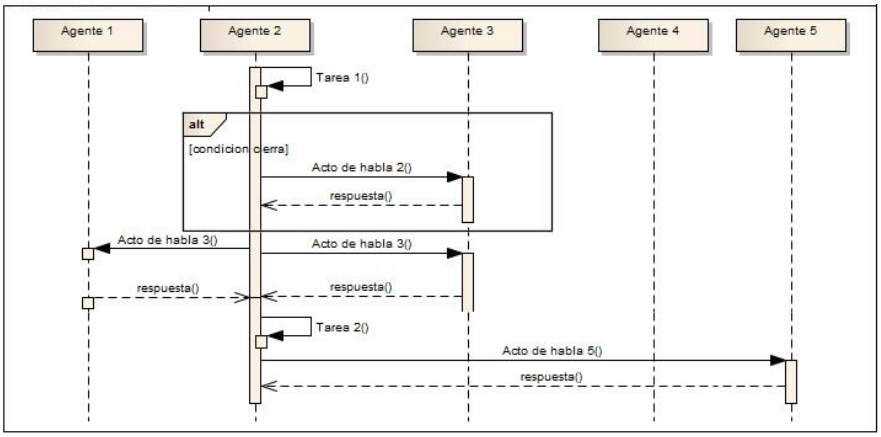

Figura 3.3: Actos de habla en MASINA

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.

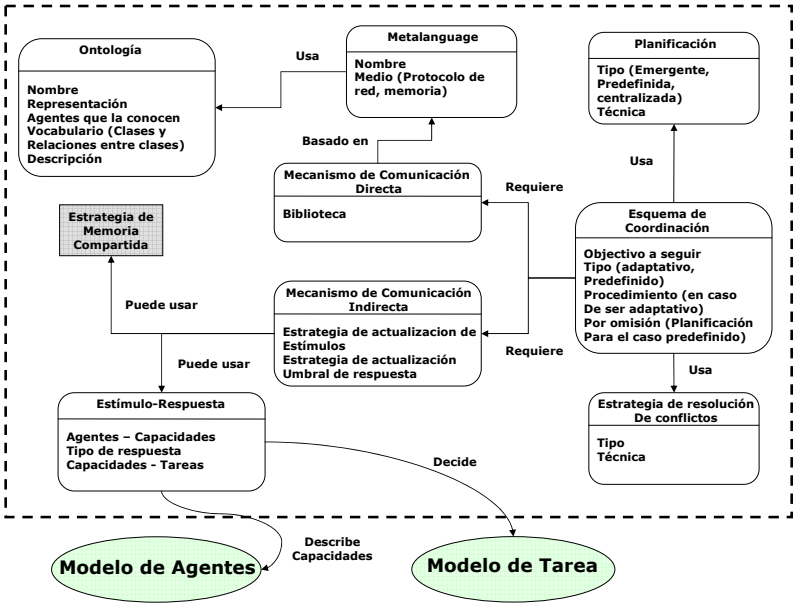

Figura 3.4: Modelo de coordinación de MASINA.

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 funcione 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 éstos con la plataforma real donde funcionarán.

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

La metodología MASINA ha sido usada para especificar modelos de referencia basados en SMA [4, 5, 8, 10, 11, 12, 19, 15, 20, 56] que permiten implementar tareas propias de ambientes distribuidos. En particular, ha sido de especial interés el desarrollo de modelos de referencia, basados en SMA, que permitan implementar tareas de automatización y control de procesos.

3.3.1. Técnica de Desarrollo de sistemas 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) [40], 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 [14] (OMT fue utilizada como base para el desarrollo de UML [41]). Del primero contiene todas las fases, incluyendo además la 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.

3.4. 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 [45], 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 [2], usando el formato presentado en [37], 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, notaciones y productos.

3.4.1. 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 éstos, y las relaciones que se establecen para cumplir los objetivos globales. La Tabla 3.1 muestra la plantilla usada para describir los casos de uso.

Caso de uso	Nombre del caso de uso
Descripción	Descripción detallada del caso de uso
Pre-condición	Condición 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
Condición de éxito	Elementos/eventos que indican el cumplimiento exitoso

Tabla 3.1: Plantilla de descripción de casos de uso.

3.4.1.1. Objetivo de la fase

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

3.4.1.2. 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.

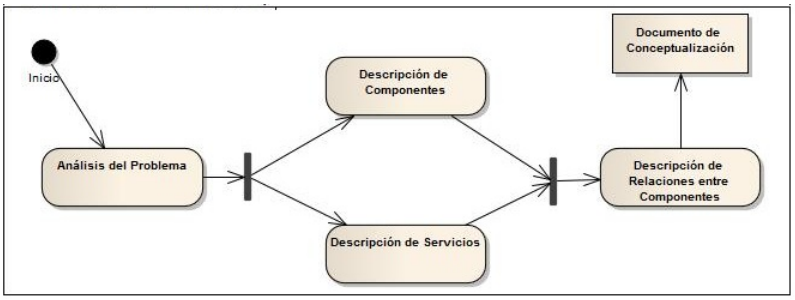

Figura 3.5: Flujo de trabajo en conceptualización.

3.4.1.3. Flujo de trabajo

En la Figura 3.5 se presenta el diagrama de actividades que muestra los pasos que se desarrollan en esta fase. La Tabla 3.2 contiene la descripción detallada de esas actividades, las técnicas a utilizar, y los productos obtenidos en cada paso.

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 3.1)	.- 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 3.2: Especificación del flujo de trabajo de la fase de conceptualización.

3.4.2. 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 en 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.

3.4.2.1. Objetivo de la fase

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

3.4.2.2. Producto principal

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

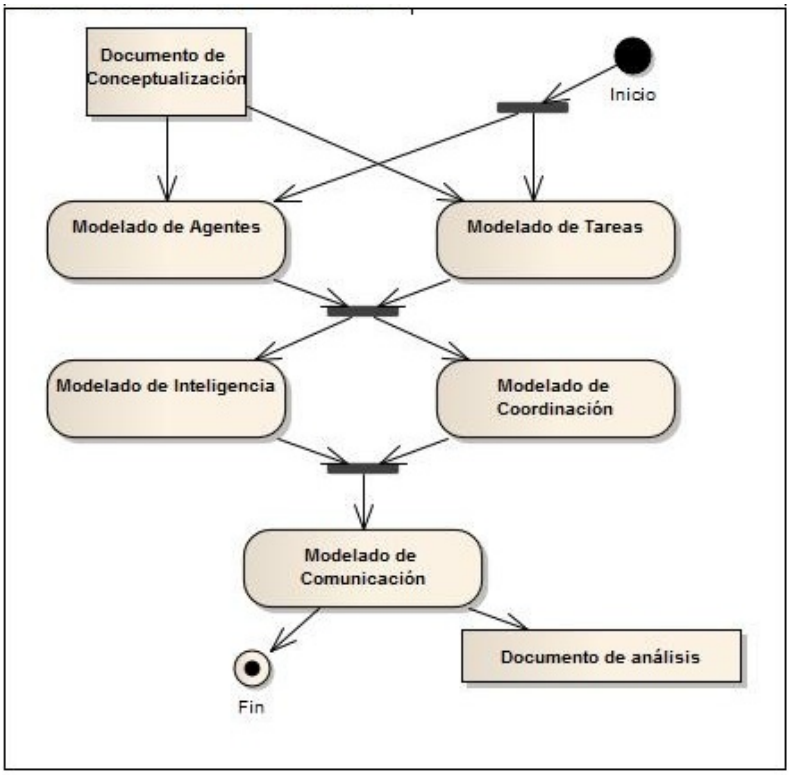

Figura 3.6: Flujo de trabajo de la fase de análisis.

3.4.2.3. Flujo de trabajo

En la Figura 3.6 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 3.3 contiene la descripción detallada de las actividades para la fase de análisis, las técnicas a utilizar y los productos obtenidos en cada paso.

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 3.4)	.- 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 3.6)	.- Tabla de relaciones Servicios-Tareas. .- Tabla de especificación de tareas
Modelado de inteligencia	.- 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 inteligencia (ver Figura 3.2)

Continua en la próxima página

Tabla 3.3 –Finaliza de la página anterior

Pasos	**Actividades**	**Técnicas y Notaciones**	**Productos**
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 3.3) .- Uso de plantillas para la especificación de las conversaciones (ver Tabla 3.7)	.- Modelado de diagramas de secuencia para cada conversación .- Tablas de especificación de conversaciones .- Diagrama de Coordinación (ver Figura 3.4)
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 3.8)	.- Tablas de especificación de actos de habla

Tabla 3.3: Especificación del flujo de trabajo de la fase de análisis.

La Tabla 3.4 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.

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

Continua en la próxima página

Tabla 3.4 –Finaliza de la página anterior

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		alor 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

Tabla 3.4: Plantilla del Modelo de Agente.

La Tabla 3.5 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 3.6).

Servicios	**Tareas**
Nombre Servicio 1	Nombre Tarea $S_1 - T_1$ ⋮ NombreTarea $S_1 - T_n$
Nombre Servicio 2	Nombre Tarea $S_2 - T_1$ ⋮
Continua en la próxima página	

Tabla 3.5 –Finaliza de la página anterior

	NombreTarea $S_2 - T_m$
Nombre Servicio L	Nombre Tarea $S_L - T_1$ ⋮ NombreTarea $S_L - T_k$

Tabla 3.5: Relación servicios-tareas del Agente.

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

Tabla 3.6: Plantilla del Modelo de Tareas.

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 3.7 y 3.8 muestran las plantillas para éstos.

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 3.7: Plantilla del Modelo de Conversación.

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

Tabla 3.8: Plantilla del Modelo de Comunicación.

3.4.3. 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.

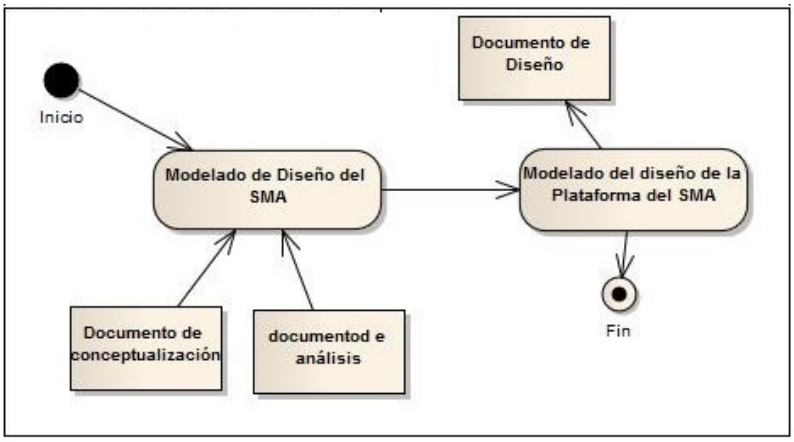

Figura 3.7: Flujo de trabajo de la fase de diseño.

3.4.3.1. Objetivo de la fase

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

3.4.3.2. 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.

3.4.3.3. Flujo de trabajo

En la Figura 3.7 se presenta el diagrama de actividades que muestra los pasos que se desarrollan en esta fase; estos pasos se detallan en la Tabla 3.9.

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 3.10, 3.11 y 3.12)	.- 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
Continua en la próxima página			

Tabla 3.9 –Finaliza de la página anterior

Pasos	**Actividades**	**Técnicas y Notaciones**	**Productos**
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 3.9: Especificación del 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 especificación del sistema se usa TDSO. En este caso, se genera el universo de clases del sistema, especificado mediante la plantilla que se muestra en la Tabla 3.10.

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

Tabla 3.10: Definición del universo de clases <nombreSistema>

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 3.11. Así mismo, cada uno de los métodos que componen la clase son especificados mediante la plantilla mostrada en la Tabla 3.12.

<fecha>		<Versión #>
# clase/TDA <nombreClase ([tipo_parámetros]) o nombreTDA> [: nombreClaseBase] {Comentarios sobre la clase}		
# 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.	.- Documentación de las declaraciones. .- Documentación de los atributos de la clase. .- Documentación de los métodos u operaciones definidas en la especificación sintáctica.
Continua en la próxima página		

Tabla 3.11 –Finaliza de la página anterior

	Especificación semántica: Presenta el escenario de cómo se deben utilizar los métodos u operaciones así como su comportamiento.	

Tabla 3.11: Definición formal de la clase <nombreClase o nombreTDA>

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

Tabla 3.12: Especificación formal del <nombreMétodo> o de la operación <nombreOperación>

3.4.4. 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 (para constatar 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).

3.4.4.1. 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.

3.4.4.2. Producto principal

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

3.4.4.3. Flujo de trabajo

En la Figura 3.8 se presenta el diagrama de actividades que muestra los pasos que se desarrollan en esta fase; estos pasos se detallan en la Tabla 3.13.

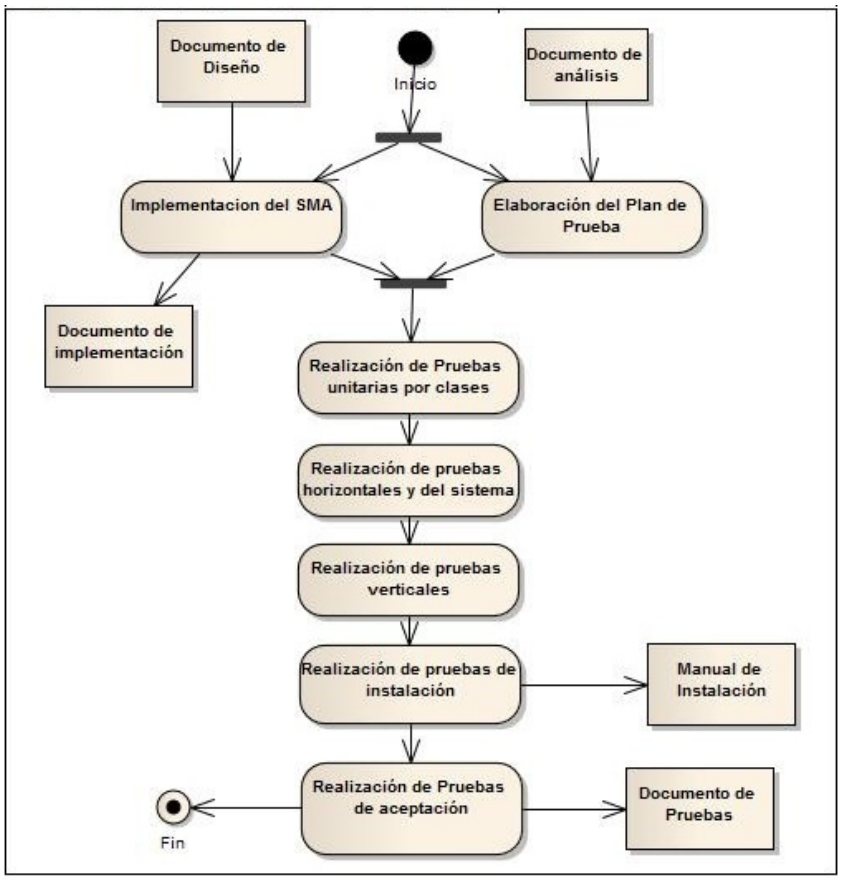

Figura 3.8: Flujo de trabajo de la fase de codificación y pruebas.

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

Continua en la próxima página

Tabla 3.13 –Finaliza de la página anterior

Pasos	Actividades	Técnicas y Notaciones	Productos
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

Tabla 3.13: Especificación del flujo de trabajo de la fase de Codificación y Pruebas

3.5. 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, sería de un gran aporte en los procesos de

automatización, para darle mayor versatilidad a los mismos (en [3], hay una comparación entre un sistema de automatización convencional y otro basado en agentes).

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 3.9).

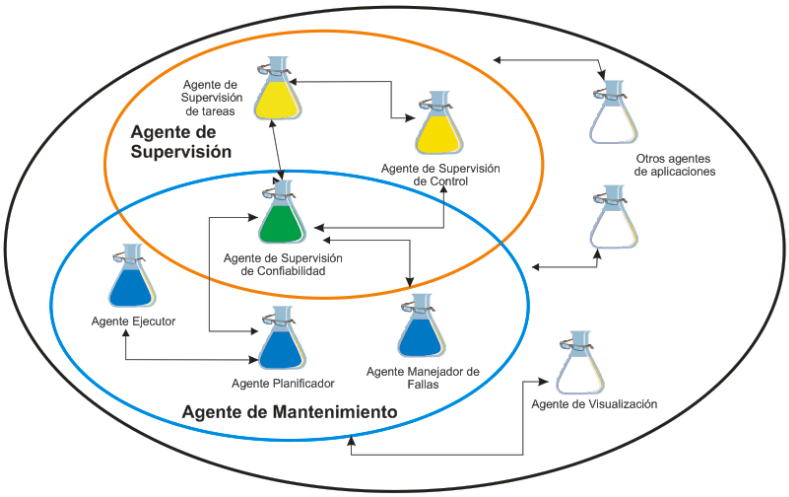

Figura 3.9: Ejemplo de un SMA para la automatización industrial.

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 específico, 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 el modelado del sistema de ingeniería orientado a agentes para este caso de estudio, que es la parte más innovadora de ella.

3.5.1. 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 configuraciones de usuarios, en función de sus roles, y desplegar los datos.

La Figura 3.10 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 3.14.

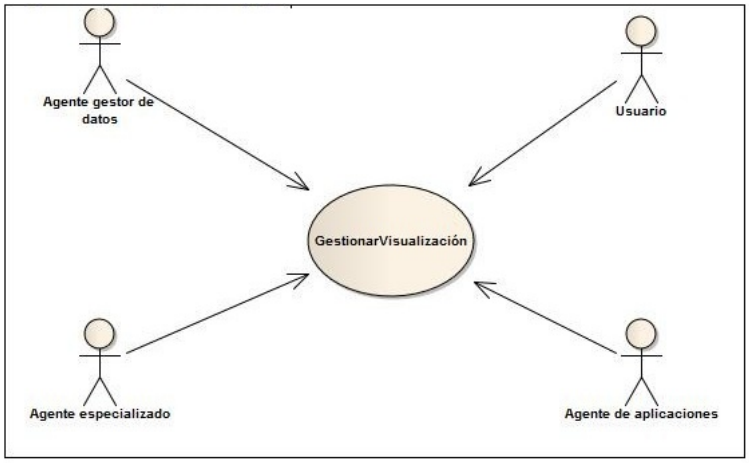

Figura 3.10: Diagrama de caso de uso para el AV

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.
Continua en la próxima página	

Tabla 3.14 –Finaliza de la página anterior

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.

Tabla 3.14: Descripción del caso de uso para el AV.

Para este agente se define un único servicio, llamado *Desplegar Datos*, y para ello realiza las siguientes actividades: consultar configuración, actualizar configuración, consultar perfil de usuario, actualizar perfil de usuario, y desplegar datos. La Figura 3.11 muestra el diagrama de actividades del AV.

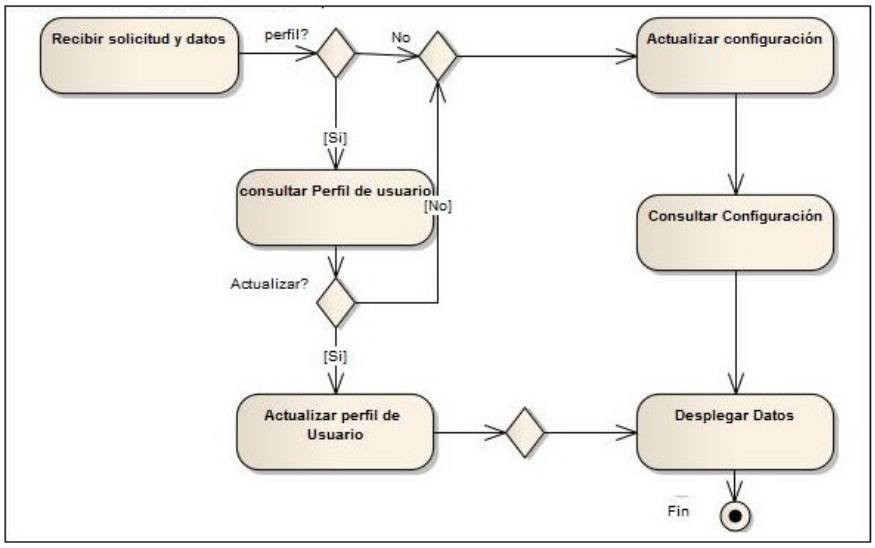

Figura 3.11: Diagrama de actividades para el AV.

3.5.2. Fase II. Análisis

Se elaboran los modelos que han sido propuestos en esta metodología.

3.5.2.1. Modelo de Agente

La Tabla 3.15 muestra un resumen del modelo de agente para el AV. Por razones de espacio esta tabla no contiene la descripción de cada uno de sus objetivos, pero si la descripción del único servicio que presta este agente.

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

Continua en la próxima página

Tabla 3.15 –Finaliza de la página anterior

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
Descripción 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		Acceso al almacenamiento a través de algún Gestor de Datos

Tabla 3.15: Modelo de Agente para el AV.

3.5.2.2. Modelo de tareas

En este caso, las tareas que ejecuta el AV se asocian al único servicio definido para él (ver Tabla 3.16). Una vez definidas las tareas, éstas se especifican en la plantilla del Modelo de Tareas indicada en la descripción de la fase de análisis en la sección anterior. La Tabla 3.17 muestra la especificación de la tarea **Realizar Consulta de Datos**.

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

Tabla 3.16: Relación servicio-tareas para el AV.

Cabe destacar que los ingredientes especificados en la Tabla 3.17 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.

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-Realizar consulta de datos	
DatoId	Identificador de los datos asociados a la consulta
UsuarioId	Identificador del usuario
Rango	Rango de visualización de los datos, si son numéricos

Tabla 3.17: Modelo de Tareas

3.5.2.3. Modelo de Coordinación

En función de los objetivos y servicio del AV, se define una conversación que permite todas las interacciones necesarias para el cumplimiento de los mismos. En la Tabla 3.18 se presenta la conversación definida para el AV, y en la Figura 3.12 se presenta el diagrama de interacción con los actos de habla de la conversación definida.

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.

Tabla 3.18: Modelo de Conversación

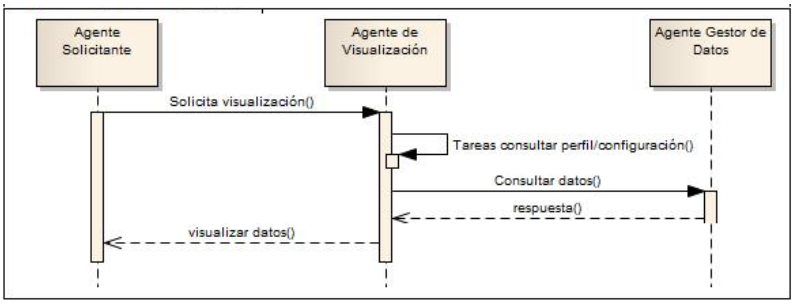

Figura 3.12: Diagrama de interacción del AV.

3.5.2.4. Modelo de Comunicación

A fin de ilustrar la especificación de los actos de habla, en la Tabla 3.19 se caracteriza uno de los actos de habla usado en la conversación definida en el modelo de coordinación.

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

Tabla 3.19: Modelo de Comunicación

3.5.3. 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 AV.

Las tablas 3.20 y 3.21 muestran la especificación básica y la especificación formal, respectivamente, para el universo de clases y tipos de datos abstractos de la clase AgenteVisualizacion.

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
Continua en la próxima página

Tabla 3.20 –Finaliza de la página anterior

+modificarPerfilAgente():Lógico +eliminarPerfilAgente(): Lógico +insertarConfiguracionAgente(): Lógico +consultarConfiguracionAgente(): Lista<Configuracion> +modificarConfiguracionAgente(): Lógico +eliminarConfiguracionAgente(): Lógico +destruirAgenteVisualizacion()

Tabla 3.20: Universo de clases del AV

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

Continua en la próxima página

Tabla 3.21 – Viene de la página anterior

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	

Continua en la próxima página

Tabla 3.21 –Finaliza de la página anterior

	x.desplegarDatosSuminstrados() → ban x.desplegarDatosConsultados → ban x.destruirAgenteVisualizacion()	

Tabla 3.21: Definición Formal de la clase AgenteVisualizacion

Para cada uno de los métodos se genera una especificación formal. Para ilustrar dicha especificación, la Tabla 3.22 muestra la descripción detallada del método *desplegarDatosSuministrados*.

10 de Mayo de 2007		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 3	crearAgenteVisualizacion **x** **x**.desplegarDatosSuministrados() → Cierto o Falso	Despliega los datos según un perfil y una configuración.

Tabla 3.22: Especificación formal del método desplegarDatosSuministrados

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, M. Cerrada, F. Hidrobo, and F. Narciso. Una Metodología para el Modelado de Sistemas de Ingeniería Orientado a Agentes. *Revista Iberoamericana de Inteligencia Artificial*, 12(38):39–60, 2008.
3. 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.

4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. J. Aguilar, F. Hidrobo, and M. Cerrada. A Methodology to Specify Multiagent Systems. *Lecture Notes in Artificial Intelligence*, 4496:92–101, 2007.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. S. A. DeLoach. Modeling Organizational Rules in the Multi-agent Systems Engineering Methodology. In *Canadian Conference on AI*, pages 1–15, 2002.

23. J. DiLeo, T. Jacobs, and S. A. DeLoach. Integrating Ontologies into Multiagent Systems Engineering. In *AOIS@CAiSE*, 2002.
24. 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.
25. M. Fowler and K. Scott. *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley, 2000.
26. A. Giret and V. Botti. Holons and agents. *Journal of Intelligent Manufacturing*, 15:645–659, 2004.
27. 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.
28. J. Gómez and J. Pavón. Methodologies for developing multi-agent systems. *Journal of Universal Computational Science*, 10(4):359–374, 2004.
29. B. Henderson-Sellers and P. Giorgini (eds.). *Agent-Oriented Methodologies*. Idea Group Publishing, 2005.
30. 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.
31. 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.
32. 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.
33. 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.
34. N. Jennings and S. Bussmann. Agent-based Control Systems. *IEEE Control Systems Magazine*, pages 61–73, 2003.
35. 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.
36. V. Marik and D. McFarlane. Industrial Adoption of Agent-Based Technologies. *IEEE Intelligent Systms*, pages 27–35, 2005.
37. 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.
38. 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.
39. 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.
40. OMG. Object Management Group. http://www.omg.org.
41. OMG. Unified Modeling Language Specification, 2003. Version 2.0, June 2003 via `http://www.uml.org`.
42. 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.

43. 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.
44. 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.
45. D. Pilone and N. Pitman. *UML 2.0 in a Nutshell*. O'REALLY, 2005.
46. A. Ríos, F. Hidrobo, J. Aguilar, and M. Cerrada. Control and Supervision Sytem Development with Intelligent Agents. *WSEAS Transactions on Systems*, 6(1):141–148, 2007.
47. M. Russ. *Learning UML 2.0*. Sebastopol O'Reilly Media, 2006.
48. 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.
49. 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.
50. 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.
51. 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.
52. 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.
53. 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.
54. 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.
55. F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Trans. Softw. Eng. Methodol.*, 12(3):317–370, 2003.
56. 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.

Parte II
Implantación de Sistemas Multiagentes

Preámbulo

Los derroteros que se han seguido en el desarrollo de este texto han sido, por un lado, la presentación del marco teórico del paradigma de agentes inteligentes, así como las metodologías y técnicas que permiten la creación de SMA para la solución informática y computacional de problemas de ingeniería. Naturalmente, ahora debemos presentar las plataformas de despliegue y desarrollo que se han venido realizando para este paradigma.

Específicamente, para desarrollar productos tecnológicos dentro del contexto de los sistemas informáticos, se siguen metodologías y técnicas establecidas por la Ingeniería de Software, lo que permite garantizar la funcionalidad del producto desarrollado, según lo especificado en los requerimientos. Ahora bien, la Ingeniería de Software en el paradigma multiagentes requiere considerar nuevas características para poder especificar, de manera precisa, el producto tecnológico basado en dicho paradigma. Algunas de esas nuevas características que se deben cumplir o satisfacer son: la reutilización de sus componentes, la interoperabilidad por los heterogéneos contextos de operación, la integración con otros sistemas, entre otros aspectos; de manera que estas soluciones tecnológicas tengan ciclos de vida importantes, sin obsolescencia. Estos aspectos han conducido al establecimiento de estándares y especificaciones para el desarrollo de soluciones tecnológicas basadas en los SMA. Además, ha conllevado al desarrollo de herramientas que apoyen los procesos de desarrollo de SMA, siguiendo las buenas prácticas establecidas en las metodologías y en los estándares, como su posterior despliegue. Así, alrededor del paradigma multiagentes existe un enorme número de propuestas de plataformas, tanto para el despliegue de SMA como para su desarrollo.

De esta manera, esta parte, novedosa en esta segunda edición, trata estos temas de manera exhaustiva, en particular, las plataformas para el despliegue o para el desarrollo de SMA. Por consiguiente, en el Capítulo 4 se presenta un estudio del estado del arte de las plataformas que permiten el desarrollo y el despliegue de aplicaciones sustentadas por el paradigma de agentes inteligentes, analizando los diferentes principios, enfoques y usos de dichas plataformas. Este estudio se inicia mediante un análisis de los estándares que se han venido desarrollando a nivel mundial para este paradigma, para después continuar con la presentación de las plataformas más comunes, algunas que permiten el despliegue de SMA, es decir dan la posibilidad de su ejecución en un contexto distribuido, otras que facilitan el desarrollo/implementación de los mismos (de los agentes, de sus capacidades cognitivas, de las conversaciones presentes en el SMA, etc.), o plataformas híbridas (que consideran ambos aspectos).

Seguidamente, en el Capítulo 5 se detallan los elementos a considerar para la construcción de un Medio de Gestión de Servicios, en base a los estándares que se analizan en el capítulo previo y a ciertos aspectos vincula-

dos a la automatización industrial. Por las características distribuidas de los SMA, la especificación precisa de los servicios a ofertar para permitir el despliegue de SMA es fundamental. Por ello, este capítulo se dedica a detallar las mismas, pero para un caso especial, para entornos de automatización. En general, un MGS se concibe para dotar los servicios fundamentales para el desempeño operacional de los agentes. Sobre la base de los sistemas distribuidos, el MGS es diseñado considerando requerimientos propios para el soporte a aplicaciones esparcidas, tales como mecanismos para la creación, el nombramiento, la localización, la búsqueda, la comunicación, entre otras características, de manera de garantizar un funcionamiento eficiente y seguro, sin afectar los objetivos propios de los SMA. Los principios de integración e interoperabilidad son fundamentales a considerar en este caso. Ahora bien, los contextos de automatización industrial generan específicos requerimientos vinculados a los tiempos de respuesta y a los comportamientos por defecto de los componentes computacionales. Para el primer aspecto, se hacen consideraciones para dar soporte a las restricciones de *tiempo real* de esos contextos industriales, considerandolo como un requerimiento de base. En ese sentido, el MGS que se presenta en el Capítulo 5 se aboca a proponer soluciones a dichas restricciones, para lo cual considera su vinculo directo al kernel de RTLinux, a través de un conjunto de servicios desarrollados en una biblioteca codificada en C^{++} del MGS, que permiten un manejo eficiente de los tiempos de respuesta de la plataforma computacional para los aspectos operativos y comunicacionales de los SMA (es decir, para su despliegue).

Finalmente, para posibilitar la implantación de un sistema especificado como una abstracción de SMA, se requiere de la construcción de un sistema que permita el desarrollo de los agentes y de las características distribuidas del SMA. En particular, considerando que en este libro se presenta una metodología para el desarrollo de SMA, los estándares del paradigma, y una plataforma para el despliegue de SMA en tiempos reales, parece lógico proponer una plataforma de desarrollo de SMA sobre las condiciones operacionales definidas por el MGS. Es así como el Capítulo 6 presenta una herramienta informática para facilitar la construcción de SMA siguiendo MASINA, sobre el MGS propuesto en el capitulo anterior. Básicamente, se presenta un **E**ntorno de **D**esarrollo **I**ntegrado para la construcción de **S**istemas **M**ulti**A**gentes (EDISMA). Este entorno se basa en la metodología MASINA, en los estándares para agentes inteligentes definidos por la FIPA, y en las características del MGS definido en el capítulo anterior. EDISMA permite garantizar uniformidad, consistencia, facilidad de integración y calidad de los componentes arquitectónicos del SMA bajo desarrollo, facilitando las fases de puesta en funcionamiento del sistema.

Esto permitirá dar el siguiente paso, sobre el uso del paradigma multiagentes como solución tecnológica a muchas de las dificultades que enfrenta la industria de producción a nivel de automatización integrada.

Capítulo 4
Estándares y plataformas: Estado del arte

Resumen: En este capítulo se presenta el estudio de varias plataformas para el desarrollo de agentes, para el despliegue de agentes, basadas en diferentes principios y con diferentes usos.

4.1. Introducción

Antes de realizar el estudio de las plataformas, es necesario evaluar el contexto o marco referencial a partir del cual se especifican las características funcionales de los SMA. Ese marco referencial lo establecen ciertos estándares. En el área de los agentes inteligentes han surgido organizaciones dedicadas a establecer estándares y definir plataformas.

La importancia de las plataformas y los estándares radica en el gran número de posibles usos que puede tener la teoría de agentes a nivel de aplicaciones computacionales, por lo que se requieren definir modelos, lenguajes y herramientas de implementación, entre otros aspectos, que faciliten el desarrollo e interoperabilidad de los SMA. Eso permitirá dar respuesta a aspectos relativos a: ¿Cómo usar los lenguajes y paradigmas de programación existentes en el diseño de SMA?, ¿Qué significa Programación Orientada a Agentes?, y además, ante el número plataformas que empiezan a encontrarse en Internet, para el desarrollo de SMA, decidir cuál usar entre ellas al responder a ciertas preguntas: ¿Cómo elegir? ¿Cómo compararlas?.

Desde varios puntos de vista, se puede establecer la importancia de las plataformas y de los estándares. Desde el punto de vista de los agentes, éstos requieren soporte para realizar las tareas especificadas en su diseño. Ellos deben poder:

- Sensar eventos, actuar o manipular el ambiente;
- Comunicarse;

- Tener nombres que sean únicos para poder ser identificados;
- Tener mecanismos de búsquedas;
- Poder cooperar, negociar y coordinarse.

La solución para brindar este soporte son plataformas de agentes, de SMA, o *Middleware* (medio de gestión de servicios) para SMA. También, desde el punto de vista de los diseñadores, la importancia radica en que ellos necesitan apoyo para desarrollar aplicaciones basadas en agentes. En particular, en cuatro etapas se requiere de apoyo:

- En el análisis del dominio del problema.
- En el diseño de la arquitectura para la solución del problema.
- En el desarrollo o construcción de una solución funcional para el problema.
- En la implementación de la solución para el problema (lanzamiento, operación y mantenimiento).

En este caso, la solución son herramientas IDE (*Integrated Development Environment*) para el desarrollo de SMA, herramientas para el despliegue de los SMA, etc. Desde el punto de vista del usuario final, ellos también requieren soporte para poder usar las aplicaciones, en particular, a nivel de:

- Seguridad: que sean confiables, que se garantice la integridad del sistema.
- Certificación de los servicios que proveen los agentes.

Particularmente, un Medio de Gestión de Servicios (MGS) es una capa de software que reside entre el *host* y el sistema operativo, por un lado, y la aplicación en el otro lado. El mismo es fundamental en entornos distribuidos, ya que provee un gran número de servicios: interfaces de programación, infraestructura de comunicación, define, en parte, el medio ambiente, etc. Como se podría deducir, para el desarrollo de SMA, un MGS es fundamental. Básicamente, un cliente de un MGS para agentes aspiraría que el mismo posea o permita:

- Mecanismos genéricos: a nivel de protocolos de cooperación, de comunicación, de negociación, de coordinación, entre otros; para la descripción de los agentes y de su organización.
- Capas de bajo nivel para los SMA: primitivas de comunicación entre agentes (KQML, ACL, entre otros), mecanismos que permitan a los agentes entrar (registrarse) y salir (desregistrarse) del SMA, duplicarse (crear otras instancias), migrar (moverse a otro nodo), etc.
- Motores que implementen el ciclo básico de comportamiento de los agentes, incluyendo lenguajes de ontologías, sistemas de razonamiento, etc.
- Acceso a los recursos disponibles: mecanismos de comunicación de bajo nivel (sockets, RPC), mecanismos de procesamiento distribuido (hebras), etc.
- Usar el concepto de agentes como una abstracción.

- Centrarse en los detalles del comportamiento de alto nivel del agente.
- Servicios de nombramiento, de transporte de mensajes y de páginas amarillas.

Esas son algunas de las razones por las que ha emergido una organización dedicada a establecer estándares para el desarrollo de sistemas basados en el paradigma de agentes, la cual ha sido identificada como FIPA (*Foundation for Intelligent Physical Agents*). Las mismas razones, han permitido la creación de un importante número de plataformas para el desarrollo y la gestión de SMA.

Este capítulo es organizado de la siguiente manera. Inicialmente se presenta una introducción a FIPA, los estándares que establece, sus razones y otros aspectos; para después presentar en detalle algunas de las plataformas y entornos más comunes y mejor documentados. Al final, brevemente, se describen otras plataformas.

Las plataformas a presentar fueron seleccionadas por alguna de las siguientes razones:

- Son populares y tienen un mantenimiento regular (corrección de errores, se le agregan características adicionales cada cierto tiempo, etc.).
- Siguen los estándares de FIPA.
- Cubren tantos aspectos como sean posible de los SMA: modelos de agente, de coordinación, de organización, etc.
- Poseen una buena documentación, se pueden descargar, el procedimiento de instalación es simple, etc.

4.2. Fundación para Agentes Físicos Inteligentes (FIPA)

La necesidad de tener estándares y normas en los SMA viene dado por las siguientes razones:

- Favorece el desarrollo de productos de software basados en el paradigma de agentes.
- Coadyuva a la definición de teorías, modelos, etc., alrededor de la teoría de agentes.
- Promueve las características deseadas de las aplicaciones basadas en agentes: abiertas, para ambientes heterogéneos y con propiedades emergentes.

FIPA es un organismo para el desarrollo y establecimiento de estándares de software para sistemas basados en agentes [1]. FIPA fue fundada como una asociación internacional sin fines de lucro en 1996, con el ambicioso objetivo de definir un conjunto completo de normas para la implementación de sistemas en los que se puedan ejecutar agentes (plataformas de agentes), de tal manera de producir especificaciones genéricas para tecnologías de

agentes. La organización suiza se disolvió en 2005 y un comité de estándares IEEE se creó en su lugar. Así, FIPA fue aceptada oficialmente por la Sociedad de Computación IEEE como parte de su familia de comités de estándares en ese año, para promover la tecnología basada en agentes y la interoperabilidad con otras tecnologías.

FIPA está estructurada en comités técnicos, grupos de trabajo, etc. Existe una fuerte participación de las empresas de telecomunicaciones, pero también de otras empresas: BT, EPFL, France Télécom, Fujitsu, HP, Hitachi, IBM, Imperial College, Intel, Motorola, NASA, Nec, NHK, Nortel Networks, NTT, Philips, Siemens, SNCF, SUN Microsystems, Telecom Italia, entre otras.

En general, FIPA produce dos tipos de especificaciones: *normativas*, que definen el comportamiento externo de un agente y garantizan la interoperabilidad con otros subsistemas especificados por FIPA; e *informativas*, que proporcionan una orientación en el uso de las tecnologías FIPA.

FIPA busca establecer o garantizar, entre otras cosas:

- La definición de una arquitectura genérica para soportar SMA.
- La interoperabilidad del transporte de mensajes entre agentes.
- Las diversas formas del lenguaje de contenidos de los agentes, a ser usados en sus comunicaciones (*Agent Communication Language*, ACL).

El conjunto completo de especificaciones que propone FIPA se clasifican en las siguientes categorías (ver Figura 4.1): para la comunicación entre agentes, para el transporte de mensajes, para la gestión de agentes, una arquitectura genérica para el despliegue de agentes, y algunas aplicaciones comunes basadas en agentes.

Los primeros documentos de FIPA indicaban las normas que permitían a una sociedad de agentes existir, funcionar y ser administrada. En primer lugar se describió el modelo de referencia de una plataforma de agentes, identificando los roles de algunos agentes claves necesarios para la gestión de la plataforma, y los lenguajes para el manejo de los contenidos de los mensajes entre los agentes. Tres funciones obligatorias se identificaron en una plataforma de agentes:

- *El Sistema de Gestión de agentes (AMS)* es el agente que ejerce el control y supervisión sobre el acceso y uso de la plataforma. Es responsable de mantener un directorio de los agentes residentes y de sus ciclos de vida.
- *El canal de comunicación de los agentes (ACC)* proporciona la ruta de acceso para el contacto entre los agentes dentro y fuera de la plataforma. El ACC es el método de comunicación predeterminado, que ofrece un servicio de enrutamiento confiable y ordenado.
- *El Directorio Facilitador (DF)* es el agente que presta servicios de páginas amarillas a la plataforma de agentes.

Como ya se dijo, las especificaciones también definen el Lenguaje de Comunicación de Agentes (ACL), que deben utilizar los agentes para el inter-

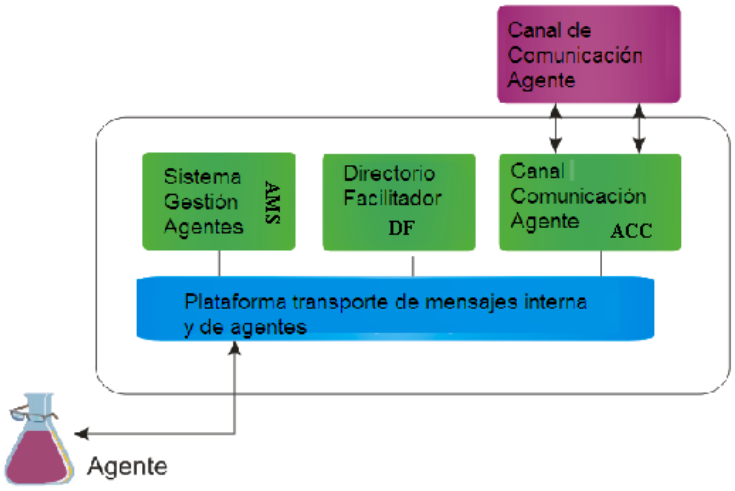

Figura 4.1: Modelo de Referencia para Agentes según FIPA.

cambio de mensajes. FIPA ACL es un lenguaje de codificación de mensajes con semántica, pero no define los mecanismos específicos para el transporte de mensajes. La sintaxis de ACL se basa en el lenguaje de comunicación KQML. Sin embargo, existen diferencias fundamentales entre KQML y ACL, siendo la más evidente la existencia de una semántica formal para FIPA ACL, que elimina cualquier ambigüedad al utilizar el lenguaje. En particular, para los problemas de comunicaciones entre agentes se han definido las especificaciones: FIPA ACL *Message Structure Specification*, FIPA *Ontology Service Specification* y FIPA SL *Content Language Specification*.

Por otro lado, FIPA también define las formas más comunes de las conversaciones entre los agentes a través de protocolos de interacción, que son los patrones de comunicación seguidos por dos o más agentes. Esta gama de protocolos va desde los simples, como los de consulta y de solicitud, hasta los más complejos, como los de contratación (conocidos como protocolos de negociación, tales como los de subasta inglesa u holandesa). Alguno de ellos son: FIPA *Request Interaction Protocol Specification*, FIPA *Query Interaction Protocol Specification*, FIPA *Request When Interaction Protocol Specification*, FIPA *Contract Net Interaction Protocol Specification*, FIPA *Brokering Interaction Protocol Specification*, FIPA *Recruiting Interaction Protocol Specification*, FIPA *Subscribe Interaction Protocol Specification*.

También se han elaborado especificaciones para la gestión y control de los agentes dentro de plataformas de agentes, como también especificaciones a nivel del transporte y para la representación de mensajes para diferentes protocolos de transporte de red, incluyendo entornos de telefonía fija y móvil: FIPA *Agent Message Transport Service Specification*, FIPA *Messaging*

Interoperability Service Specification, etc. Como también ya se dijo, FIPA ha especificado algunos ejemplos de aplicaciones de agentes (agentes FIPA), con su ontologías y servicios ofrecidos. Algunos ejemplos de agentes FIPA son: FIPA *Personal Travel Assistance Specification* y FIPA *Personal Assistant Specification*. También ha especificado algunos aspectos genéricos como: FIPA *Agent Software Integration Specification*, FIPA *Quality of Service Specification* y FIPA *Network Management and Provisioning Specification*. Más recientemente, otros estándares se han venido definiendo en las especificaciones FIPA, en particular, para la integración de agentes, la movilidad de agentes, la seguridad de agentes, los servicios de ontología, y la comunicación con el agente humano.

Algunos sistemas computacionales basados en FIPA son:

- JADE
- JIAC
- JACK
- Zeus
- Fipa-OS

Finalmente, existen otros estándares de interés para FIPA, como los esfuerzos de la OMG (*Object Management Group*) para crear estándares para objetos móviles [2], o los vinculados a servicios de Ontologías [3, 4]: DAML, OIL, OWL, etc.

4.3. Ambientes híbridos

A continuación se presentan las plataformas más comunes actualmente existentes para el desarrollo y/o el despliegue de agentes (MGS para SMA). Se comienza, en está sección, con las plataformas que soportan ambas tareas, es decir son herramientas para desarrollar SMA, y además soportan el despliegue/ejecución de las mismas.

4.3.1. JADE

JADE (*Java Agent DEvelopment Framework*) es un software para desarrollar aplicaciones basadas en agentes que sigue las especificaciones de FIPA, para garantizar SMA interoperables [33, 5, 68]. El objetivo de JADE es simplificar el desarrollo de agentes, garantizando seguir los estándares establecidos por la FIPA, a través de un conjunto de servicios ofrecidos por la plataforma. JADE puede ser considerado como un *middleware* de agentes, que implementa una plataforma para la gestión de los agentes y un IDE (*Integrated Development Environment*) para el desarrollo de agentes. El *middleware*

de JADE cumple con las especificaciones FIPA, y a través de un conjunto de herramientas gráficas soporta las fases de implementación y depuración de agentes. Se ocupa de todos aquellos aspectos que no son propios a un agente, que son independientes de las aplicaciones donde vayan a ser usados. En particular, incluye:

- Una plataforma de agentes compatible con FIPA.
- Un paquete basado en Java para desarrollar agentes.
- Varias extensiones para ejecutarse en dispositivos con aplicaciones específicas (PDA, teléfonos inteligentes, etc.).

JADE ha sido totalmente implementado en el lenguaje Java. Además, las extensiones recientes de JADE son compatibles con los entornos móviles de Java (J2ME CLDC MIDP 1.0 [38, 6, 68]). JADE es un software libre distribuido por Telecom Italia, bajo los términos de la LGPL (*Lesser General Public License*) .

La plataforma de agentes se puede distribuir en varios *hosts*. Es suficiente una aplicación Java, y por lo tanto una Máquina Virtual Java (JVM) se ejecuta en cada host. Cada JVM es básicamente un contenedor de agentes que proporciona un completo entorno de ejecución para éstos, y permite que múltiples agentes se ejecuten simultáneamente en el mismo host. Se trata de una plataforma de agentes distribuida (ni siquiera deben compartir el mismo sistema operativo), que tiene un contenedor por cada host en el que se están ejecutando agentes, y la configuración se puede controlar a través de una interfaz gráfica de usuario remota. La configuración puede cambiarse en tiempo de ejecución, moviendo los agentes de una host a otro según las necesidades. De esta manera, JADE crea múltiples contenedores destinados a los agentes, cada uno de los cuales puede ejecutarse en uno o en varios sistemas.

La plataforma JADE soporta la coordinación de múltiples agentes FIPA y proporciona una implementación estándar del lenguaje de comunicación FIPA-ACL, que facilita la comunicación entre agentes y permite la detección de servicios que se proporcionan en el sistema. JADE ha participado con éxito en las pruebas de interoperabilidad de FIPA, y ha sido ampliamente utilizado en varios proyectos.

Para la implantación de SMA, JADE proporciona:

- Un entorno de ejecución en el que los agentes se ejecutan.
- Bibliotecas de clases para la creación de agentes mediante la herencia y la redefinición de comportamientos.
- Un conjunto de herramientas gráficas para el monitoreo y la administración de la plataforma.

En particular, JADE cumple con las siguientes especificaciones de FIPA: páginas de servicio amarillas, transporte de mensajes, servicio de análisis de nombres, y una biblioteca de protocolos de interacción FIPA lista para

ser usada. La plataforma de los agentes JADE incluye todos los componentes obligatorios para gestionar una plataforma de agentes según FIPA: el ACC, el AMS y el DF. Además, todas las comunicaciones de los agentes se realizan a través de pase de mensajes, donde FIPA ACL es el lenguaje para representar los mensajes. De esta manera, el modelo completo de comunicación FIPA está implementado, y sus componentes han sido totalmente integrados: protocolos de interacción, lenguajes de contenido ACL, sistemas de codificación, el uso de ontologías, protocolos de transporte, etc. El mecanismo de transporte, en particular, es como un camaleón, ya que se adapta a cada situación: de forma transparente hace la elección del mejor protocolo disponible. Java RMI, notificación de eventos, HTTP e IIOP, se utilizan actualmente, pero más protocolos pueden ser fácilmente añadidos.

La plataforma JADE posee diferentes herramientas que permiten, entre otras cosas, la depuración de los agentes, la movilidad del código y del contenido de los agentes, la posibilidad de ejecución paralela del comportamiento de los agentes, y la definición del lenguaje de comunicación y de las ontologías a usar por los agentes. Por otro lado, siguiendo la especificación FIPA, JADE tiene un contenedor principal que tiene dos agentes especiales:

- *DF (Directory Facilitator):* proporciona un directorio que anuncia qué agentes están disponibles en la plataforma. Para acceder al agente DF se usa la clase "jade.domain.DFServicez sus métodos: *register()*, *deregistrer()*, *modify()* y *search()*.
- *AMS (Agent Management System):* es el único que puede crear y destruir a otros agentes, destruir contenedores y detener la plataforma. Para acceder a los servicios del AMS, cuando un agente es creado, éste ejecuta automáticamente su método *register()*. Cuando se destruye un agente se debe ejecutar el método *takedown()*, y automáticamente se llama al servicio *deregister()* del AMS.

JADE define la Clase *Agent*, la cual es una superclase que permite a los usuarios crear agentes JADE. Así, toda nueva clase de agente hereda de la clase *Agent()*. Esta clase suministra métodos que permiten ejecutar las tareas básicas de los agentes como:

- Pasar mensajes utilizando objetos ACLMessage (Agent Communication Language).
- Dar soporte al ciclo de vida de un agente.
- Planificar y ejecutar múltiples actividades al mismo tiempo.

El ciclo de vida de un agente JADE sigue la propuesta FIPA. Estos agentes pasan por diferentes estados:

1. Inicial: el agente se ha creado pero no se ha registrado todavía en el AMS.
2. Activo: el agente ya ha sido registrado y posee nombre. En este estado puede comunicarse con otros agentes.

3. Suspendido: el agente se encuentra parado porque su hilo de ejecución se encuentra suspendido.
4. Esperando: se encuentra bloqueado a la espera de un suceso.
5. Eliminado: el agente ha terminado, por tanto el hilo terminó su ejecución y ya no está más en el AMS.
6. Tránsito: el agente está migrando a una nueva ubicación.

El "comportamiento" de un agente define las acciones a realizar bajo un determinado evento. Los diferentes comportamientos que el agente adopta se definen a partir de la clase abstracta *Behaviour*, para lo cual usa el método *addBehaviour*. La clase *Behaviour* contiene los métodos abstractos: i) *action()*: Se ejecuta cuando la acción tiene lugar; ii) *done()*: Se ejecuta al finalizar el comportamiento.

Por otra parte, el usuario puede redefinir los métodos *onStart()* y *onEnd()* que el agente posee (ejecutados al inicio o al final del funcionamiento del agente). Adicionalmente, existen otros métodos como *block()* y *restart()*, usados para la modificación del comportamiento del agente. Cuando un agente esté bloqueado se puede desbloquear de diferentes maneras:

1. Recepción de un mensaje.
2. Cuando haya transcurrido el tiempo asociado a *block()*.
3. Por una llamada a *restart()*.

Como se dijo antes, el pase de mensajes ACL es la base de la comunicación entre los agentes. El envío de mensajes se realiza mediante el método *send()* de la clase *Agent*. A dicho método hay que pasarle un objeto de tipo *ACLMessage*, que contiene la información de los destinatarios, lenguaje usado en la codificación del mensaje, y el contenido del mensaje. Estos mensajes se envían de modo asíncrono, tal que los mensajes que se van recibiendo se irán almacenando en una cola de mensajes. Existen dos tipos de recepción de los mensajes ACL: bloqueante y no bloqueante. Para ello se proporcionan los métodos *blockingReceive()* y *receive()*, respectivamente. En ambos métodos se puede hacer filtrado de los mensajes que se quieren recuperar de la cola. De esta manera, la arquitectura de comunicación ofrece una mensajería flexible y eficiente, donde JADE crea y gestiona una cola de mensajes ACL entrantes privados, para cada agente. Los agentes pueden acceder a su cola a través de una combinación de varios modos: bloqueo, votación, tiempo de espera y basado en un patrón.

Básicamente, los agentes se implementan como un hilo por agente, pero los agentes a menudo necesitan ejecutarse en paralelo. Así, además de la solución multi-hilo, ofrecidos directamente por el lenguaje Java, JADE también admite la programación de comportamientos cooperativos donde se planifican las tareas sobre él. JADE incluye prototipos de comportamientos listos para utilizarse, de las tareas más comunes en la programación de agente, tales como los protocolos de interacción FIPA, despertar bajo una cierta condición, etc. En el caso de las ontologías, un sistema de gestión de

ontología de agentes ha sido desarrollado, así como el soporte a lenguajes y ontologías que pueden ser implementadas y registradas con los agentes. JADE ofrece una llamada a *JessBehaviour* que permite la integración con JESS [49, 7], un entorno para la programación de reglas con su motor de razonamiento, donde JADE proporciona la cáscara del agente y garantías (cuando sea posible) del cumplimiento de FIPA, mientras que JESS realiza el razonamiento (se puede integrar con Protege [8], una plataforma de gestión de ontologías actualmente muy utilizada).

JADE tiene una extensión, denominada WADE (*Workflows and Agents Development Environment*), que es un sistema de *workflow* que permite crear procesos mediante un editor gráfico llamado *WOLF*. WADE es una plataforma independiente de dominio, construido encima de JADE. WADE permite la definición de componentes como flujos de trabajo que pueden ser fácilmente reutilizados, que aparecen directamente en la opción Editor. Algunas de las cosas que permite WADE son:

- Una consola Web de administración, que admite la autenticación de usuarios y perfiles,
- Un módulo llamado *WadeInterface*, que permite el lanzamiento y el control de los flujos de trabajo a través de su interfaz

El desarrollo de JADE todavía continúa. Ya se han previsto nuevas mejoras e implementaciones, la mayoría de ellas en colaboración con los usuarios de la comunidad interesados. Además, actualmente existe una buena cantidad de aplicaciones y productos de desarrollo que se han realizado alrededor de JADE, que muestran las fortalezas y las oportunidades abiertas por la plataforma. La concepción de "código abierto" de JADE lo ha hecho posible.

4.3.1.1. Características de JADE

En cuanto al contenedor, cada agente vive en un contenedor. Un contenedor es una maquina virtual de Java, el cual provee un completo ambiente de ejecución para los agentes, permitiendo que varios de ellos puedan ejecutarse concurrentemente, controlando sus ciclos de vida (crearlos, suspenderlos, eliminarlos, etc.) y comunicaciones (envío y enrutamiento de mensajes). Por otro lado, existe un contenedor especial, llamado *front-end* (FE), que ejecuta los agentes de administración y representa a toda la plataforma ante el mundo exterior; y otro contenedor especial para mostrar la interfaz Web (ver Figura 4.2, extraída de [5]). Este contenedor ejecuta un agente llamado *Remote Management Agent (RMA)* que posee su propia Interfaz Gráfica de Usuario (IGU).

El contenedor FE es el que provee todos los servicios FIPA para darle soporte a la plataforma de agentes provista por JADE (ver Figura 4.3, extraída de [5]). Además de los componentes básicos previstos por FIPA (AMS y DF), posee una tabla descriptora de los contenedores actualmente en el sistema.

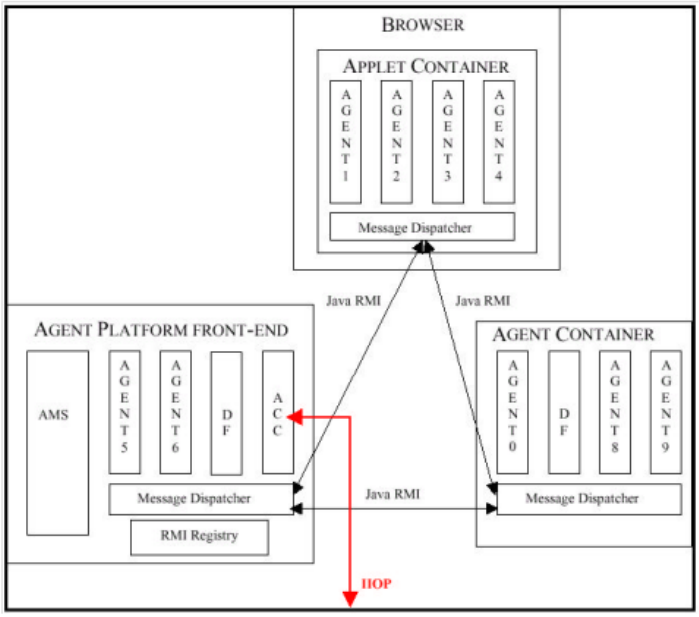

Figura 4.2: Contenedores en JADE

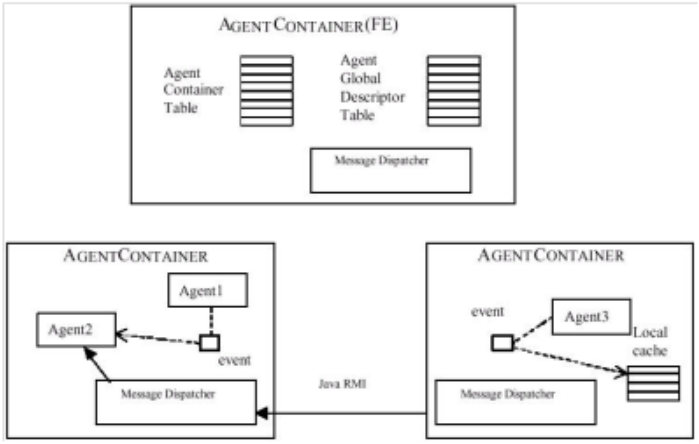

Figura 4.3: Contenedor FE de JADE

JADE, como se dijo antes, además de una biblioteca, ofrece herramientas para gestionar la plataforma de agentes en ejecución, y controlar y depurar la sociedad de agentes. Todas estas herramientas se ejecutan como agentes propios de JADE.

La IGU para la administración, seguimiento y control del estado de los agentes y de la plataforma, permite detener y reiniciar los agentes; crear e iniciar la ejecución de un agente en un *host* remoto, a condición de que un contenedor de agentes ya se esté ejecutando. La IGU permite, además, controlar otras plataformas remotas de agentes FIPA compatibles. El RMA adquiere la información acerca de la plataforma, permitiendo la ejecución de los comandos para modificar el estado de ésta (crear agentes, cerrar contenedores, etc.), a través del AMS.

El agente gestor del directorio (DF) también tiene una IGU, a través de la cual se puede administrar y configurar los agentes y servicios anunciados. Mediante el uso de esta interfaz gráfica, el usuario puede ver las descripciones de los agentes registrados, registrar y dar de baja agentes, modificar la descripción de un agente registrado, buscar descripciones de agentes, entre otras cosas. La IGU también permite federar el DF con otros DF, y crear una compleja red de dominios y subdominios de páginas amarillas. Cualquier DF federado, incluso si reside en una plataforma de agentes no-JADE remota, puede ser controlado por la misma IGU, tal que las mismas operaciones básicas se pueden ejecutar sobre el DF remoto.

Una serie de herramientas gráficas se han desarrollado para apoyar la fase de depuración de los agentes, por lo general, una tarea compleja en sistemas distribuidos. Los usuarios JADE pueden depurar sus agentes con el agente *dummy* (simulado) y el agente de *sniffer*. El agente *dummy* es una herramienta para la inspección de los intercambios de mensajes entre los agentes. Este agente facilita la prueba y validación de la interfaz de los agentes antes de su integración en un SMA.

El agente *sniffer*, a través de su IGU, permite el seguimiento de los mensajes intercambiados en una plataforma de agentes JADE. Cuando el usuario decide rastrear un agente o un grupo de agentes, cada mensaje dirigido a, o procedente de, el agente o grupo de agentes, se sigue y se muestra en la ventana del *sniffer*, utilizando una notación similar a los diagramas de secuencia de UML.

Por otro lado, el agente *introspector* permite monitorizar y controlar el ciclo de vida de un agente en ejecución, y sus mensajes intercambiados, en particular, su cola de mensajes enviados y recibidos.

Desde el punto de vista del diseño, JADE propone un estilo propio para dar soporte a características fundamentales de los agentes, como la autonomía y la sociabilidad. Además, incorpora capacidades para implementar los actos de habla [8]. Así, JADE propone las propiedades de diseño de agentes siguientes:

- Los agentes son autónomos, entonces son objetos activos, con al menos un hilo de Java, para iniciar de manera proactiva nuevas conversaciones, hacer planes y alcanzar metas.
- Los agentes son sociales, por lo que se permite la concurrencia intra-agentes. Eso tiene como resultado permitir que un agente pueda participar en muchas conversaciones al mismo tiempo.

- Los mensajes son actos de habla, por lo que se utiliza la mensajería asíncrona. Es la mejor manera de intercambiar información entre dos agentes independientes.
- Los agentes pueden decir "no", entonces es necesario un modelo de comunicación peer-to-peer. En ese caso, en un SMA son iguales el remitente y el receptor (en contraposición a los sistemas cliente/servidor en el que se supone que el receptor obedece al remitente). Un agente autónomo también debe poder ignorar un mensaje recibido, siempre y cuando lo desee.

Las consideraciones anteriores ayudan a decidir el número de hilos necesarios en una implementación de agentes: el requisito de autonomía obliga a que cada agente tenga al menos un hilo, y el requisito de sociabilidad conduce a la existencia de muchos hilos por agente.

El modelo de ejecución JADE trata de limitar el número de hilos a través de la abstracción *comportamiento*: una colección de comportamientos están programados y son ejecutados para llevar a cabo las funciones del agente. En el caso de un agente JADE, éste ejecuta en su propio hilo Java todos sus comportamientos. Así, JADE utiliza un modelo de ejecución de hilo-por-agente con la programación cooperativa intra-agente de comportamientos.

En particular, los agentes JADE programan sus comportamientos como una *planificación cooperativa en una pila*, en el que todos los comportamientos se ejecutan desde el tope de una pila única (desde la parte superior de la pila), y un comportamiento se ejecuta hasta que termine, sin poder ser sacado por otros comportamientos (programación cooperativa). La elección de no guardar el contexto de ejecución de un comportamiento significa que los comportamientos de agentes empiezan desde el principio cada vez que son ejecutados. Por lo tanto, cuando un estado de un comportamiento debe conservarse a través de múltiples ejecuciones, se debe almacenar en variables de instancias de comportamiento. El modelo de JADE es un esfuerzo para proporcionar un paralelismo de grano fino en un hardware de grano grueso. Una regla general para la transformación de un método ordinario Java en un comportamiento JADE es:

- Convertir el cuerpo del método en un objeto cuya clase hereda de Comportamiento.
- Convertir las variables locales del método en las variables de instancias de Comportamiento.
- Agregar el objeto Comportamiento a la lista de comportamientos del agente.

Las directrices anteriores permiten aplicar la técnica de reificación: los objetos comportamientos de los agentes reifican un método y un hilo separado al ejecutarse. Por otro lado, los agentes JADE poseen una única cola de mensajes de los que se recuperan los mensajes ACL.

En general, el desarrollo de un agente en JADE implica extender la superclase *Agent* y ejecutar las tareas específicas del agente escritas como subclases de la clase comportamiento. En particular, los agentes definidos por el usuario heredan de la superclase la capacidad de registrarse y darse de baja de la plataforma, y un conjunto de métodos básicos (por ejemplo, enviar y recibir mensajes de ACL, protocolos de interacción estándar, etc.). También heredan de la superclase dos métodos: *addBehaviour()* y *removeBehaviour()*, para administrar la lista de comportamientos del agente. La Figura 4.4 (extraída de [5]), muestra el diagrama de clases con la sus descripciones básicas.

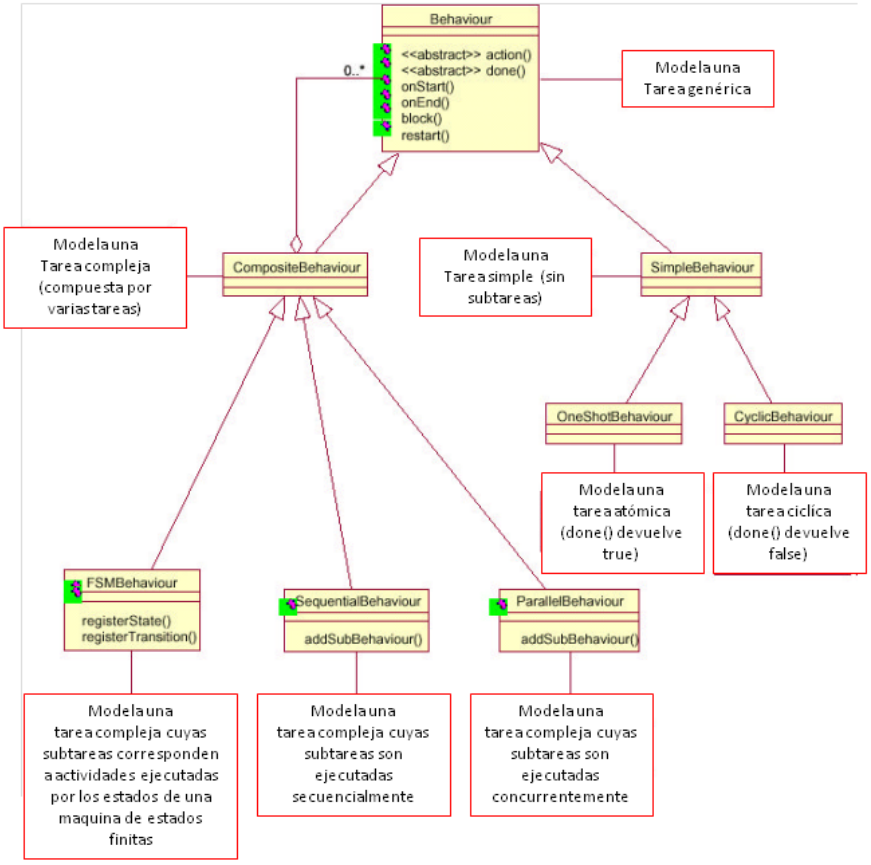

Figura 4.4: La abstracción *comportamiento* de JADE

El comportamiento es una clase abstracta que proporciona el esqueleto de una tarea primaria a realizar. Posee tres métodos: el método *action()*, que representa la tarea a ser llevada a cabo por la clase comportamiento; el método *done()*, utilizado por el planificador del agente, que debe devolver

true si el comportamiento ha terminado y *false* en el caso contrario (en ese caso el método *action()* debe ser ejecutado de nuevo); y el método *reset()*, que se utiliza para reiniciar un comportamiento desde el principio.

En general, JADE sigue un enfoque composicional para que los desarrolladores de aplicaciones puedan crear sus propios comportamientos, los cuales podrían estar basados en los simples provistos por JADE. De esta manera, la estructuración de tareas complejas se puede hacer como la agrupación de otras más simples. La clase *ComplexBehaviour* es el comportamiento que se estructuraría con los sub-comportamientos. Para ello, tiene dos métodos: *addSubBehaviour()* y *removeSubBehaviour()*.

Además de *ComplexBehaviour*, JADE define algunas otras subclases de comportamiento, por ejemplo, *SimpleBehaviour* es usado para implementar pasos atómicos del trabajo de un agente. Un comportamiento implementado por una subclase *SimpleBehaviour* es ejecutado por el planificador de JADE en solo paso. Adicionalmente, JADE define dos subclases más para enviar y recibir mensajes: *SenderBehaviour* y *ReceiverBehaviour*.

Por otro lado, JADE permite la recursividad en la ejecución de los comportamientos. Se asemeja a la técnica utilizada para las IGU, donde cada *widget* de la interfaz es una hoja de un árbol cuyos nodos intermedios son los *widgets* de contenedores especiales de sus hojas. Sin embargo, existe una diferencia importante: los comportamientos en JADE materializan las tareas a ejecutar, por lo que la programación y la suspensión de tareas son actividades que deben ser consideradas en los comportamientos a ejecutar.

4.3.1.2. Aspectos adicionales de JADE

La plataforma JADE ha sido evaluada con respecto a la escalabilidad y tolerancia a fallos de la misma, dos temas muy importantes para infraestructuras robustas de software distribuido. En cuanto a la escalabilidad, tres variables son de interés: el número de agentes en una plataforma, el número de mensajes entre agentes que se pueden emitir, y el número de conversaciones simultáneas que se pueden soportar.

Con respecto a la escalabilidad en el número de agentes, JADE define una arquitectura distribuida donde grupos de agentes relacionados se pueden implementar en el mismo contenedor o en contenedores independientes, lo cual puede ser usado para manejar el número de hilos por sitio como la carga de la red entre los hosts. La escalabilidad de JADE con respecto al número de mensajes es estrictamente dependiente de las capas de comunicación inferiores (IIOP, RMI, etc.). Finalmente, con respecto al número de conversaciones simultáneas (en las que un mismo o varios agentes pueden participar), JADE es escalable por su arquitectura de planificación de dos niveles: cuando un agente arranca una nueva conversación nuevos hilos no se inician ni nuevas conexiones se crean, sólo se crea un nuevo objeto de comportamiento. Así que la única sobrecarga asociada al inicio de las con-

versaciones es el tiempo para crear el nuevo objeto comportamiento y su ocupación de memoria; para lo cual se puede crear una colección de comportamientos con anterioridad.

Desde el punto de vista de la tolerancia a fallos, JADE no funciona muy bien debido al punto de fracaso representado por el contenedor especial FE único y, en particular, por el AMS. Un AMS replicado podría conceder mayor tolerancia a fallos a la plataforma. Sin embargo, la arquitectura de mensajería descentralizada de JADE permite que un grupo de agentes pudiera seguir trabajando incluso en la presencia de un fallo del AMS. JADE requiere de un mecanismo de reinicio para el contenedor FE y los agentes del sistema según FIPA.

JADE cumple muy bien el supuesto de FIPA de que sólo se debe especificar el comportamiento externo de los componentes del sistema, dejando los detalles de implementación y las arquitecturas internas para desarrolladores de agentes. Su modelo de agente muy general puede ser fácilmente especializado para implantar, por ejemplo, agentes reactivos o arquitecturas BDI. También, la abstracción comportamiento del modelo de agente permite una fácil integración con software externo. Por ejemplo, *JessBehaviour* que permite el uso de JESS como motor de razonamiento de un agente.

Por otro lado, JADE soporta la movilidad de un agente. Es posible mover o "clonar" un agente JADE en diferentes contenedores, pero dentro de la misma plataforma de agentes JADE. JADE no permite la movilidad interplataforma. También, JADE soporta agentes virtuales replicados, es decir, entidades virtuales que se identifican con un nombre (más precisamente, un identificador), pero no corresponden a agentes concretos. Detrás de ese nombre hay uno o más agentes concretos, llamados agentes de réplicas virtuales. JADE se encarga de enviar mensajes dirigidos a un agente virtual replicado (VR), es decir, a una réplica adecuada. Cada agente de réplica tiene su propio nombre que difiere del nombre del agente VR, esa gestión la hace JADE. Un agente VR no tiene una ubicación, mientras que, por supuesto, cada agente de réplica se ejecuta en un contenedor bien definido. Otras mejoras se han introducido en el Sistema de Gestión de Servicios para administrar los contenedores, entre éstas, la posibilidad de establecer un tiempo de espera (por mensaje, predeterminado) en la entrega de *ACLMessages* cuando el FE está desconectado temporalmente y una serie de correcciones para gestionar los problemas de red.

4.3.1.3. LEAP

LEAP (*Lightweight Extensible Agent Platform*) es un entorno de desarrollo y ejecución para agentes inteligentes, siendo el precursor de la segunda generación de plataformas compatibles con FIPA [38, 6]. Representa un gran desafío técnico, aspirando a convertirse en el primer entorno de desarrollo integrado de agentes capaz de generar aplicaciones de agentes y ejecutarlas

en entornos derivados de JADE, compatible con una gran familia de dispositivos (computadoras, teléfonos móviles, etc.) y de mecanismos de comunicación (TCP/IP, WAP, ...). LEAP es el nombre de un proyecto financiado por el programa IST (*Information Society Technologies*) de la Comunidad Económica Europea, que ha desarrollado la biblioteca LEAP, la cual puede ser usada desde la plataforma JADE, proporcionando un entorno de ejecución para aplicaciones móviles basada en Java (J2ME CLDC).

El objetivo de LEAP es suplir las necesidades de infraestructuras y servicios abiertos para aplicaciones móviles. Tres servicios básicos son provistos: gestión del conocimiento (para requisitos de conocimientos individuales), coordinación descentralizada (para la gestión de trabajos colaborativos), y gestión de la migración (planificación y coordinación del proceso). LEAP propone una plataforma de referencia para responder a las necesidades de comunicación y cooperación de los equipos móviles, sobre la base de las normas y capacidades de los teléfonos o dispositivos móviles actuales.

Se introduce la noción de aprendizaje móvil, vital en los enfoques basados en agentes para sistemas móviles. El aprendizaje móvil se basa en la idea de que "en cualquier momento y en cualquier lugar se debe aprender". Por lo tanto, el aprendizaje de los contenidos y servicios debe estar siempre disponible al agente, cuando lo quiera y donde quiera que vaya. Sin embargo, el ambiente de aprendizaje móvil tiene una serie de limitaciones, relacionadas con las limitaciones de los propios dispositivos móviles, en cuanto a su potencial de procesamiento, capacidad de memoria baja, vida limitada de la batería, etc. Estas limitaciones se han reducido en la actualidad, por la incremento en las capacidades computacionales de los dispositivos móviles actuales. Otras limitaciones están relacionadas con las redes inalámbricas, que tienen alta latencia, retardos de transmisión y bajo ancho de banda, lo que implica que el tamaño de los datos intercambiados debe ser optimizado.

Básicamente, la plataforma que se ha propuesto se llama JADE-LEAP, apta paran dispositivos móviles. JADE-LEAP es una extensión de la plataforma JADE que se puede implementar en PCs, servidores, y en dispositivos con recursos limitados, como los teléfonos móviles. Para ello usa:

- *Pjava*: para ejecutar JADE-LEAP en los dispositivos portátiles que soportan J2ME CDC o PersonalJava, como los PDAs.
- *Midp*: para ejecutar JADE-LEAP en los dispositivos portátiles que soportan MIDP1.0, tales como los teléfonos móviles que pueden usar Java.
- *Android*: para ejecutar JADE-LEAP en dispositivos compatibles con Android 2.1 o posterior.
- *DotNet*: para ejecutar JADE-LEAP en PC y servidores de la red que ejecutan Microsoft .NET Framework.

Estas versiones proporcionan la misma API (*Application Programming Interface*) para los desarrolladores, por lo que ofrece una capa homogénea sobre una diversidad de dispositivos y tipos de red, excepto la versión de MIDP que tienen algunas características no compatibles en comparación

con las otras versiones de JADE-LEAP. La Figura 4.5, que ha sido tomada de [6], muestra un esquema general para el entorno de ejecución de JADE-LEAP.

JADE-LEAP proporciona dos modos de ejecución para adaptarse a las limitaciones de un dispositivo dado. El modo de ejecución normal *Stand-alone* sugeridas en entornos de red, y soportada por Pjava y Android. En este modo de ejecución un contenedor completo se ejecuta en el dispositivo donde se activa JADE. El modo de ejecución *Split* es obligatorio en MIDP, y recomendable en Pjava. En este modo de ejecución el contenedor se divide en una interfaz (que se ejecuta en el dispositivo donde se activa JADE) y un FE (que se ejecuta en un servidor remoto), unidos por medio de una conexión permanente.

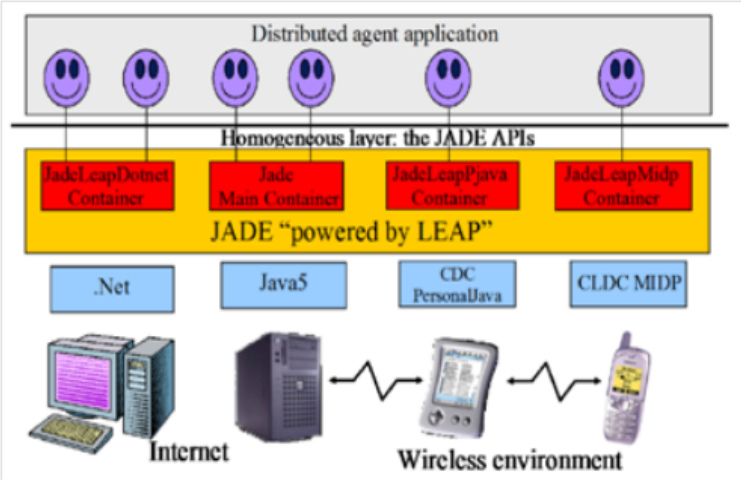

Figura 4.5: El entorno de ejecución JADE-LEAP.

Este modo de ejecución utiliza menos memoria y necesita menos procesamiento en el dispositivo móvil, ya que la interfaz es más ligera que un contenedor completo. Además, las tareas de procesamiento intensivo están en el servidor remoto. También, tiene la ventaja de minimizar el ancho de banda requerido y optimizar la conexión inalámbrica con el contenedor principal, puesto que las comunicaciones con el contenedor principal se realizan a través del FE, sin el uso del enlace inalámbrico.

Un detalle interesante es la arquitectura multiagente propuesta del sistema de aprendizaje móvil, basado en la conciencia del contexto, el aprendizaje de contenidos y la plataforma JADE-LEAP. La Figura 4.6, tomada de [6], muestra la arquitectura de aprendizaje propuesta para JADE-LEAP, en ella aparecen los agentes que se detallan a continuación:

- *Agente de Interfaz*: se trata de un agente que tiene varias tareas: autenticación de los agentes, autorización de usuarios, etc. Actúa como un punto de comunicación entre los dispositivos y el sistema. Además, envía solicitudes a la plataforma JADE-LEAP para crear y enviar agentes a los dispositivos móviles, e informa al Agente Supervisor para que actualice o almacene información relativa al perfil de los agentes nuevos.
- *Agente Sensor*: sensa el ambiente y reacciona en consecuencia a los cambios. También, tiene un papel de supervisión y seguimiento del agente que aprende en su proceso de aprendizaje. Envía información sobre las funciones del dispositivo (tamaño de la memoria, potencia de procesamiento, conectividad disponible, costos de comunicación, ancho de banda, y nivel de la batería), que se almacenan en la base de datos de las características del contexto. Además, envía algunos datos del proceso de aprendizaje, entre éstos: duración, nivel de concentración (cuántas veces fue interrumpido por un evento externo, etc.), y ubicación actual del agente que aprende.
- *Agente Tutor*: agente móvil que gestiona el proceso remoto de aprendizaje. Las principales tareas del agente tutor son: gestionar el material de aprendizaje, guardar el sitio donde se encuentra el agente aprendiendo para empezar desde ese punto la próxima sesión, asegurar la visualización de los servicios y contenidos de aprendizaje de acuerdo con las preferencias del agente y capacidades de los dispositivos, y, en colaboración con el agente de sensor, recuperar el contenido de la pruebas hechas al agente que aprende (sus respuestas).
- *Agente consciente del contexto*: consiste en un módulo analizador y otro de adaptación al contexto. El módulo de análisis revisa la información enviada por el agente sensor y extrae datos relacionados con el contexto. Después, clasifica esos datos para ser tratados por el módulo de adaptación del contexto, que envía la información del perfil del usuario y la información del contexto al agente supervisor. El módulo de adaptación al contexto utiliza la información obtenida para, por ejemplo, si el usuario tiene una conexión de ancho de banda limitada entonces reducir el contenido multimedia, y en el peor de los casos, sustituirlo por texto. El agente de adaptación al contexto predice el contexto futuro para determinar las actividades apropiadas a realizar. Para el ejemplo anterior, se transmitiría sólo los datos con un tamaño pequeño.
- *Agente Supervisor*: tiene el papel de supervisión de la funcionalidad del sistema. Es el único agente que tiene la capacidad de cambiar y actualizar los datos de los repositorios de objetos de aprendizaje: (características del contexto, perfiles de los agentes que aprenden, etc.), con la ayuda del agente de interfaz.
- *Agente de Adaptación*: personaliza los contenidos de aprendizaje a los estilos de aprendizaje y los dispositivos. Esta tarea se realiza a través de dos módulos, el de adaptación de estilos de aprendizaje y el de adaptación de contenidos. Estos dos módulos se coordinan entre sí, es decir, el módulo

de adaptación de estilo busca los objetos de aprendizaje apropiados, luego, el módulo de adaptación de contenidos ajusta los cursos y estrategias de enseñanza del objeto, según el perfil de usuario y del dispositivo.

- *J2ME (Java2 Micro Edition)*: se trata de la aplicación que hace el despliegue, se ejecuta en el dispositivo móvil de aprendizaje (PDA, teléfonos inteligentes, etc.). Muestra una interfaz usable y apropiada, que se adapta a las capacidades de visualización de la pantalla. A través de esta interfaz, da acceso a los usuarios al material de aprendizaje.

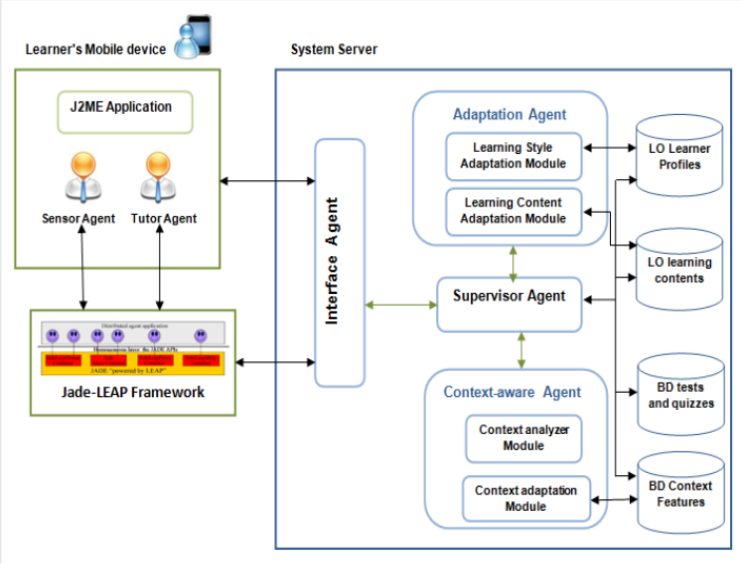

Figura 4.6: Arquitectura de aprendizaje propuesta para JADE-LEAP.

4.3.2. *FIPA-OS*

FIPA-OS fue la primera implementación de código abierto del estándar FIPA, lanzada por primera vez en agosto de 1999, cuya licencia es EPL [9]. FIPA-OS es compatible con la mayoría de las especificaciones de FIPA actuales. Provee un conjunto de herramientas basadas en componentes implementadas en Java. Uno de los aportes más significativos es la versión pequeña de FIPA-OS (μ FIPA-OS para PDA y teléfonos móviles inteligentes), que ha sido desarrollado por la Universidad de Helsinki.

En general, las herramientas de FIAP-OS están orientadas a la construcción de agentes usando componentes compatibles con FIPA. Provee compo-

nentes obligatorios (es decir, componentes requeridos por todo agente FIPA-OS para ejecutarse), componentes con implementaciones intercambiables, y componentes opcionales (por ejemplo, componentes que un agente FIPA-OS puede opcionalmente usar). En la Figura 4.7, extraída de [9], se destacan los componentes disponibles y su relación con el resto.

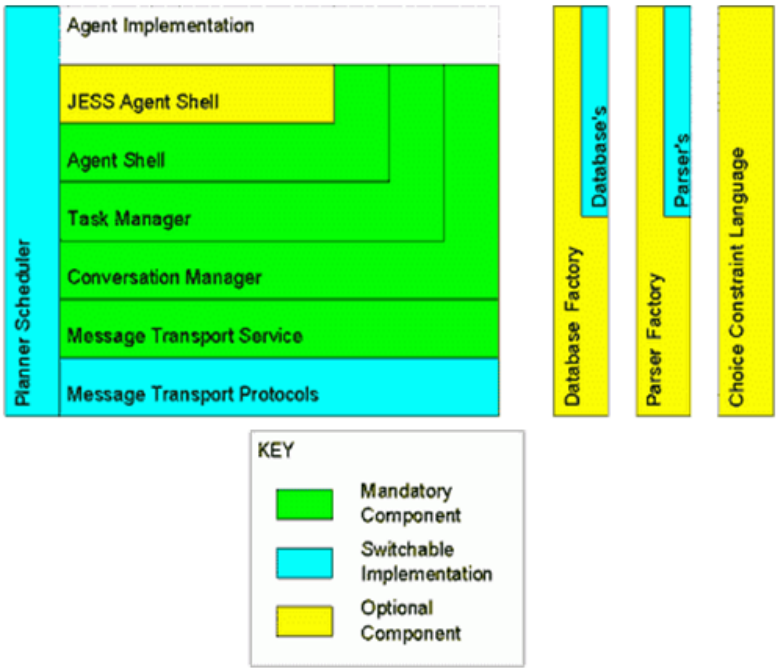

Figura 4.7: Componentes de FIPA-OS.

La base de datos, el analizador y los componentes CCL son opcionales, y no tienen una relación explícita con los otros componentes. El Planificador General tiene la capacidad de interactuar con todos los componentes de un agente. La clase *FIPAOSAgent* proporciona un intérprete para la ejecución del agente. El Administrador de Tareas proporciona la capacidad de dividir la funcionalidad de un agente en unidades más pequeñas disjuntas. El objetivo es que cada tarea tiene la capacidad de enviar y recibir mensajes, y tienen poca o ninguna dependencia del sitio donde se ejecuta el agente.

El Administrador de Conversación ofrece la posibilidad de rastrear el estado de una conversación, así como mecanismos para agrupar los mensajes de la misma conversación juntos. Si se especifica en una conversación un protocolo determinado (FIPA o definido por el usuario), el Administrador de Conversación se asegura del cumplimiento de dicho protocolo en la conversación.

El MTS (Servicio de Transporte de Mensajes, *Message Transport Service*) proporciona la capacidad de enviar y recibir mensajes hacia y desde un agente. Una variedad de versiones de MTS se proporcionan con FIPA-OS, compatibles con: RMI, SSL-RMI, http, etc.

Los agentes FIPA-OS pueden aprovechar las capacidades de razonamiento que provee JESS. Además, provee mecanismos para interactuar con bases de datos. A nivel de comunicación, permite que el contenido de un mensaje se exprese en una forma que contiene sólo la información semántica. Adicionalmente, puede llevar ese contenido a XML, *Properties Java script* o *Abstract Binding*. Soporta las especificaciones de FIPA a nivel de comunicación (FIPA ACL Message Structure Specification, FIPA Interaction Protocol Library Specification, etc.), de administración de agentes (FIPA Agent Management Specification), de transporte de mensajes (FIPA ACL Message Structure Specification, etc.) y de integración de agentes (FIPA Agent Software Integration Specification).

4.3.3. Zeus

Es una plataforma abierta desarrollada en Java, la cual posee tres bibliotecas [40, 41, 10]: una con utilitarios para desarrollar agentes, otra con herramientas para su creación, y una de componentes para agentes. Puede ser considerada como un conjunto de herramientas para construir aplicaciones multiagentes, que permiten definir funcionalidades para agentes genéricos y la planificación y programación de acciones de un agente. Por otra parte, Zeus ofrece servicios para las comunicaciones del agente, utilizando FIPA ACL para el transporte de mensajes y TCP/IP como el protocolo de transporte. El enfoque de Zeus para la planificación y programación representa las metas y acciones usando descripciones, que incluyen los recursos que necesitan y las pre-condiciones que deben cumplirse. Esto permite que las metas se representen mediante una cadena de acciones que tienen que cumplirse. Esta cadena de acciones se construye mediante un proceso de encadenamiento hacia atrás.

En general, Zeus es un entorno integrado para la rápida construcción de aplicaciones de agentes colaborativos. Fue realizado por el Laboratorio de Investigación en Sistemas Inteligentes de la British Telecom. La documentación Zeus es abundante, y pone un fuerte énfasis en la importancia de los aspectos metodológicos, ya que dicha metodología de creación de agentes es vital para el uso del conjunto de herramientas de Zeus. La metodología de Zeus descompone en cuatro etapas el desarrollo de un agente: análisis de dominio, diseño, realización y soporte para la ejecución.

Para la etapa de análisis no se proporciona ninguna herramienta de software. Como los agentes se definen por su función y por su comportamiento, el modelado de las funciones se realiza con diagramas de clases UML y pa-

trones de diseño. La etapa de diseño del agente consiste en la definición del modelo de agente para cada uno de éstos. Los agentes pueden ser orientado a tareas, a objetivos o de colaboración. La etapa de realización o desarrollo del agente consiste en cinco actividades:

- *Creación de la ontología*: para lo cual tiene la herramienta Zeus Ontology Editor, que permite la definición del conocimiento declarativo, representa los conceptos importantes dentro del dominio de aplicación,
- *Creación del agente*: el agente Zeus genérico está configurado para cumplir con las responsabilidades específicas que se le definan, para lo cual Zeus propone la herramienta Zeus Agent Editor,
- *Creación de las características del agente*: define los atributos de los agentes,
- *Configuración de las tareas del Agente*: define los parámetros de ejecución de las tareas de los agentes,
- *Implementación del agente*: herramienta para generar el código fuente del agente, a través del Code Generation Editor.

Como se ve, para esta etapa, las actividades son soportadas por herramientas de Zeus. Las ontologías se pueden reutilizar, no así los agentes.

La fase de soporte para la ejecución o despliegue de los agentes se hace a través de la herramienta Visualiser. Visualiser permite observar al SMA desde diferentes puntos de vista: la organización de agentes, las interacciones, la descomposición de la tarea global, las estadísticas y los estados internos de los agentes. También es posible controlar los estados de los agentes individualmente, e incluso configurar los agentes en tiempo de ejecución. Las herramientas de implementación utilizan interfaces gráficas fáciles de usar. Las modificaciones en la composición de la sociedad de agentes requieren la recompilación del sistema.

La principal particularidad de Zeus es la completa integración de todas las fases, desde el diseño hasta la implementación. Proporciona herramientas, utiliza técnicas actuales de ingeniería de software (patrones de diseño, UML). Los servicios provistos por Zeus a los agentes son:

- *Servidor de nombre y directorio de agentes*: para facilitar el descubrimiento de la información y de agentes.
- *Visualizador de la sociedad de agentes*, el cual posee los siguientes componentes: i) *Society Viewer*: permite ver todos los agentes de una sociedad, las interrelaciones, los mensajes intercambiados, ii) *Reports Tool*: descompone y distribuye la ejecución de las tareas, iii) *Agent Viewer*: observa y monitorea los estados internos de los agentes, iv) *Control Tool*: revisa y modifica de forma remota los estados internos de los agentes, y, v) *Statistics Tool*: muestra estadísticas de los agentes o de la sociedad de agentes en diversos formatos.

A nivel de comunicación, Zeus posee:

- Un lenguaje de comunicación entre agentes basadas en las performativas de FIPA;

- Un sistema de pase de mensajes basado en conectores asíncronos;
- Un editor para la descripción de ontologías de dominios, basado en el lenguaje de contenidos ACL;
- Un lenguaje de representación de conocimiento basado en marcos, para representar los conceptos de un dominio dado.

Como Zeus utiliza la última versión de componentes de la interfaz gráfica *Swing*, se puede ejecutar en cualquier plataforma con la máquina virtual JDK 1.2. ZEUS ha sido probado con éxito en Microsoft Windows 95/98/NT4 y en Solaris.

4.3.4. JACK

Las aplicaciones JACK consisten en una colección de agentes autónomos que toman entradas desde el medio ambiente y pueden comunicarse con otros agentes [11, 36, 69]. Cada agente se define en términos de sus objetivos, conocimiento y capacidades sociales, y se deja en su entorno para que realice su función de manera autónoma. JACK posee una licencia LGPL, siendo un entorno multiplataformas para la construcción, el funcionamiento y la integración de SMA.

JACK permite crear aplicaciones para entornos dinámicos y complejos: cada agente es responsable de la consecución de sus propios objetivos, reaccionando a los eventos y comunicándose con los otros agentes del sistema. No hay necesidad de programar explícitamente las interacciones de toda la aplicación, las interacciones surgen como un subproducto de los objetivos individuales y las capacidades de los agentes, lo que lo hace muy interesante para el desarrollo de aplicaciones "emergentes" [29].

Está construido basado en la arquitectura BDI (creencias, deseos, intenciones) . En JACK, los agentes se definen en términos de sus creencias (lo que saben y lo que saben hacer), sus deseos (qué metas le gustaría alcanzar), y sus intenciones (las metas que actualmente se han comprometido a alcanzar). Esta completamente escrito en Java, es altamente portable y funciona en cualquier dispositivo, desde PDAs a servidores multi-CPU. Por estar basado en Java, puede ejecutar múltiples hilos en varios CPUs. Además, tiene una IGU independiente, y se integra fácilmente con bibliotecas de terceros. Sus principales características son:

- Tiene requisitos de recursos bajos, diseñados para manejar cientos de agentes,
- Las comunicaciones entre agentes son transparentes,
- Provee herramientas gráficas para el desarrollo de agentes.

JDE es el entorno de desarrollo de JACK, es un IDE comercial desarrollado por Agente Oriented Software Ltd. JDE permite a los desarrolladores

de agentes la edición del código con un editor integrado. Además, el IDE proporciona un editor de gráficos que permite la visualización de los componentes de un SMA. También, permite compilar y ejecutar una aplicación directamente desde el IDE.

Posee el Lenguaje de Agentes JACK (JAL), que se basa en los sistemas de planificación reactivos para arquitecturas de agente BDI, similar a los lenguajes Jason, y 3APL. Sin embargo, en lugar de proporcionar un lenguaje basado en lógica, JAL es una extensión de Java aplicando algunas características de los lenguajes lógicos. Se añade un número de construcciones sintácticas de Java, lo que permite a los programadores crear planes y bases de creencias, todo ello de manera gráfica a través del IDE.

En JAL, los planes pueden estar compuestos por métodos de razonamiento, y se agrupan en capacidades que, juntas, componen una habilidad específica de un agente. Otro mecanismo de estructuración presente en JAL es la capacidad de definir grupos de agentes, u organizaciones de agentes, una noción que es cada vez más importante en el diseño orientado a agentes para procesos de auto-organización.

Así, JACK provee como un entorno para la creación, el funcionamiento y la integración de SMA basados en Java, utilizando un enfoque basado en componentes. JACK se centra principalmente en la etapa de desarrollo. El juego de herramientas (JDE) es provisto a través de un ambiente gráfico. El compilador JAL traduce los programas de JAL a programas Java puro, además provee una biblioteca de clases de apoyo, llamada Jack Agent Kernel.

Cuando se crea un agente, una instancia se inicia en el sistema, la cual puede hacer varias cosas diferentes, por ejemplo, responder a objetivos establecidos explícitamente por otro agente o el usuario, o a estímulos externos que se representan como eventos. En ambos casos, el agente responderá si se cree que el evento u objetivo se puede responder. Al recibir un evento, el agente busca a través de todos sus planes disponibles los que son relevantes para la solicitud y lo aplica. Para ello, ejecuta las acciones dentro del plan elegido. Así, es un modelo donde se considera a los agentes como colecciones de planes que se ejecutan en condiciones específicas.

El desarrollo en JACK es una combinación de la programación típica en Java con clases elementales, extendida por JAL para los componentes específicos del agente. Las extensiones permiten describir el comportamiento del agente, sus capacidades, planes y eventos. El compilador JAL traduce esto en Java puro. Una herramienta de software gráfico también se proporciona para facilitar la gestión de grandes proyectos. La etapa de desarrollo JACK es aplicable a SMA compuestos de agentes BDI.

Aunque JACK aborda el tema de la comunicación entre los agentes y proporciona una red de comunicaciones, no hace uso de ningún lenguaje de comunicación de agentes estándar. Se utilizan mensajes que pasan a través de la capa de comunicación JACK. La red se ejecuta como una capa de red y, de forma predeterminada, utiliza UDP (*User Datagram Protocol*) , como

protocolo de transporte. Debido a que UDP es no orientado a conexión y no garantiza la entrega de paquetes, no hay garantía de la entrega de los mensajes.

Siguiendo el modelo de agente BDI, las creencias del agente son todos los elementos de información que tiene sobre su entorno, o de otros agentes o de sí mismo. Los deseos son simplemente metas, o lo que el agente desea lograr. Las intenciones corresponden a lo que se suele llamar planes. JACK facilita la planificación a través de una clase base "plan", que es extensible para la creación de planes de gran alcance.

JACK usa por omisión una clase de agente que puede ampliarse mediante JAL; en virtud de que JAL proporciona una declaración de eventos, que permite a los agentes no sólo razonar acerca de los planes que corresponden a eventos específicos, sino también razonar sobre lo que se planea utilizar para un evento específico. Este se considera un tipo de meta-razonamiento. JACK no proporciona directamente ninguna funcionalidad de alto nivel que apoye la adaptabilidad.

JACK también proporciona el paquete *SimpleTeam*, que facilita la construcción de planes de trabajo compartido. Finalmente, JACK contiene un examinador de objetos (*JACOB*) que proporciona el modelado de objetos para la comunicación de objetos entre los agentes. Los agentes pueden transmitir solamente datos.

4.3.5. JIAC

JIAC (*Java-based Intelligent Agent Componentware*) es un sistema basado en Java que permite el desarrollo y el funcionamiento de agentes y servicios distribuidos a gran escala [60, 67, 66]. La plataforma soporta el diseño, implementación y despliegue de sistemas de agentes. Todo el proceso de desarrollo de agentes, desde la concepción hasta la implementación de SMA completos, puede ser soportado por JIAC. También permite la reutilización de agentes y servicios, e incluso modificarlos en tiempo de ejecución. Las características centrales de JIAC son la distribución, la escalabilidad, la adaptabilidad y la autonomía.

JIAC proporcionada una amplia biblioteca que se compone de servicios, componentes, e incluso agentes, que se pueden integrar en una aplicación. Los agentes individuales se basan en una arquitectura de componentes, que proporciona la funcionalidad básica para la comunicación y la gestión de procesos. Se han implementado extensiones a dominios específicos, que facilitan el desarrollo rápido y sencillo de aplicaciones en esos dominios. Algunas de sus propiedades son:

- Es ejecutable en dispositivos de diferentes tipos, desde dispositivos móviles hasta sistemas de servidores.

- Permite la implementación de agentes escalables, independientes de los dispositivos.
- Ofrece servicios para la construcción y despliegue de agentes.
- Ofrece una infraestructura de comunicación que se abstrae de los protocolos de red y de transporte usados.

Una versión de JIAC, llamada MicroJIAC, se desarrolló en 2006 siguiendo un enfoque de diseño de abajo hacia arriba para aprovechar plenamente todas las características de Java, reduciendo lo requerido de él. El tamaño del archivo de Java se reduce al eliminar las clases no utilizadas. Además, el análisis y procesamiento de la configuración de la aplicación se realiza en tiempo de compilación, por lo que un analizador XML no es necesario en tiempo de ejecución. Eso le permite a MicroJIAC:

- Dar soporte a aplicaciones tiempo real.
- Ser adaptable a nivel de sitios (nodos) y agentes.
- Permitir la migración.

En particular, en MicroJIAC una aplicación se compone de uno o más nodos que alojan a uno o más agentes. Las aplicaciones son construcciones virtuales de grupos de nodos y agentes juntos. Un nodo es el entorno de ejecución para agentes, y proporciona las rutinas de inicio. Cada nodo se asocia con una única JVM. De están manera, los recursos esenciales y dependientes de la plataforma que los agentes necesitan, como las conexiones de red, los obtienen a través del nodo. JIAC permite la personalización de los nodos con componentes específicos. Los agentes hacen referencias directas a los componentes y funciones internas de un nodo a través de interfaces. De esta manera, cada nodo es extensible a través de componentes. Este mecanismo de extensión permite encapsular las funcionalidades del sistema a nivel de los nodos y de todos los agentes que se ejecutan en él. Un sistema de comunicación, por ejemplo, es una extensión del nodo. El acceso al mismo es otorgado a través de una interfaz definida en el nodo, así el desarrollador de agentes no tiene que preocuparse por los detalles de la tecnología de comunicación en uso. Otros nodos pueden proporcionar las mismas interfaces para el canal de comunicación entre agentes, pero con diferentes implementaciones de bajo nivel.

Los agentes son las entidades en MicroJIAC que ejecutan aplicaciones especificas. Ellos se componen de cuatro tipos de elementos de agentes: sensores, actuadores, comportamientos reactivos y comportamientos activos. Además, hay una capa de gestión simple dentro de cada agente. Esta capa es responsable de la inicialización y de la correcta transición entre los elementos instalados en el agente, así como de la dotación y eliminación de funcionalidades en el agente. Por lo general, los desarrolladores de agentes no tienen que usar sus propias clases de agentes. En el caso de que no haya otra solución, el desarrollador debe tener en cuenta que una extensión o modificación a la clase agentes puede restringir la portabilidad.

Por otro lado, los agentes sólo pueden procesar la información si hay un vocabulario para describir dicha información. El vocabulario se define en ontologías, y refleja el entorno del agente así como sus eventos internos. MicroJIAC ofrece un tipo abstracto de datos que encapsula el mundo del agente. El desarrollador debe describir el dominio con el modelo de datos adecuado. Por otro lado, las ontologías se definen directamente como clases Java con los atributos que las caracterizan.

En JIAC, hay dos formas diferentes para generar conocimiento: un sensor percibe el entorno y lo transforma en nuevo conocimiento para el agente; o los elementos internos del agente proveen nuevos conocimientos debido a sus actividades. La representación del conocimiento mediante clases Java permite intercambiarlo.

Dos elementos fundamentales del modelo de agente de MicroJIAC son los elementos que permiten la manipulación y la lectura del entorno del agente (sensores y actuadores): los sensores muestrean activamente el medio ambiente y proporcionan los datos recogidos al agente. Los actuadores manipulan esos datos sobre el medio ambiente. El contenido de los datos generados por los sensores se especifica explícitamente en la representación del conocimiento. Así, la información intercambiada con el medio ambiente del agente se realiza en los sensores y actuadores. Estos datos desacoplados permiten su uso en la planificación y en la definición de funcionalidades de los agentes, totalmente independiente de los dispositivos. Cada actuador define una interfaz a través de la cual los otros elementos del agente pueden acceder a él. El desarrollador de agentes debe asegurarse que la lógica interna del agente permanece oculta, y se mantenga esa interfaz definida para los sensores y actuadores.

Un agente MicroJIAC tiene dos tipos de comportamiento. El comportamiento reactivo, que sólo se activa si los datos específicos están disponibles. La ejecución de comportamientos reactivos está desacoplada de la aparición de nuevos datos en la memoria, por lo tanto, son asíncronos de las operaciones de lectura del sensor. El segundo tipo de comportamiento es el comportamiento activo. Trabaja periódicamente con los datos de la memoria a largo plazo, y es capaz de manipular el entorno del agente a través de los actuadores. Los comportamientos activos se utilizan para realizar lógicas y funcionalidades de planificación de orden superior. Ambos tipos de comportamiento, reactivo y activo, deben realizarse sin introducir dependencias específicas con los dispositivos.

Por otro lado, MicroJIAC usa dos tipos de hilos. Hilos para los nodos (realiza todo los procesos para la gestión de los nodos) e hilos para cada agente en ejecución. Para ello se utilizó una especialización de la clase *java.lang.Thread*. Durante la ejecución de un nodo o agente, los hilos sólo pueden ser creados por el sistema completo, de lo contrario la lógica que se ejecuta en el nuevo hilo no podría acceder a los atributos o referencias relacionadas con el ámbito de la aplicación. Este concepto está presente en cada

versión de MicroJIAC, pero sólo la de tiempo real la implementa completamente.

El ciclo de vida de un agente contiene cuatro estados: creado, inicializado, iniciado y destruido. Por omisión, los nodos y los agentes se rigen por un administrador de ciclo de vida. Hay una interfaz específica que define los métodos para las transiciones de estado. Por otro lado, la planificación es una parte integral de MicroJIAC. Los componentes de planificación pueden ser: periódicos y esporádicos. Para ello, el planificador usa los siguientes parámetros:

- Plazo: duración de la ejecución.
- Prioridad de los pedidos.
- Costos estimados.

La ejecución periódica tiene dos parámetros adicionales: el tiempo de inicio de la primera ejecución y la distancia temporal entre dos ejecuciones. La ejecución esporádica, en cambio, amplia el conjunto de parámetros con el tiempo mínimo entre dos ejecuciones. Los comportamientos activos siempre son ejecuciones periódicas, mientras que las conductas reactivas son siempre esporádicas. La edición en tiempo real de MicroJIAC maneja el tiempo de forma mucho más restrictiva que las ediciones normales.

4.3.6. Janus

Janus es una plataforma multiagente de código abierto implementado en Java 1.6, que permite a los desarrolladores crear aplicaciones Web y multiagentes [50, 12]. Proporciona un conjunto completo de elementos para desarrollar, ejecutar, exhibir y controlar aplicaciones basadas en agentes. Janus puede ser utilizado como una plataforma orientada a agentes, orientada a organizaciones de agentes u holónica. Se basa en el metamodelo organizacional CRIO, en el que los conceptos de roles y organización son entidades de primer nivel. También gestiona el concepto de agentes recursivos y holón. Así, Janus es una plataforma multiagentes que hace frente a la aplicación y el despliegue de Sistemas Holónicos (SH)) y multiagentes. CRIO adopta el enfoque de desarrollo de sistemas basado en modelos (MDA, *Model Driven Architecture*), y los elementos de este metamodelo se organizan en tres ámbitos: (I) Dominio del problema, definición del problema por el usuario en cuanto a los requisitos, las organizaciones, los roles y las ontologías. (II) Dominio de solución, se dirige a la solución holónica del problema descrito antes. (III) Dominio de implementación, describe la implementación en la plataforma elegida, que corresponde a la plataforma Janus.

SH se basan en entidades auto-similares y recurrentes, llamadas holones. En los SMA, la visión de los holones se acerca más a la de agentes recursivos

o compuestos. Un holón es una estructura auto-similar compuesta de entidades. Un holón se puede ver, en función del nivel de observación, ya sea como una entidad autónoma "atómica" o como una organización de holones. Usando la perspectiva holónica, el diseñador puede modelar un sistema con entidades de diferentes granularidades. Esa forma de diseño es permitida por Janus.

La arquitectura de la plataforma Janus se muestra en la Figura 4.8, que ha sido tomada de [12]. El corazón de la plataforma es su núcleo, el cual implementa el modelo de organización y el concepto de holón. Este núcleo también tiene un módulo de simulación y holones a cargo de las operaciones de la plataforma. Las diversas características proporcionadas por el núcleo Janus se describen a continuación:

- *El sistema de gestión de la organización*, maneja las organizaciones (grupos) de agentes. Proporciona mecanismos para su creación, instanciación, etc.
- *El sistema de gestión de holón*, herramientas necesarias para la gestión del ciclo de vida de un holón: identificar, poner en marcha, parar, etc.
- *El canal de comunicación*, controla el intercambio de mensajes dentro de la plataforma y con plataformas remotas,
- *El sistema de gestión de identidades*, proporciona los mecanismos necesarios para la asignación de una dirección única (GUID) para todos los elementos de la plataforma: holones, grupos, roles, etc.
- *Directorios/Repositorios*, mantiene directorios de todos los grupos, holones, etc. definidos en el núcleo.
- *Sistema de programación y observación de holones*, Janus ofrece dos políticas básicas para la programación de holones: un modelo de ejecución concurrente y un motor síncrono.
- *Sistema de registro (logging)*, todas las aplicaciones basadas en Janus tienen acceso a un sistema de registro integrado a la plataforma.

La arquitectura de Janus sigue a FIPA. Sólo ACL no se aplica plenamente.

4.3.7. Multi-AGent Environment - MAGE

MAGE (Multi-AGent Environment) es un entorno orientado a agentes para el diseño, la integración y la ejecución de software basados en agentes [13]. MAGE cuenta con herramientas para soportar la representación del estado mental, la planificación y el razonamiento de un agente, y la negociación, la cooperación y la comunicación entre agentes. MAGE es un software totalmente implementado en Java. Su *middleware* cumple con las especificaciones FIPA. Además, simplifica la integración de aplicaciones a través de múltiples esquemas de reutilización y diseño de software orientado a agentes con una IGU, que a su vez permite simplificar la administración. El tiempo de ejecución puede ser distribuido a través de varias máquinas. La

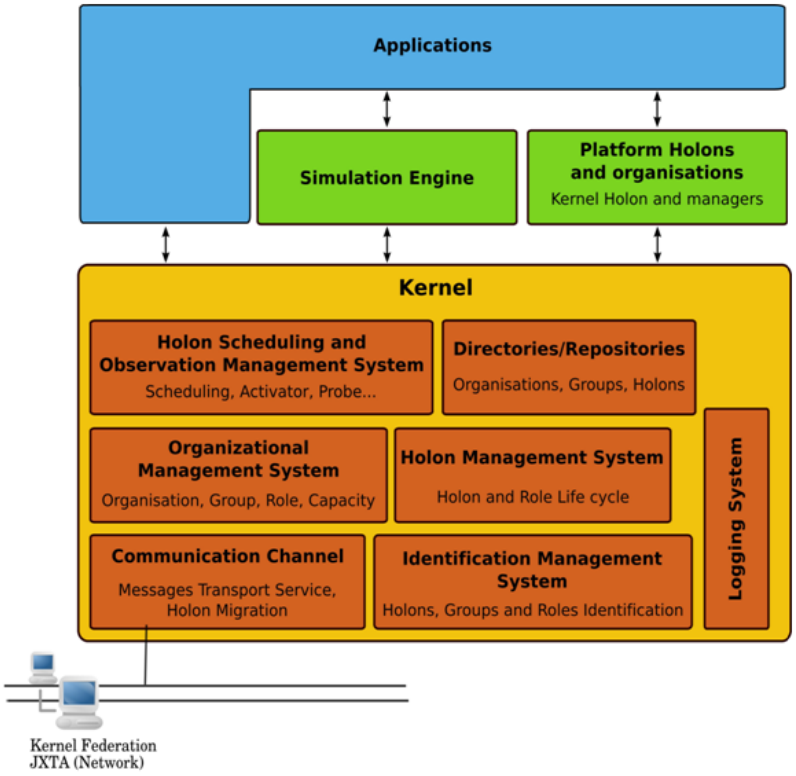

Figura 4.8: Arquitectura general de la plataforma Janus.

configuración puede cambiarse incluso, en tiempo de ejecución, y puede ser hecha por los agentes al moverse de una máquina a otra.

La arquitectura de la plataforma MAGE se muestra en la Figura 4.9 (disponible desde [13]), y consta de los siguientes componentes:

- *Sistema de gestión de agentes* (AMS): ejerce el control y supervisión de la plataforma MAGE.
- *Manejador de directorio* (DF): proporciona servicios de páginas amarillas a los agentes, tales como el registro de servicios, búsqueda y actualización de servicios, etc.
- *Servicio de transporte de mensajes* (MTS): es el método de comunicación predeterminado entre agentes de distintas plataformas de agente.
- *Agente*: es el actor fundamental en MAGE que combina uno o más capacidades de servicio en un modelo de ejecución unificado e integrado. Cada agente puede ser diseñado con una función particular para diferentes aplicaciones.
- *Software*: describe colecciones de instrucciones ejecutables accesibles a través de un agente.

Además, se han previsto dos módulos auxiliares para apoyar el diseño de sistemas de agentes: una biblioteca de agentes y componentes con funcionalidades específicas. El usuario puede construir agentes con muchos tipos de componentes provistos por MAGE. También, el usuario puede seleccionar el tipo de agente adecuado usando la biblioteca de agentes.

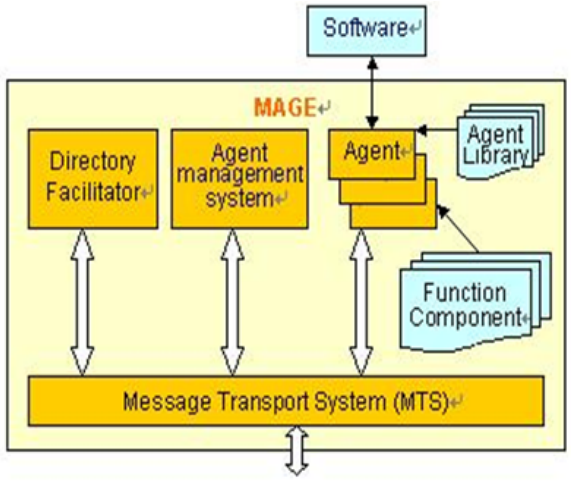

Figura 4.9: Arquitectura de MAGE.

La arquitectura de un agente MAGE consta de seis componentes: el núcleo del agente, las capacidades básicas, el sensor, el comunicador, los módulos funcionales, y la base de conocimientos. La Figura 4.10 (extraída de [13]), muestra la arquitectura genérica de un agente.

El ciclo de vida del agente MAGE incluye seis estados: iniciado, activo, en espera, suspendido, en tránsito y desconocido. MAGE soporta la transición entre los diferentes estados (ver Figura 4.11, tomada de [13]).

MAGE proporciona una herramienta gráfica para el diseño, llamada: XMIDE [13]. El diseño orientado a agentes con la IGU, permite la especificación de un sistema de agentes a partir de dos modelos: el Modelo de Agentes (describe la relación jerárquica entre las diferentes clases de un agente), y el Modelo de Interacción (describe las responsabilidades de una clase de agentes, los servicios que proporcionan y las relaciones entre los agentes). Esto incluye la sintaxis y la semántica de los mensajes utilizados para la comunicación entre agentes y con otros componentes del sistema. MAGE es una de las plataformas de agentes que están conectadas a *agentcities.net*.

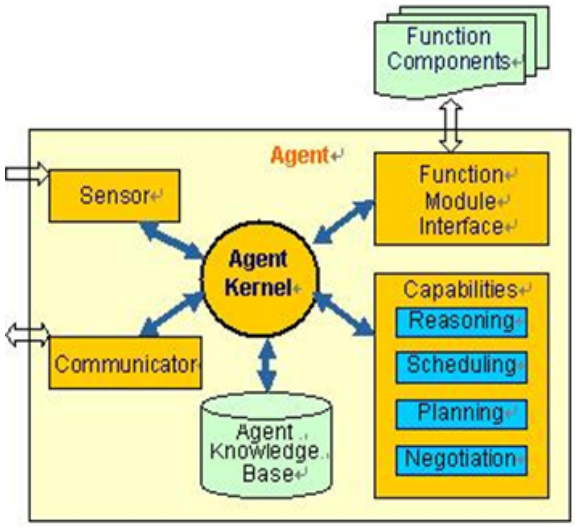

Figura 4.10: Arquitectura de un agente de MAGE.

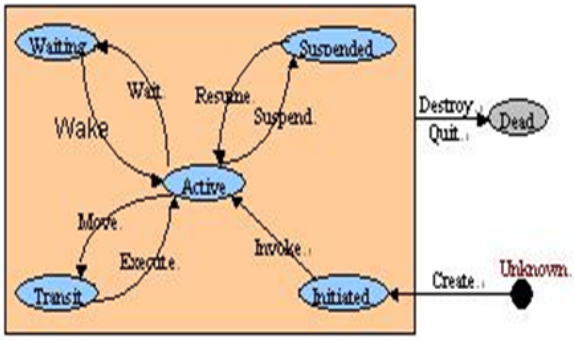

Figura 4.11: Ciclo de Vida de un agente de MAGE.

4.3.8. OMAS

OMAS (*Open System Multi-Agent*) es una plataforma multiagente que permite la creación de prototipos de agentes cognitivos [32]. En la actualidad funciona en plataformas Windows y ofrece varios modelos de agentes. El objetivo fue desarrollar una herramienta en la que un agente pudiese ser programado fácilmente, mediante la adición de poco código como *plugin*. Para ello, se tomaron las siguientes decisiones de diseño:

- Los agentes son totalmente independientes e interactúan en forma P2P (no existe un registro central de los agentes ni de sus competencias, no hay páginas blancas o amarillas, etc.).
- Los agentes tienen una larga vida.
- Los agentes son multi-hilo y pueden hacer varias tareas al mismo tiempo.
- Los agentes pueden unirse o salir de la plataforma en cualquier momento.
- Los agentes no pueden viajar de una máquina a otra.
- Los agentes se pueden organizar en grupos.
- Los agentes tienen habilidades.
- Los agentes tienen objetivos que son diferentes de las habilidades.
- Los agentes pueden comunicarse con el usuario a través de un agente asistente personal.
- Los agentes tienen ontologías.
- Los agentes pueden razonar.
- Hay un IDE para los agentes.

Los agentes, una vez creados, se quedan con vida hasta que alguien o algo los elimina. Los agentes son completamente libres de unirse y salir de un grupo de agentes en cualquier momento. Por lo tanto, dentro de un grupo de agentes se realiza una comunicación de bajo nivel utilizando el protocolo UDP , lo que significa que todos los agentes puedan recibir todos los mensajes. Sin embargo, el *middleware* filtra los mensajes, y entrega sólo aquellos dirigidos a un agente.

Los agentes OMAS tienen un conjunto de habilidades que corresponden a las tareas que un agente puede hacer. Una habilidad se programa como una función. Hay dos tipos de habilidades: una habilidad atómica y una habilidad compleja. Las habilidades atómicas son ejecutadas por el agente sin ayuda. Mientras las complejas generan subtareas para ser ejecutadas por otros agentes. Si un agente recibe un mensaje, pero no tiene la habilidad correspondiente, simplemente lo ignora. Los agentes tienen una ontología que puede ser ampliada, y se utiliza para crear instancias de una base de conocimientos. La ontología se debe formalizar mediante el lenguaje *MOSS*, que tiene su mecanismo de razonamiento.

El ACL especifica la estructura y las performativas de los mensajes que se pueden utilizar para comunicarse (tipo de mensajes). Además, los mensajes pueden ser enviados como punto-a-punto, *multicast*, *broadcast*, o *Contract-Net*. Las performativas de OMAS son algo diferentes de las de FIPA/KQML. En OMAS no está especificado el lenguaje de los contenidos. Por lo tanto, es responsabilidad del desarrollador de aplicaciones utilizar el lenguaje más apropiado.

OMAS ofrece tres tipos de agentes:

- Agentes de Servicio: son agentes regulares que proporcionan una serie de servicios. Se pueden considerar como servicios Web. Las diversas partes de la estructura de un agente de servicio son:

  - Interfaz de red, manejo de mensajes y registro.
  - Habilidades, que contienen los conocimientos correspondientes a las tareas que el agente puede hacer.
  - Mundo, que contiene un modelo del mundo, a saber, una descripción de otros agentes.
  - Tareas, un modelo de las tareas actuales activas en el sistema.
  - Ontología, que contiene la ontología y la base de conocimientos.
  - Auto, que contiene una descripción del mismo agente, sus habilidades, etc.

- Agentes Asistentes Personales: son agentes complejos que pueden interactuar con el usuario mediante un cuadro de diálogo en lenguaje natural. Ellos usan ontologías y un mecanismo de diálogo complejo. Los agentes asistentes personales actúan como mayordomos digitales y pueden ser programados en el idioma nativo del usuario.
- Agentes de Transferencia (carteros): se utilizan como enlace entre dos sitios remotos, o para la conexión de la plataforma con los servicios Web, o para interactuar con otras plataformas multiagentes. Por lo general utilizan un protocolo TCP.

Un IDE está disponible para propósitos de depuración, lo que permite examinar el contenido de los agentes y mostrar su comportamiento [32].

4.3.9. DECAF

DECAF (*Distributed*, *Environment-Centered Agent Framework*) es una plataforma de desarrollo de agente de propósito general que fue diseñado específicamente, para apoyar la concurrencia de operaciones distribuidas [51]. La arquitectura soporta SMA altamente distribuidos.

DECAF consiste en un conjunto de módulos bien definidos (de inicialización, para envío de mensajes, de planificación y programación, de ejecución y coordinación), que funcionan, en conjunto, para controlar el ciclo de vida de un agente. Cada uno de los módulos se implementa como un hilo separado en Java. También tiene una estructura de tarea representada por un núcleo, que se comparte entre todos los módulos de control. Su estructura permite que los desarrolladores de agentes se centren en la lógica de la tarea, para lo cual proveen un conjunto de herramientas: bibliotecas de componentes, IGU de programación, sistema de gestión del agente (AMA), *middleware* (servidor de nombres de agente, etc.).

DECAF proporciona los servicios de arquitectura necesarios para un agente inteligente para la comunicación, la planificación, el seguimiento de la ejecución, la coordinación, el aprendizaje y el auto-diagnóstico, que constituye el "sistema operativo" interno del agente.

Cuando se inicia un agente, el módulo de inicialización del agente se ejecuta. Después de la inicialización, cada módulo se ejecuta continuamente y al mismo tiempo en su propio hilo de Java. Al crearse el agente, se pasa el control al *distribuidor de mensajes* que espera un mensaje. Estos mensajes se colocan en la cola de mensajes entrantes. Un mensaje entrante contiene una performativa KQML y su información asociada. Un mensaje entrante puede resultar en dos acciones: es parte de una comunicación de una conversación en curso (en este caso, el despachador encuentra la acción correspondiente en la cola de espera de acciones y establece las tareas del agente a realizar), o es parte de una nueva conversación (se coloca en la cola de objetivos)

El planificador controla la cola de objetivos y define una cola de tareas para cada objetivo, con las acciones a realizar por el agente. El programador controla la cola de tareas. El propósito del programador es determinar qué acciones pueden ser ejecutadas, y en qué orden deben ser ejecutadas. El ejecutor se pone en funcionamiento una vez que una acción se coloca en la cola del ejecutor, colocando la acción en el tope de la cola para ser ejecutada inmediatamente. Una de dos cosas pueden ocurrir en ese momento: la acción puede completarse normalmente o una acción puede fallar. En ese último caso se indica la falla. El módulo ejecutor iniciará cada tarea en su propio hilo.

Algunas de las herramientas de desarrollo que se proponen son [51]:

- *Editor de Planes:* el archivo del plan se crea usando una IGU. DECAF proporciona una interfaz de programación de componentes de software, que permite la reutilización de componentes.
- *Construcción de Acciones:* permite un desarrollo muy rápido de acciones para los agentes. El desarrollador no tiene que escribir ningún código de comunicaciones, no tiene que preocuparse de las invocaciones, no tiene que analizar y programar los mensajes, para escribir y programar el agente.

El archivo del plan es un árbol que representa la programación de un agente, y cada nodo hoja del programa representa un procedimiento que el usuario debe escribir. El lenguaje DECAF admite todas las construcciones habituales en programación, tales como la selección y los lazos, además de "acciones persistentes estructuradas". También permite definir característica de la programación orientada a agentes de describir en términos generales el método para la consecución de un objetivo. En DECAF se utiliza el concepto de nube para representar una acción que no es local (no la ejecuta el agente). Siguiendo el paradigma de nubes del ámbito computacional, la nube se preocupa de los aspectos vinculados a las comunicaciones, de las entregas de los mensajes a los agentes, entre otras actividades. El programador debe solo procesar las respuestas.

Además, DECAF tiene herramienta de generación de prueba. El programa de generación de pruebas para DECAF permite la generación al azar de archivos de planes, creados con ciertas características, para ser pruebas de

diferentes aspectos. Con sólo unos pocos cambios en los parámetros, muchos tipos diferentes de archivos de plan pueden ser generados y utilizados. Por otro lado, el *middleware* permite el flujo de información en un SMA. Para ello tiene una serie de componentes:

- *Servidor de nombres de agentes* (ANS): es un programa autónomo con un API fijo que hace el registro de los agentes. Actividades sencillas como registrar, anular un registro y buscar agentes, pueden ser manejados por este servidor invocado desde conexiones de *socket* simples.
- *Agente para realizar mapeos*: sirve como una herramienta de páginas amarillas para ayudar a los agentes en la búsqueda de otros agentes en la comunidad, que puedan brindar un servicio útil para sus tareas. Un agente anuncia sus capacidades con él, y si esas capacidades cambian o ya no están disponibles. El agente de mapeo almacena las capacidades en una base de datos local. Un solicitante que desee hacer una consulta le formula la consulta pidiéndole los agentes que coincidan. Un agente también puede suscribirse a este agente, para ser informado cuando se agregan o se quitan servicios de interés.
- *Agente corredor*: que anuncia las capacidades de proveedores que se han registrado con él. El corredor, a su vez, puede anunciarse al agente que realiza mapeos.
- *Agente de extracción de información* (IEA): permite cumplir las solicitudes procedentes de fuentes externas, y el monitoreo de fuentes externas de información de manera periódica.
- *Agente proxy:* permite que cualquier agente pueda comunicarse con cualquier objeto que utilice KQML o constructos de mensajes FIPA. El agente proxy es un agente que utiliza direcciones fijas y *socket* para la comunicación con cualquier otra aplicación. A través del agente proxy, aplicaciones fuera del SMA y de DECAF tienen acceso a servicios del SMA.
- *Agente de administración de agentes* (AMA) crea una representación gráfica de los agentes actualmente registrados en ANS, así como de las comunicaciones entre dichos agentes. AMA también consulta al agente generador de mapeos para recuperar el perfil de cada agente. Este perfil contiene información acerca de los servicios prestados por un agente.

4.3.10. SEAGENT

SEAGENT es una plataforma basada en la Web semántica para el desarrollo de agentes [45]. La infraestructura de comunicación y ejecución de SEAGENT se parece a otros entornos de desarrollo de agentes existentes, como DECAF y JADE. Sin embargo, para apoyar y facilitar el desarrollo de SMA usando la Web semántica, SEAGENT incluye las siguientes características, que las otras plataformas de agentes existentes no tienen:

- Proporciona una función específica dentro de la arquitectura interna del agente para manejar el conocimiento, mediante OWL (*Web Ontology Language*).
- El directorio de servicios de SEAGENT almacena las capacidades de los agentes utilizando una ontología especial, basada en OWL, y proporciona un motor semántico de coincidencias para encontrar los agentes con capacidades relacionadas.
- Provee un lenguaje de contenido, denominado *Seagent Content Language* (SCL), basado en FIPA-RDF, para transferir el contenido semántico dentro de los mensajes.
- Introduce un nuevo servicio denominado Servicio de Gestión de Ontología (OMS). La característica más importante de este servicio es definir mapeos entre ontologías de la plataforma y ontologías externas. Además, se ofrece un servicio de traducción de ontologías a los agentes de la plataforma.
- Es compatible con el descubrimiento y la invocación dinámica de servicios Web semánticos, mediante la introducción de un nuevo servicio en la plataforma para el descubrimiento de servicios semánticos. Además, provee comportamientos de agentes reutilizables y la invocación dinámica de los servicios descubiertos.

La arquitectura de software de SEAGENT es en capas. Las capas, y sus paquetes, han sido especialmente diseñadas para apoyar el desarrollo de SMA en entornos de la Web Semántica. Es lo suficientemente general para permitir el despliegue de agentes en entornos Web semánticos. Los capas de SEAGENT son:

- *Capa de la infraestructura de comunicación*: es la capa inferior, responsable de abstraer la implementación de la infraestructura de comunicación de la plataforma. SEAGENT implementa la comunicación entre los agentes según FIPA. Aunque la capa de infraestructura de comunicación puede transferir cualquier contenido que esté basado en FIPA ACL, la plataforma SEAGENT soporta SCL (*Simple Common Logic*) de forma predeterminada. SCL es una ontología OWL específica para definir contenidos de ACL. Extiende FIPA-RDF mediante la definición de nuevos conceptos y relaciones. Esto permite que el contenido sea fácilmente interpretado, y se aprovecha de la inserción de conceptos/individuos de ontologías OWL, para formar la base de conocimiento de los servicios y agentes.
  Con el fin de ser usado con FIPA-ACL, el lenguaje de contenido satisface tres requisitos. Los dos primeros están relacionados a los estados que el lenguaje debe ser capaz de representar: proposiciones y acciones. Esto se hace en SEAGENT mediante la definición de estos dos conceptos en la ontología SCL. El tercer requisito es que debe ser capaz de representar objetos, incluyendo la identificación de expresiones referenciales para describir objetos. Para lograr esto, se han definido consultas en OWL en

la ontología llamada "*Agent Match Ontology*". Los conceptos de esta ontología se utilizan para definir el contenido de los mensajes.

- *Capa con el corazón de la plataforma*: la segunda capa incluye paquetes que proporcionan la funcionalidad principal de la plataforma. El primer paquete, llamado *Agency*, se encarga de la funcionalidad interna de un agente. Este permite la creación de agentes de propósito general y/o dirigido a objetivos. En este sentido, el paquete ofrece un "sistema operativo para agentes" que hace coincidir los objetivos con los plan predefinido, que se definen utilizando el formalismo *HTNplanning*. El programa ejecuta y supervisa los planes.
  Desde la perspectiva de la Web semántica, la arquitectura interna de un agente debe ser compatible con las normas de las ontologías de la Web semántica, para la mensajería y el manejo interno de conocimientos. Para ello, el paquete proporciona un soporte para analizar e interpretar el lenguaje de contenidos SCL que maneja la semántica de mensajería basada en Web. Por otro lado, *Agency* proporciona dos interfaces para el manejo del conocimiento semántico, uno para la gestión de la ontología local y el otro para las consultas. Aunque la versión actual soporta JENA (Herramienta de Web Semántica de código abierto para Java, que proporciona una API para extraer y escribir datos en grafos RDF), otros entornos de gestión de conocimiento semántico y motores de consulta se pueden integrar a la plataforma.
  El segundo paquete incluye servicios en sub-paquetes, uno para cada servicio de la plataforma. SEAGENT ofrece todos los servicios estándares de FIPA, incluyendo AMS y DF. Sin embargo, estos servicios estándares se implementan de forma diferente, usando las capacidades de la Web semántica. DF utiliza una ontología OWL para definir las capacidades de los agentes e incluye un motor semántico para buscar agentes. El proceso de mapeo se hace con el motor de inferencia *Seagent Matching Engine*. Este motor hace coincidir los servicios de los agentes con la solicitud de servicio recibida. Almacena definiciones de servicio de agente en una base de datos. En realidad, esta base de datos es un modelo ontológico de los servicios de los agentes. Por lo tanto, cada servicio de agente que se registra en el DF también está representado en esa ontología. *Seagent Matching Engine* utiliza un razonador básico, llamado *Ontolog*, para determinar la relación semántica entre los servicios de los agentes. *Ontolog* define un árbol de conceptos de servicios (jerarquía) y encuentra la distancia entre dos servicios dados (es decir, entre el servicio solicitado y el publicitado). Del mismo modo, AMS hace las descripciones de los agentes en OWL utilizando FIPA *Agent Management Ontology*, y puede ser consultado semánticamente y aprender descripciones de cualquier agente que actualmente resida en la plataforma.
  Además de la implementación de servicios estándares de manera semántica, la plataforma SEAGENT ofrece dos nuevos servicios para simplificar el desarrollo de SMA usando la Web semántica. El primero de ellos se lla-

ma *Semantic Service Matcher* (SSM), y es responsable de la conexión de la plataforma con los servicios Web alojados en el exterior de la plataforma. Para la publicación de los servicios, SSM se basa en el estándar Semantic Web Services (OWL-S). Este conocimiento es utilizado por el motor de inferencia semántico para el descubrimiento de servicios. El segundo servicio es el Servicio de Gestión de Ontologías (OMS). Se comporta como un repositorio central de ontologías de dominio utilizados dentro de la plataforma, y proporciona funciones básicas de gestión de ontologías para la actualización y consulta de éstas. Pero el aspecto más importante de OMS es que soporta la traducción entre ontologías. A través del uso del traductor entre ontologías, cualquier agente de la plataforma se puede comunicar con un SMA y/o servicios fuera de la plataforma, incluso si utilizan diferentes ontologías.

- *Capa de comportamientos reutilizables*: incluye planes genéricos. Estos planes están divididos en dos paquetes independientes del dominio, que pueden ser utilizados por cualquier protocolo de comunicación de agentes conocido (de subasta inglesa, holandesa, etc.). Uno de los paquetes, llamado, *Generic Semantic Behaviors*, incluye sólo los comportamientos relacionados con la Web semántica. En la versión actual, el comportamiento semántico genérico más importante es el que realiza el descubrimiento y la invocación dinámica de servicios externos. Este comportamiento utiliza los servicios de SSM para descubrir el servicio deseado, y a continuación invoca dinámicamente el servicio Web encontrado. Por lo tanto, los desarrolladores pueden incluir el descubrimiento dinámico de servicios externos y la capacidad de invocarlos en sus planes, simplemente insertando ese comportamiento reutilizable como una tarea en su definición del plan.

4.4. Plataformas para el despliegue de SMA

En esta sección se presentan algunas plataformas que se comportan como MGS (*middleware*) para SMA, las cuales proveen los mecanismos necesarios para que los mismos se puedan ejecutar.

4.4.1. MaDKit

MaDKit (*Multiagent Development Kit*) es una plataforma multiagente escrita en Java, licenciada bajo GNU GPL, que incluye [52, 53, 14, 47]:

- La creación y gestión del ciclo de vida de los agentes.
- Una infraestructura para la comunicación entre agentes.

- Una arquitectura de agentes heterogénea sin un modelo de agente predefinido.
- Herramientas de simulación multiagentes.

MaDKit no impone ninguna consideración sobre la estructura interna de los agentes, lo que permite a un desarrollador implementar libremente sus propias arquitecturas de agentes. Fue desarrollado en el LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier). Contrariamente a las otras plataformas que se presentan en este capítulo, MaDKit es, sobre todo, un motor de ejecución de SMA.

Con MaDKit se hace un diseño orientado a organizaciones, esta etapa incluye la definición del modelo de organización (grupos, roles), el modelo de interacción (protocolos, mensajes), y otras entidades específicas (tareas, objetivos, etc.).

MaDKit no ofrece un modelo de agente, tiene que ser implementado en Java desde el inicio, o modificarse para trabajar con MaDKit. Puesto que ninguna suposición se hizo sobre el modelo de agente, cualquier tipo de modelo se puede utilizar. Sin embargo, todo el código del agente tiene que ser implementado en Java. El despliegue de agentes MaDKit tiene lugar en el *G-box*, una especie de recinto de seguridad del agente, donde los agentes pueden ser creados, modificados y destruidos, con una interfaz gráfica. Varios *G-box* se pueden conectar para conseguir una distribución a través de una red. Un modo sencillo de consola también está disponible para simplificar el despliegue. El *G-box* permite la configuración dinámica de las características editables del agente,

La principal característica de MaDKit es que es, principalmente, un motor de ejecución multi-agente. Esto conduce a un desarrollo y despliegue simple, ya que la plataforma se centra en la infraestructura de los agentes. El principal defecto de MaDKit, es que la construcción de agentes requiere escribir su código, puesto que no hay modelo de agente preestablecido.

Así, MaDKit es un conjunto de paquetes de clases de Java que implementa el núcleo del agente, y las diversas bibliotecas de mensajes. Como se muestra en la Figura 4.12, la cual ha sido extraída de [14], MaDKit posee tres componentes: Un *micro-kernel* de agentes, un entorno gráfico y un conjunto de servicios.

El *micro-kernel de agentes* es un núcleo pequeño de agente optimizado. El núcleo está envuelto en un agente especial que permite el control y seguimiento del modelo de agente. Algunas de las tareas del micro-kernel son:

- *Gestión del ciclo de vida del agente:* El núcleo es responsable de la creación y la destrucción de los agentes, y de las tablas que mantienen las referencias a los agentes. Se asigna un identificador único global a cada agente a partir de su creación.
- *Manejo de los mensajes locales:* El kernel maneja el enrutamiento y distribución de los mensajes entre agentes locales.

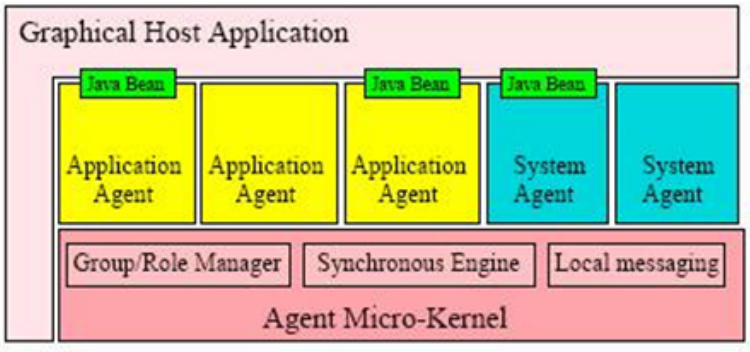

Figura 4.12: Componentes de MaDKit.

El desarrollador tiene la libertad de definir el comportamiento de los agentes. Existe una clase principal (*Control* and *life-cycle*) para un agente MaDKit, que define primitivas relacionadas con el pase de mensajes, además agrupa y administra funciones, pero no implementa una política de ejecución específica. El pase de mensajes en la plataforma MaDKit se definen por herencia de una clase de mensaje genérico. Una subclase se añade para soportar la ejecución basada en hilos concurrentes. Subclases adicionales implementan la ejecución sincrónica a través de un planificador externo. La comunicación se logra a través de mensajes asíncronos, con primitivas para enviar directamente un mensaje a otro agente.

MaDKit utiliza agentes para lograr la migración, seguridad dinámica, y otros aspectos de la gestión del sistema. Estos diferentes servicios están representados en la plataforma como funciones específicas, definidas en una estructura organizativa abstracta. Esto permite un alto nivel de personalización, ya que estos agentes de servicio pueden ser reemplazados sin dificultades.

Los agentes MaDKit pueden ser distribuidos de forma transparente sin cambiar nada en el código del agente. La característica distribuida de la plataforma de agentes se basa en dos funciones:

- *El agente comunicador:* utilizado por el micro-kernel para enrutar los mensajes no locales a otros agentes comunicadores de plataformas lejanas.
- *La gestión de grupo de agentes:* permite distribuir a agentes y funciones en grupos, como también el envío de mensajes. Los grupos se pueden generar de diferentes tamaños.

La arquitectura gráfica de MaDKit se basa en componentes gráficos independientes, utilizando Java Beans [53, 14]. Cada agente es el único responsable de su propia interfaz gráfica. Una interfaz de agente puede ser una simple etiqueta, una construcción compleja basada en múltiples *widgets* u

otra estructura gráfica. Un intérprete inicia el kernel y configura las interfaces de los distintos agentes.

4.4.2. SPADE

SPADE (*Smart Python multi-Agent Development Environment*) es una plataforma libre de SMA, desarrollada en Python en 2005 en la Universidad Politécnica de Valencia, basada en la mensajería instantánea XMPP [46, 27]. SPADE es compatible con FIPA, y es independendiente del lenguaje y la plataforma. Usa como protocolo de transporte la mensajería instantánea XMPP (*eXtensible Messaging and Presence Protocol*), basada en XML y varias extensiones desarrolladas para el mismo. La comunicación entre los agentes permite introducir funciones como la gestión de la lista de contactos de los agentes, la posibilidad de recibir una notificación, cada vez que cualquier otro agente conocido lo requiera (incluido en su lista de contactos), y facilita la participación de manera simultánea en multi-conferencia. La plataforma SPADE está construida alrededor del marco de comunicación *Jabber* [48]. El núcleo de la plataforma está diseñado sobre esta capa, que se comunica con los otros elementos de la plataforma. Las principales características de SPADE son:

- Es compatible con el estándar FIPA: AMS y DF son soportados.
- Admite dos protocolos de transporte de mensajes: HTTP y XMPP. Esta característica le permite comunicarse con diferentes plataformas.
- Admite dos lenguajes diferentes: FIPASL y RDF, que son las dos principales opciones para los lenguajes de contenido utilizados en la comunicación entre agentes.
- Soporta la movilidad de los agente. Un agente SPADE puede pasar de una plataforma a otra plataforma autorizada, y continuar su ejecución en el nuevo sitio.
- La plataforma proporciona autenticación de usuario y contraseña para mejorar la seguridad, así como una conexión cifrada para la confidencialidad.
- Tiene un agente especial para proporcionar una visión de la plataforma. Esta IGU proporciona un mecanismo para cargar y descargar los agentes en la plataforma. Permite al usuario buscar agentes y servicios, y enviar mensajes a cualquier agente.
- Fue desarrollado en Python. Esto permite la ejecución de la plataforma en varias arquitecturas y sistemas operativos, como Windows, Linux, MacOS, Windows Mobile, PalmOS, SymbianOS para teléfonos móviles, etc.
- Una de las características potentes de SPADE es la posibilidad de utilizar la notificación de presencia entre los componentes. Esto le permite al sistema determinar, en tiempo real, el estado actual de los componentes que están conectados a la plataforma.

- La gestión de una conversación con más de un agente es posible gracias a un mecanismo de *Jabber*.
- El protocolo utilizado para enviar mensajes es un protocolo cliente-servidor distribuido. Los servidores que enrutan los mensajes de *Jabber* están diseñados para soportar un gran número de usuarios y mensajes.

Las características de la interfaz son:

- La plataforma y los agentes exportan su propia interfaz Web.
- La interfaz es configurable a través de plantillas.
- Cada agente exporta su identificador como un código de respuesta rápida (QR: *quick response code*).
- Existe un agente *Instrospector* para analizar el comportamiento de los agentes.
- Tiene un visor de los mensajes enviados.
- Posee un servicio de búsqueda de agentes.

El modelo de agente de SPADE es el siguiente: cada componente en el interior de la plataforma es un agente. Agentes SPADE pueden enviar mensajes unos a otros y a otras plataformas. De esta manera, el modelo de agentes se compone básicamente de un mecanismo de conexión a la plataforma, un distribuidor de mensajes, y un conjunto de diferentes tareas que el despachador da a los mensajes (ver Figura 4.13, extraída de [27])). Cada agente necesita un identificador llamado Jabber ID (*JID*) y una contraseña válida para establecer una conexión con la plataforma. Un agente puede ejecutar varias tareas simultáneamente. Una tarea es un proceso que un agente puede ejecutar usando patrones de repetición. SPADE proporciona algunos tipos de tareas predefinidas: cíclicas, una vez, periódicas, definidas como máquinas de estados finitas. Cada agente puede tener tantas tareas como lo desee. Cuando llega un mensaje al agente, el despachador de mensajes lo coloca en la cola de tarea correcta (ver Figura 4.13).

4.4.3. Magentix

Magentix es una plataforma para la gestión de agentes que usa los servicios proporcionados por el sistema operativo [30, 15]. Por lo tanto, una de las decisiones de diseño es que está escrito en C sobre el sistema operativo Linux. Los enfoques actuales de desarrollo más comunes se basan en lenguajes interpretados como Java o Python. Estos diseños permiten construir, por ejemplo, *middlewares* sobre la Máquina Virtual Java. Aunque estos *middlewares* ofrecen algunas ventajas como la portabilidad y la facilidad de desarrollo, el uso de servicios de sistemas operativos (OS) para desarrollar plataformas de soporte a agentes, en lugar de utilizar *middlewares*, mejora notablemente el rendimiento y la escalabilidad. En este caso, la plataforma

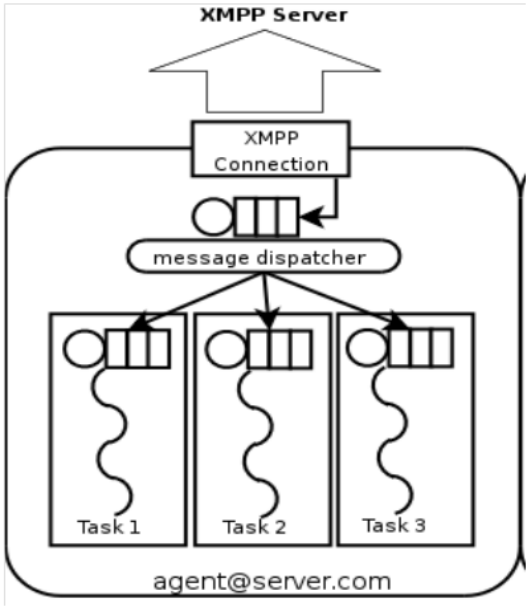

Figura 4.13: Un agente en SPADE.

de agentes se puede ver como una extensión de la funcionalidad ofrecida por un sistema operativo.

En particular, Magentix es un sistema distribuido compuesto por un conjunto de equipos ejecutando Linux. Se replica la información en cada sitio para lograr una mejor eficiencia. Cada uno de los sitios presenta una estructura de árbol de procesos. La ventaja de la gestión del árbol de procesos ofrecido por Linux, y el uso de algunos de sus servicios, como señales, memoria compartida y hilos de ejecución, proporcionan un escenario adecuado para el desarrollo de una plataforma de agentes robusta, eficiente y escalable.

En el caso concreto de Magentix, su servicio de comunicación ha sido desarrollado para ofrecer un alto rendimiento. Magentix proporciona mecanismos avanzados de comunicación, tales como manejo de grupos de agentes, administrador para ejecutar protocolos de interacción, así como un mecanismo de seguridad para proporcionar control de autenticación, integridad, confidencialidad y control de acceso.

Así, la estructura de cada sitio Magentix es un árbol de procesos de tres niveles. En el nivel superior esta el proceso principal. Este proceso es el primero lanzado en cualquier sitio cuando se agrega a la plataforma. Por debajo de este nivel está el de servicios. Magentix proporciona algunos servicios para apoyar la ejecución del agente: AMS, un DF y un Director de la Unidad Organizacional (OUM). Los servicios están representados por medio de agentes de servicios, replicados en cada sitio. Agentes que representan el

mismo servicio de gestión de la información, replican y se comunican entre sí con el fin de mantener la información actualizada. Por último, en el tercer nivel, están los agentes de los usuarios.

Con esta estructura de árbol de procesos, el proceso principal maneja los agentes de servicio, es decir, que puede finalizar a cualquier agente de servicio para un apagado controlado de la plataforma, y también detecta de inmediato si algún agente de servicio no está activo. De la misma manera, el agente AMS tiene un amplio control de los agentes de los usuarios de su sitio. Cada agente de usuario está representado por un proceso hijo Linux diferente del AMS, que se ejecuta en el mismo sitio.

Magentix ofrece soporte para diferentes distribuciones de Linux (como Ubuntu, Fedora, etc.), así como para MacOS. La interoperabilidad entre agentes heterogéneos se alcanza por medio del uso de un lenguaje de comunicación estándar y ontologías para las interacciones.

El OUM permite la comunicación entre-grupos de agentes. Esto es útil para el desarrollo de aplicaciones que utilizan metodologías de desarrollo orientadas a la organización de agentes. Un grupo de agentes en Magentix se llama unidad organizativa (en adelante, la unidad), y puede ser visto como una caja negra desde el punto de vista de los agentes externos. Las unidades también pueden estar compuestas de unidades anidadas. Los agentes pueden interactuar con un agente de la unidad o con una unidad de una manera transparente. Cada unidad tiene algunas propiedades asociadas a la misma. Cada agente tiene un nombre único, igual que cada unidad. Con el fin de interactuar con cualquier unidad, el usuario debe especificar uno o más agentes de la unidad: estos agentes se denominan agentes de contacto. El usuario también puede especificar la forma en que los mensajes tienen que ser entregados a los agentes de contacto. Así, los mensajes dirigidos a la unidad serán entregados a los agentes de contacto de diversas maneras:

- *Unidifusión*: los mensajes dirigidos a la unidad se entregan a un solo agente que es responsable de recibir mensajes.
- *Multicast*: varios agentes pueden ser designados para recibir mensajes.
- *Round Robin*: cada mensaje dirigido a la unidad se entrega a un agente de contacto diferente, definido de acuerdo con una política circular.
- *Aleatoria*: varios agentes pueden ser definidos como agentes de contacto. Sin embargo, el mensaje se entrega a uno, de acuerdo con una política al azar.
- *Sourcehash*: varios agentes pueden ser los agentes de contacto. Sin embargo, cualquier mensaje dado se entrega a uno de los agentes responsables de la recepción de mensajes, de acuerdo con el sitio donde está situado el remitente del mensaje.

Por último, cada unidad tiene un gestor asociado a la misma. Este agente es responsable de añadir, eliminar o modificar los miembros y agentes de contacto. Por omisión, es el agente que crea la unidad, y es el único permitido para eliminarla. Toda esta información sobre las unidades en Magentix

es gestionada por el servicio OUM. OUM es un servicio distribuido, compuesto por agentes OUM que se ejecutan en cada sitio. La interacción entre los agentes y el servicio OUM se lleva a cabo mediante el envío de mensajes, usando la ontología del servicio OUM.

Para representar la información, Magentix usa RDF (*Resource Description Framework*) , un lenguaje de descripción de recursos en la Web. Generalizando el concepto de un "recurso de la Web", RDF también se puede utilizar para representar información acerca de otros elementos, incluso cuando no se pueden recuperar directamente de la Web. RDF se basa en la idea de realizar las descripciones utilizando identificadores Web, llamados identificadores uniformes de recursos (URI); y los recursos en términos de propiedades simples. La estructura de cualquier expresión en RDF es una colección de tripletas, cada una consta de un sujeto, un predicado y un objeto. El sujeto puede ser cualquier recurso, el predicado es una propiedad del sujeto, y el objeto denota el valor de esa propiedad. Al conjunto de tales tripletas se llama un grafo RDF.

Usando las características de RDF, un marco para la gestión de la información ha sido diseñado para Magentix. Así, se permite que un agente Magentix gestione toda su información como grafos RDF. El marco se basa en ofrecer una API para la gestión de RDFs. Está diseñado como un contenedor (*wrapper*) para bibliotecas existentes de gestión RDF. Magentix usa *Redland*, un conjunto de bibliotecas en C, de software libre, que proporcionan soporte para RDF. Permite la manipulación de grafos y tripletas RDF, URI, etc.

Una de las funcionalidades de Magentix, donde RDF es utilizado, corresponde a la representación de mensajes. Los agentes usan RDF para describir el contenido del mensaje. El encabezado y el contenido del mensaje en Magentix se representan como modelos RDF serializados, como en XML. Por lo tanto, sólo se necesita un analizador, lo que simplifica el proceso de análisis y serialización de los mensajes.

En cuanto a la representación de la información de los servicios Magentix, una ontología para interactuar con ellos ha sido definida usando OWL. La ontología se centra, principalmente, en la descripción de los recursos que gestionan los servicios (sitios, agentes, servicios, unidades organizativas, etc.). Por lo tanto, un cambio en la implementación no tiene ningún efecto sobre la forma en que los servicios son invocados. Para lograr interacciones ricas y flexibles entre los agentes y los servicios Magentix, la ontología también incluye acciones que pueden ser solicitadas a un servicio Magentix (creación de un nuevo agente al AMS, registro de un servicio al DF, creación de una nueva unidad al OUM, etc.). De esta manera, cualquier agente Magentix que conoce la ontología puede interactuar con los servicios.

El modelo de seguridad de Magentix se basa en el protocolo *Kerberos* [31], y el mecanismo de control de acceso de Linux. Este modelo proporciona autenticación, integridad y confidencialidad. Por medio de este modelo, cada agente tiene una identidad que puede mostrar al resto de agentes y servicios de Magentix. Los agentes Magentix pueden tener tres tipos de identidad:

- *La identidad del agente*. Esta identidad la crea el AMS cuando instancia al agente.
- *La identidad del usuario*. La identidad de su propietario, es decir, la identidad del usuario que creó el agente.
- *La identidad de la unidad*. La identidad de la unidad donde el agente está.

Un agente siempre tiene, al menos, su identidad y la del propietario. Por lo tanto, un agente Magentix está provisto de más de una identidad, lo cual puede ser usado para gestionar la seguridad del sistema. Detalles de ese proceso se pueden ver en [15].

Los agentes Magentix se representan como procesos Linux. Internamente, cada agente se compone de hilos de Linux: un solo hilo de ejecución del agente (hilo principal), un hilo para el envío de mensajes (hilo emisor) y un hilo para recibir mensajes (hilo receptor).

Magentix proporciona una plantilla para el desarrollo de agentes, escrita en C++. Ofrece diferentes métodos para gestionar el ciclo de vida del agente, así como el envío y recepción de mensajes. La interacción con servicios se lleva a cabo por medio de una API específica.

Para la gestión de las conversaciones, Magentix proporciona dos funcionalidades:

- *Buzón de correos*: se utilizan para la gestión de los mensajes entrantes al agente. De manera predeterminada, cada agente tiene un buzón único, llamado buzón de correo predeterminado, que recibe cada mensaje dirigido a éste. Magentix permite a los desarrolladores crear nuevos buzones y asociar un identificador de conversación a ellos. Entonces, cuando se recibe un mensaje, con este identificador de conversación, este mensaje se enruta al buzón correspondiente.
- *Administración de conversaciones*: las interacciones entre los agentes Magentix se centran en las conversaciones. Por lo tanto, es importante no sólo el envío de mensajes entre agentes, sino también reproducir los patrones de conversación. Cada interacción entre un par de agentes, puede requerir el intercambio de más de un mensaje. Por otra parte, los patrones de intercambio de mensajes se repiten en varias interacciones: acceder a algunos servicios, solicitar información, enviar propuestas, etc. Así, la definición de patrones de comunicación, para especificar los intercambios de mensajes en una interacción específica, es una característica interesante y útil para los desarrolladores de agentes. FIPA define varios protocolos de interacción para las conversaciones; para procesar esos protocolos de interacción, Magentix define un gestor de conversación. Cuando un agente está utilizando uno de estos protocolos, el gestor de la conversación es el encargado de administrarlo automáticamente, y garantizar su correcta ejecución. Varios gerentes de conversación se pueden activar en un momento dado para un agente, cada uno a cargo de la gestión de los diferentes protocolos de interacción donde participa. El gerente de la conversación es una abstracción que oculta los conceptos básicos de las

conversaciones, tal que sólo se tiene que especificar lo que debe hacerse en cada paso del protocolo, lo que permite fácilmente la ejecución concurrente y la gestión de varias conversaciones.

4.4.4. Plataforma basada en CALM

En [63] proponen una plataforma que permite la gestión de grupos de agentes individuales. La plataforma puede ser utilizada como base para SMA dinámicos dirigidos a servicios ubicuos. La plataforma de agentes incluye funciones importantes para la gestión de los agentes basada en FIPA (por ejemplo, AMS y DF están registrados en la plataforma como agentes).

Es una plataforma de agentes flexible y escalable basada en la composición de contenedores con varios grupos de agentes jerárquicos, como JADE. Sin embargo, se puede gestionar a los agentes de manera más eficiente por la composición de los contenedores en grupos con jerarquías múltiples de agentes. Además, cuenta con clases y funciones que permiten la comunicación entre agentes en diversos lenguajes: Java, C++, etc. Es compatible con JADE. Los SMA desarrollados con esta plataforma de agentes son flexibles, ya que soportan tanto la gestión de grupos como de agentes individuales al mismo tiempo. Tiene funciones esenciales para la comunicación de los agentes e incluye funciones de vigilancia de agentes. Además, permite la reflexión para la auto-configuración. La plataforma ha sido implementada en CALM (*Component-based Autonomic Layered Middleware*), desarrollado para entornos ubicuos.

La Figura 4.14 muestra la plataforma de agentes [63], que incluye el módulo con los protocolos de transporte de mensajes (MTP) para el procesamiento de ellos. También, contiene el módulo responsable de la gestión de la plataforma de agentes. El módulo de MTP se construye como sigue:

- *Comunicación HTTP*: permite la comunicación entre los agentes usando HTTP. Un agente tiene un servidor y un cliente HTTP. Cuando se recibe un mensaje, el servidor HTTP lo procesa. Cuando se envía un mensaje se utiliza al cliente.
- *Codificador/decodificador ACL*: es un módulo responsable del manejo de mensajes ACL. Un agente puede enviar un mensaje simultáneamente a varios agentes.
- *Servicio de mensajería*: gestiona la cola de mensajes, permitiendo recibir/enviar mensajes desde/a varios agentes al mismo tiempo.

Los principales módulos del núcleo de la plataforma son:

- *Contenedor principal*: gestiona la plataforma de agentes, utilizando la biblioteca MTP.

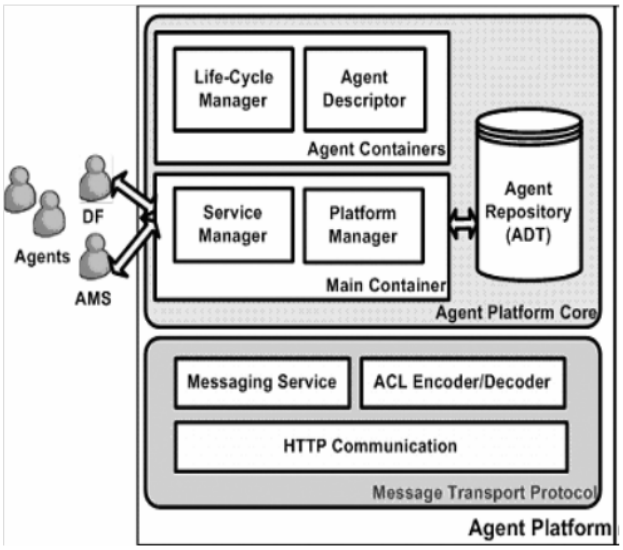

Figura 4.14: La estructura de la plataforma de agentes basada en CALM.

- *Administrador de la plataforma*: es un submódulo que se encarga de la inicialización de la plataforma y la entrega de mensajes desde el MTP al módulo de procesamiento del agente.
- *Administrador de servicios*: submódulo que registra y gestiona los servicios adicionales, tales como un módulo de reflexión.
- *Contenedor de agentes*: es un módulo que gestiona los grupos de agentes que tienen funciones jerárquicamente similares.
- *Administrador del ciclo de vida*: es un módulo que gestiona y controla la actividad de un agente en un hilo. Los estados son iniciar, suspender, reanudar, detener, etc.
- *Descriptor de agentes*: es el módulo de gestión de la información de descripción de los agentes, tales como la identificación, la dirección, etc. Posee una tabla *hash* llamada ADT (*Agent Description Table*), para la gestión de la descripción de los agentes, usando como clave el identificador del agente.

El núcleo de la plataforma regula las relaciones entre los módulos y servicios, a través del flujo de acción principal. La Figura 4.15, tomada de [63], muestra la secuencia entre los principales módulos de la plataforma principal, en el caso de inicialización de la plataforma de agentes, el registro y la liberación de un agente, y el envío de mensajes.

Cuando la plataforma de agentes comienza a operar, crea primero un contenedor principal. El contenedor principal lee el archivo de configuración que está en formato XML para la inicialización, que tiene la información

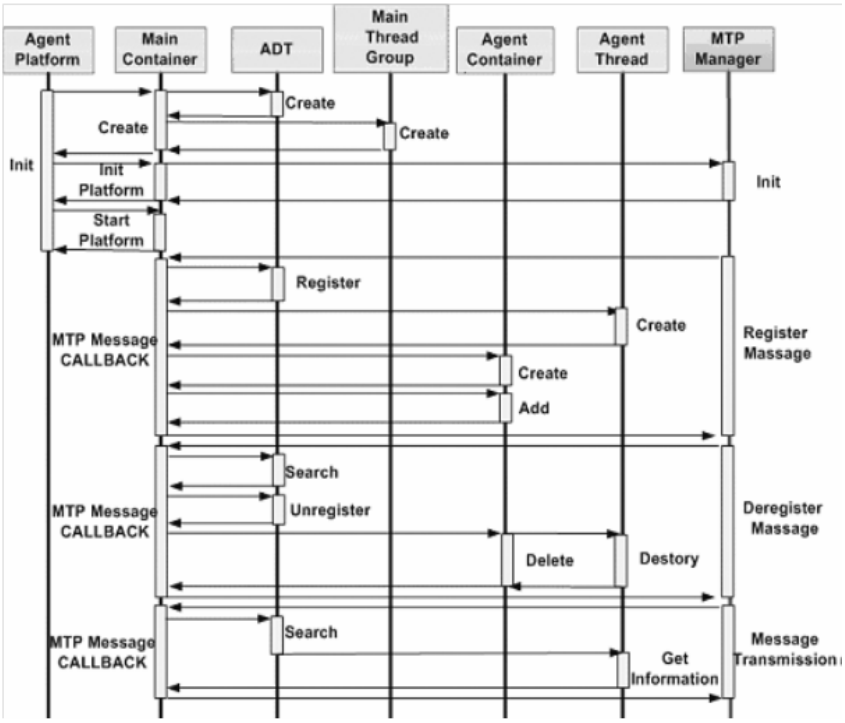

Figura 4.15: Secuencias principales de la plataforma basada en CALM.

sobre la versión de la plataforma, recursos disponibles, etc. También crea una ADT para acceder directamente a la información de descripción de los agentes. Entonces, termina la inicialización al enviar un mensaje ACL de inicialización al módulo de servidor de comunicaciones MTP.

Cada agente envía un mensaje de ACL para registrarse en la plataforma de agentes. La plataforma, después de comprobar que el contenido del mensaje se trata de un mensaje de registro, extrae el nombre del agente, dirección, información de descripción, información sobre el contenedor, etc. A continuación, examina en la ADT si ha sido ya registrado o no. Si ya se ha registrado, la plataforma envía un mensaje de error. Después de que el agente se ha registrado en el ADT, la plataforma de agentes ejecuta un hilo. El hilo agente tiene la información de descripción del agente, y supervisa de forma dinámica al agente. A continuación, comprueba si ya existe el contenedor de agentes. Si no, se crea uno, de lo contrario se agrega al contenedor de agentes existente.

Cuando un agente envía un mensaje de destrucción, el núcleo de la plataforma de agente recibe un mensaje a través de ACL. Entonces confirma la información, lo busca en el ADT, y elimina la información del agente corres-

pondiente. Asimismo, destruye el hilo y suprime la referencia al agente de la tabla del contenedor del agente.

A diferencia de JADE, el núcleo de la plataforma de agentes propuesto prevé múltiples dominios de grupos, incluso jerárquicos. Eso le hace tomar mucho tiempo para encontrar la información de los agentes. Por otro lado, el hilo de cada agente se convierte en un agente virtual que tiene un estado y funciona independientemente. El núcleo de la plataforma importa la biblioteca MTP y los módulos principales consisten en clases. MTP consta de la biblioteca *DllQueuing* para el procesamiento de las colas de mensajes, *NetworkLib* para la comunicación HTTP y *ACL Parser DLL* para el procesamiento de mensajes ACL.

4.4.5. TinyMAS

La plataforma *Multiagente Tiny* es una plataforma ligera, que permite implementar y ejecutar sistemas basados en agentes. Esta plataforma ha sido escrita por S. Galland y N. Gaud para los cursos multiagentes del Departamento de Ciencias de la Computación de la Universidad de Tecnología de Belfort [28]. La plataforma *Multiagente Tiny* contiene los siguientes componentes:

- *org.arakhne.tinyMAS.core*: implementa el núcleo de la plataforma incluyendo un planificador, páginas blancas, páginas amarillas, y los agentes.
- *org.arakhne.tinyMAS.network*: una extensión de la plataforma para soportar núcleos distribuidos en una red.
- *org.arakhne.tinyMAS.situatedEnvironment*: una extensión de la plataforma que proporciona funciones para entornos situados: medio ambiente, percepción.
- *org.arakhne.tinyMAS.demo*: ejemplos que ilustran varias características de la plataforma *tinyMAS*.

4.4.6. TrustMAS

TrustMAS (*Trusted Communication Platform for Multi-Agent Systems*) proporciona una plataforma para agentes móviles que garantiza confianza y anonimato [65, 64]. La plataforma incluye técnicas de anonimato, basadas en un algoritmo de paseo aleatorio [65], para ser usadas por los agentes, garantizando comunicaciones anónimas. Todos los agentes que intervienen en la plataforma se benefician de la confianza y el anonimato que se proporciona para sus interacciones. También, TrustMAS provee la capacidad de intercambiar información mediante el uso de canales ocultos. Además, es posible implementar agentes moviles.

TrustMAS da la oportunidad de construir una comunidad de agentes. En este tipo de entornos, como en la sociedad humana, la confianza y el anonimato se conviertan en aspectos importantes, puesto que permiten a los agentes crear y gestionar sus relaciones. Por un lado, TrustMAS permite la gestión de la información oculta. Por otra parte, con el fin de minimizar la incertidumbre de las interacciones, TrustMAS usa un esquema de niveles de confianza entre los agentes. Así, cada agente se adhiere a los mecanismos de confianza y anonimato de la plataforma TrustMAS que desee. En particular, el modelo de confianza se basa en el comportamiento específico de los agentes: después de un proceso de diálogo que significa que los agentes son de confianza. Otros modelos de confianza pueden ser incluidos.

En general, en TrustMAS existen dos grupos de agentes:

- Agentes ordinarios (OA), que utilizan la plataforma para beneficiarse de los dos servicios de seguridad (confianza y anonimato).
- Agentes que poseen la misma funcionalidad básica que OA, con capacidades adicionales para usar canales secretos.

4.4.7. JAS

JAS (*Java Agent Services*) define una especificación estándar y un API para el despliegue de una infraestructura orientada a servicios de agentes [16, 17]. Se trata de una implementación, por la comunidad de Java, de la *FIPA Abstract Architecture*. La intención de JAS es servir de base para la creación de aplicaciones de calidad comercial basadas en las especificaciones FIPA. En concreto, proponen una API de Java para la implementación de arquitecturas abiertas, que soporten *plugins* de plataformas de servicios de terceros. La API provee interfaces para la creación de mensajes, codificación de mensajes, transporte de mensajes, directorios y nombramientos. Además, JAS proporciona algunas API para RMI, LDAP y HTTP. Este diseño garantiza que las implementaciones de sistemas basados en JAS sean transparente a los cambios en las tecnologías subyacentes, sin causar interrupciones en la prestación de servicios.

4.4.7.1. UMAP

UMAP (*Universal Multi-Agent Platform*) es una plataforma basada en JADE, que proporciona un conjunto de soluciones listas para usarse en el desarrollo de agentes, dejando a los programadores la tarea de implementar la lógica de negocio de los agentes [18]. La plataforma UMAP fue desarrollada en el Instituto de Informática de la Universidad Politécnica de Silesia, Polonia, usando Microsoft .NET Framework, y se hizo disponible a partir del 2010.

El diseño de la plataforma UMAP sigue las normas de FIPA, permitiendo la creación de soluciones de software personalizadas,que se basan en la cooperación de muchos agentes. Proporciona un esqueleto de agente, que constituye una base para cualquier desarrollo. El Agente UMAP tiene una interfaz fija que permite la comunicación y la gestión dentro de la plataforma.

UMAP se distribuye como una aplicación de *Microsoft Windows*, que se utiliza para iniciar y administrar agentes. El proceso de instalación es rápido, automático y se realiza mediante el programa de instalación disponible en el sitio Web de UMAP.

Además de la aplicación de *Microsoft Windows* ("Sistema de Gestión de UMAP"), que permite ejecutar la plataforma, durante la instalación se crea la biblioteca *UMAP.Core.dll*. Esta biblioteca es necesaria para el desarrollo de agentes personalizados. Contiene la clase base de un agente y todos los componentes de la plataforma de gestión de SMA. El objetivo principal de UMAP es permitir a los usuarios crear sus propios sistemas, utilizando la misma plataforma multiagente. Por lo tanto, un elemento fundamental es la interfaz común para la clase *Agent*.

En UMAP, un agente está representado por la clase abstracta *Agent*, que define cuatro métodos: la creación de un agente se reduce a la aplicación de la lógica de funcionamiento cuando se reciben mensajes (método *handleMessage()*) y, si es necesario, se ejecuta el trabajo del agente (método *run()*). Opcionalmente, es posible definir el comportamiento del agente en el momento de la creación (método *OnActivate()*) y se puede interrumpir su ejecución (método *OnDeactivate()*).

La interfaz de agente está diseñada para permitir diversos esquemas de funcionamiento: como un programa que se ejecuta en segundo plano, como una aplicación de *Microsoft Windows* o como un programa que encapsula una aplicación existente. El diseñador de la lógica del agente no tiene que centrarse en cómo implementar la comunicación entre los agentes, ni cómo monitorear su trabajo, enfocándose en la lógica del sistema que desarrolla. Los agentes de software, creados en UMAP, se distribuyen y se ejecuta como piezas .NET. Cada pieza contienen una referencia a la clase *agent*. Esto permite la identificación de agentes y su ejecución bajo el control del contenedor de UMAP. La implementación de un agente se puede realizar en diversos lenguajes de programación, entre éstos, C++ y Visual Basic.

Debido a que UMAP se ha desarrollado utilizando la plataforma *Microsoft .NET Framework 2.0*., los sistemas que se construyen con UMAP pueden trabajar en los sistemas operativos que soportan esta tecnología.

4.5. IDE para agentes

Como es bien sabido, un IDE es un sistema informático compuesto por un conjunto de herramientas de programación. Un IDE puede dedicarse, en exclusiva, a un solo lenguaje de programación o bien puede utilizarse para varios.

Por lo general, un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de IGU. Por consiguiente, un IDE pueden ser una aplicación por sí sola, o pueden ser parte de aplicaciones existentes.

En este contexto, en esta sección se presentan algunos IDEs para desarrollar agentes.

4.5.1. Amine

Amine es una herramienta multi-capa Java, de código abierto, para el desarrollo de sistemas inteligentes y SMA [19, 55, 54, 56]. Provee un entorno de desarrollo integrado modular. Está compuesto por cuatro niveles jerárquicos:

- *La capa ontológica*: provee estructuras, procesos e interfaces gráficas para especificar el "vocabulario conceptual" y la semántica de un dominio.
- *La capa algebraica*: construida en la parte superior de la capa ontológica, proporciona "estructuras de datos" (*AmineSet*, *AmineList*, *Term*, etc.), operaciones sobre esas estructuras, e interfaces gráficas para definir y utilizar las estructuras y operaciones.
- *La capa de programación*: se construye en la parte superior de la capa algebraica, provee paradigmas/lenguajes de programación para definir y ejecutar procesos. Tres paradigmas de programación son posibles de usar: a) paradigma de patrones y basado en reglas de programación, incrustado en *PROLOG+CG*; b) paradigma de programación basado en la activación y propagación, incrustado en el lenguaje *SYNERGY*, y c) paradigma de programación basado en ontologías, que se ocupa de la integración automática del conocimiento en una ontología.
- *La capa multiagentes*: proporciona *plugins* para el desarrollo de agentes. Se puede utilizar en conjunción con un entorno de desarrollo Java.

Amine no provee el nivel básico para el desarrollo de SMA (es decir, las capacidades de comunicación de los agentes), ya que este nivel es ofrecido por otros proyectos de código abierto (como JADE). Amine ofrece plugins que permiten su uso y varias IGU para editar ontologías.

La plataforma Amine se puede utilizar como un entorno modular integrado para el desarrollo de sistemas inteligentes.

Amine propone usar a JADE para la gestión de la comunicación entre los agentes, de acuerdo con las especificaciones FIPA. De esta manera, JADE provee el manejo del nivel inferior de los SMA: creación y comunicación entre los agentes, mientras que Amine se encarga del nivel más alto: capacidades cognitivas y reactivas de los agentes. La capa SMA de Amine contiene un *plugin* implementado como un paquete: *AmineJade*. Este paquete ofrece dos clases:

- *PPCGAgent*, clase que amplía la clase *Agent* proporcionada por JADE con un intérprete para *Prolog+CG* y con otros atributos y métodos (ejemplo, *sendAndWait*).
- *JadeMAS*, provee métodos para crear y poner en marcha un SMA.

4.5.2. AgentTool

AgentTool es una herramienta basada en la metodología MASE (*Multiagent Systems Engineering*), que busca automatizar el proceso de diseño. Es independiente de la arquitectura multiagente particular, de la estructura de cada agente, del lenguaje de programación y del marco de comunicación. AgentTool provee soporte a tres de los siete pasos de la metodología MASE [20, 44, 43]:

- *Creación de clases*: se identifican las clases de los agentes desde los roles. Al igual que con los objetivos y roles, por lo general, se define una relación uno a uno entre los roles y las clases de agentes. Sin embargo, el diseñador puede combinar varios roles en una sola clase de agentes. Dado que los agentes heredan también conversaciones, a menudo es deseable combinar dos roles que comparten un alto volumen de tráfico de mensajes.
- *Construir conversaciones*: habitualmente se ejecuta en paralelo con el paso siguiente (montaje de agentes). Los dos pasos están estrechamente vinculados ya que la arquitectura de agente, definida en Montaje de agentes, debe implementar las conversaciones y los métodos definidos en la construcción de conversaciones. Una conversación en MASE se define como un protocolo de coordinación entre dos agentes. La comunicación es un par de máquinas de estados finitos que definen una conversación entre dos clases de agentes participantes.
- *Montaje de los agentes*: en este paso se crean las interioridades de las clases de agentes. Este proceso se ha simplificado mediante el uso de un lenguaje de modelado de arquitecturas, que combina lenguajes de descripción de arquitecturas con lenguajes de restricciones de objetos.

La parte de AgentTool más atractiva es la capacidad de trabajar en diferentes partes del sistema y en varios niveles de abstracción indistintamente,

lo que refleja la capacidad de diseño incremental de MASE. La operación "*TabbedPane*" de AgentTool implementa esta capacidad, el paso en que se está trabajando es representado por el esquema actual, y se dispone de acciones para moverse hacia atrás o hacia adelante a través de la metodología.

La construcción de un SMA utilizando AgentTool comienza con un diagrama de clases del agente. Una conversación sólo puede existir entre clases de agente, por lo que se añaden las clases de los agente antes que las conversaciones. Así, la construcción de conversaciones en AgentTool se hace una vez que se han definido las clases de los agentes. AgentTool provee mecanismos de verificación de las conversaciones, con los que se puede garantizar conversaciones sin errores.

El montaje de los componentes de un agente en AgentTool consiste en permitir que sus componentes internos se pueden agregar, quitar y manipular. Todos los componentes de un agente pueden tener diagramas de estado de componentes asociados, y subcomponentes adicionales por debajo de ellos.

Finalmente, AgentTool soporta la generación automática de código.

4.5.3. SIMBA

SIMBA (SIstema Multiagente Basado en Artis) es una plataforma para el desarrollo de SMA en tiempo real [37]. Esta arquitectura es una extensión del enfoque de arquitectura de agentes *Artis* para entornos de tiempo real. Básicamente, SIMBA se puede considerar como un conjunto de agentes *Artis* y sus interacciones, que proporciona servicios de tiempo acotado.

4.5.4. AgentScope

AgentScope es una herramienta que apoya las fases experimentales de desarrollo de SMA [58, 59]. El objetivo del proyecto AgentScope es permitir a los investigadores, ver el trabajo experimental desde una perspectiva más ambiciosa. AgentScope da apoyo a la medición, el análisis y la evaluación de escenarios para los algoritmos bases de los SMA desarrollados. Eso permite que los procesos de comunicación complejos, que subyacen en los SMA, tales como la coordinación, la negociación, la colaboración y la formación de coaliciones, se puedan estudiar en detalle. AgentScope se compone de un conjunto reducido de interfaces, en las que un SMA se puede construir de forma independiente de un tipo particular de plataforma, que permite realizar esos estudios. AgentScope define interfaces genéricas para:

- *La comunicación entre los agentes*: es un paquete independiente que define las clases y las interfaces básicas que se necesitan para escribir protocolos.

La abstracción base es un "protocolo": una implementación de un algoritmo distribuido que ofrece un servicio en particular a los "agentes" que se encuentra en los nodos de un sistema distribuido. Cada agente participante ejecuta una instancia del protocolo. Esta interfaz consiste en tres clases (protocolos, mensajes y eventos), y tres interfaces (Agente, Dirección y Reloj). Las clases representan los principales objetos que manipula un protocolo, las interfaces representan conceptos de apoyo al sistema. Una implementación de un protocolo se hace usando la clase *Protocolo*. La interfaz del agente proporciona métodos para el envío y recepción de mensajes y el establecimiento y la recepción de eventos.

- *La medición y el análisis de comportamiento de los agentes*, define un método genérico para experimentar y probar el comportamiento de los protocolos. La interfaz de medición y análisis está diseñada para permitir que el código de medición de un protocolo pueda ser fácilmente cambiado y reemplazado, dependiendo del experimento que se ejecute, para lo cual ofrece clases para definir: (1) los datos que deben registrarse en el comportamiento de un protocolo durante una prueba experimental, y (2) cómo se deben manipular los datos para proporcionar el producto final del experimento. La abstracción de base es una "bitácora" (un espacio en blanco en la que una instancia registra información del protocolo, junto con los métodos que definen la forma de procesar la información).
  La interfaz de medición y análisis consta de dos clases, bitácora (*LogBook*) y su subclase *ActiveLogBook*, y una interfaz, *Logger*. *LogBook* manipula los datos registrados durante un experimento. Para escribir un experimento, un diseñador debe usar la subclase *LogBook* para especificar los datos de interés, y los métodos para su análisis. La subclase *ActiveLogBook* agrega la funcionalidad de registrar los datos de forma independiente a la aplicación del protocolo. El *kit* de herramientas AgentScope contiene un paquete de apoyo para usar esas clases, así como para generar, manipular y analizar gráficos y tablas de datos.
- *La configuración de escenarios experimentales*, proporciona un medio para configurar los experimentos. La configuración y el funcionamiento de un experimento se divide en varias partes. Un experimento es un escenario experimental completo, mientras que los ensayos son pruebas individuales de ese escenario. El núcleo del escenario experimental se separa de las partes del experimento que se pueden configurar en cada ensayo. Para ello se proponen clases en esta interfaz, que permiten: (1) ejecutar una serie de ensayos, (2) capturar la información acerca de la configuración del entorno, (3) inicializar la configuración de los agentes para un escenario experimental, y (4) proveer la información sobre la configuración de un ensayo en particular. AgentScope proporciona una sencilla interfaz de línea de comandos, una interfaz Web, y un gestor de archivos genérico.

En AgentScope, en lugar de colocar los agentes y sus comportamientos en general, se pueden usar abstracciones de agentes que proporcionen un conjunto de servicios distribuidos específicos para realizar las pruebas. Este

enfoque considera un SMA no sólo desde la perspectiva de un agente, sino también como un sistema distribuido.

4.5.5. IMPACT

IMPACT es un sistema desarrollado con el propósito de proporcionar un marco para construir agentes en la parte superior de fuentes heterogéneas de conocimiento [34, 62]. IMPACT proporciona la idea de un programa de agente escrito bajo un lenguaje de llamadas a procedimientos. Un código de llamadas puede ser visto como el encapsulamiento de cualquiera código legado. Esto le da independencia al código de agente generado, lo que permite definir nociones deónticas en las acciones de los agentes en ciertos momentos: "obligatorio", "permitido", "prohibido", etc.

Esta lógica de los programas de agentes se asemeja a programas lógicos extendidos con modalidades deónticas. El sistema IMPACT proporciona una serie de características, incluyendo el registro de los servicios de agentes disponibles y páginas amarilla. IMPACT ha sido extendido para soportar razonamiento temporal y probabilístico.

4.5.6. AgentBuilder

AgentBuilder es una entorno para implantar agentes basado en la programación orientada a agentes, desarrollado por Acronymics Inc. Se basa en el lenguaje *Reticular Agent*, que es una extensión de *Agent Shoham* [21, 61]. Como el lenguaje de agente utilizado no está destinado para la programación directa, un desarrollador de agente tiene que utilizar el IDE de AgentBuilder, que consiste en una variedad de herramientas de apoyo para todos los aspectos de la construcción de aplicaciones de agentes. El IDE se concibe para ocultar el código de agente. Ofrece sencillas funcionalidades de gestión de proyectos que se integran con la herramienta.

AgentBuilder proporciona un modelo mental de gran alcance para sus agentes, lo que permite a los desarrolladores especificar aspectos tales como, creencias, intenciones, compromisos y reglas de comportamiento. Además, hace uso de estándares como KQML para el lenguaje de comunicación entre agentes, UML para modelar el conocimiento, etc. Así, AgentBuilder proporciona IGUs para casi todas las tareas de desarrollo de agentes, tales como: organización general del proyecto, especificación de protocolos, especificación del comportamiento del agente, creación y depuración de agentes. Todos ellos están unidos bajo un IDE sencillo en un entorno Microsoft Windows.

AgentBuilder está basada en Java. Su interfaz proporciona la mayoría de las funcionalidades de *MASDK* (*Multi-Agent System Development Kit*). El depurador puede realizar tareas básicas, como establecer puntos de interrupción y pasar a través de acciones de los agentes. La ejecución puede ser detenida, los valores cambiados, la ejecución puede ser reanudada, etc. Es adecuado para la implementación de agentes que utilizan la arquitectura BDI. Los agentes pueden ser dinámicamente agregados a un SMA que se ejecuta, pero esta función no está en la herramienta.

4.5.7. Agent Factory

Es un entorno de desarrollo que ofrece soporte para la edición y montaje de los distintos componentes de un agente [39]. Contiene una estructura en capas para el desarrollo y despliegue de aplicaciones orientado a agentes. El centro de este marco es el lenguaje de agentes AF-APL (*Agent Factory Agent Programming Language*). El interpretador de AF-APL está incrustado en el entorno distribuido *Run-Time Environment* (RTE), compatible con FIPA.

AF-APL se basa en el paradigma de la programación orientada a agentes, siguiendo los lineamientos de Shoham, ampliado con conceptos de la arquitectura BDI, para incluir creencias y planes. Los detalles de este modelo se pueden encontrar en [39]. Específicamente, el modelo define el estado mental de un agente comprendido por creencias y compromisos.

En AF-APL, el conjunto de creencias se componen de un conjunto de declaraciones sobre el estado actual del medio ambiente. El programador puede declarar explícitamente, para cada agente, un conjunto de sensores (perceptores) y efectores (actuadores). Los perceptores son instancias de clases de Java que definen cómo convertir los datos del sensor en creencias del agente. Del mismo modo, un actuador es una instancia de una clase Java que tiene dos funciones: definir el identificador de la acción que se debe utilizar cuando se refieran al actuador, y contener el código que implementa la acción. Estas declaraciones definen la configuración del agente, y especifican el programa del agente. Una herramienta llamada *VIPER* permite la composición de agentes, y usa los diagramas de secuencia de UML para definir el protocolo de comunicación entre los agentes. Además de las herramientas que se han proporcionado para apoyar el desarrollo de agentes AF-APL, el *Agent Factory Development Environment* incluye un conjunto de herramientas para la verificación y depuración de aplicaciones orientadas a agentes.

4.5.8. Agent World Editor

Es una herramienta para el diseño de SMA y la generación de código de agentes [22]. La herramienta propone un modelo de agente abstracto, que lo hace compatible con tres marcos de agentes diferentes, de la familia JIAC. Sus características son:

- Montaje rápido de configuraciones complejas de agentes.
- Interfaz gráfica intuitiva.

Permite el seguimiento del comportamiento de los agentes en tiempo de ejecución, para fines de depuración o de demostración. Como muchos de los componentes del SMA son posiblemente ejecutados en sistemas físicos diferentes, mantiene una visión general del medio ambiente donde se ejecuta, a través de un método para monitorear infraestructuras de SMA, llamado *ASGARD* (*Advanced Structured Graphical Agent Realm Display*) . ASGARD proporciona una representación gráfica en 3D de los componentes conectados. Sus principales características son:

- Visualización en 3D de la infraestructura de agentes.
- Vista en línea de los agentes y nodos de agentes en la plataforma distribuida.
- Visualización de la comunicación y las migraciones en el sistema.
- *Plugins* para funciones especializadas.
- Manipulación de los estados y migración de los agentes.

Proporciona un conjunto de herramientas de diseño, basado en BPNM (*Business Process Modeling Notation*), de fácil comprensión e independiente de la plataforma. Entre las herramientas tenemos:

- Un editor que soporta el modelado BPMN.
- Un editor para el diseño de procesos basado en patrones.
- Un sistema para importar servicios existentes (desde la Web).
- Un visor de la estructura de los procesos.
- Un simulador.

También permite:

- La generación de texto en lenguaje natural.
- La validación y traducción de expresiones
- La transformación a *BPEL* y a *JIAC*
- La importación y exportación a STP-BPMN.

4.5.9. JASON

JASON es un entorno para el desarrollo de SMA, que usa una extensión del lenguaje de programación orientado a agente *AgentSpeak* para programar el comportamiento de los agentes [35, 23]. *AgentSpeak* es un lenguaje de programación orientado a agentes, muy usado para programar arquitecturas BDI. JASON es un software de código abierto desarrollado en Java, se distribuye bajo la licencia GNU LGPL, y permite la personalización de la mayoría de los aspectos de un agente o de un SMA. Se presenta como un *plugin* para cualquier editor (Eclipse, etc.), y puede ser usado por diferentes plataformas de despliegues de SMA (por ejemplo, JADE). Algunos detalles interesantes son:

- Los actos de habla son la base de la comunicación inter-agentes.
- Las anotaciones de creencias en las fuentes de información son un meta-nivel de información que pueden ser usados tanto para la comunicación entre agentes como para la definición de funciones.
- Usa extensiones del lenguaje para declarar meta-eventos, objetivos declarativos, variables de orden superior, etc.
- Las funciones son completamente adaptables (en Java).
- Posee una biblioteca de las "acciones internas" esenciales.
- Se puede extender directamente por acciones definidas por el usuario, programadas en Java.

Jason se distribuye con un IDE [23], que proporciona una interfaz gráfica para la edición de un archivo de configuración de un SMA, así como para la edición del código de los agentes individuales programados en *AgentSpeak*. A través de la IDE, también es posible ejecutar y controlar la ejecución de un SMA, y distribuir los agentes en una red de una manera simple. El IDE proporciona otra herramienta, llamada *Mind Inspector*, que permite al usuario inspeccionar los estados internos de los agentes, cuando el sistema se ejecuta en modo de depuración.

4.5.10. IDEs para desarrollar agentes móviles basados en Java

Diferentes *kits* de desarrollo de agentes móviles están disponibles. Cada *kit* tiene sus propias fortalezas y debilidades. Algunos basados en Java son [24, 35, 25, 68]:

- *Aglets*: este kit de desarrollo de software es del Laboratorio de investigación de IBM de Tokyo. Los agentes móviles, llamados Aglets, son implementados en forma de hilos en la máquina virtual de Java. El espacio de datos de Aglets contiene referencias a los recursos del sistema, así como

a otros Aglets. Tiene una buena IGU, pero es débil a nivel de seguridad. Otro aspecto negativo es la escalabilidad, no es interoperable con diferentes plataformas.

- *Jumping Beans*: desarrollado por Adastra Ingeniería, es un marco comercial de implementación de agentes móviles. En este marco, el código de los agentes móviles incluye clases Java específicas, así como las clases que son parte de Java. Los agentes móviles se implementan como hilos de Java.
- *Grasshopper*: desarrollado por GMD FOKUS, es basado en MASIF. Es una plataforma móvil abierta para agentes inteligentes, basada en Java, compatible con FIPA.
- *Voyager*: es un kit de desarrollo de agentes móviles en Java, de Recursion Software Inc. Entre sus características destacan: el soporte para servicios de directorio, y la posibilidad de creación y conexión de redes de directorios con el fin de formar una estructura de directorios interrelacionados.
- *Anchor Agent Development Toolkit*: facilita la gestión segura y la transmisión entre agentes móviles en entornos heterogéneos. Se basa en Aglets.

4.6. Plataformas especiales para implantar SMA

En esta sección se presentan otras propuestas, que por sus características particulares son de interés. Específicamente, se describe una plataforma basada en modelos de normas para organizar comunidades de agentes, otra plataforma basada en flujos de trabajo que siguen la especificación *BPEL4WS* para desarrollar SMA, y finalmente, uno de los proyectos mas ambiciones en el área, que consiste en proponer una plataforma mundial para el despliegue de agentes heterogéneos.

4.6.1. THOMAS

THOMAS (*MeTHods, Techniques and Tools for Open Multi-Agent Systems*) es una plataforma que propone un modelo de normas para la organización de los agentes, que cubre tanto el nivel institucional como el de interacción entre agentes, proporcionando un marco adecuado para la gestión de organizaciones de agentes [42].

Se han definido sistemas normativos en el campo de la informática como "sistemas basados en comportamientos donde las normas tienen el papel de establecer los conceptos normativos que permiten describir o especificar dichos comportamientos" [42]. Por lo tanto, "un SMA normativo (SMAN) es aquel que tiene mecanismos para representar, comunicar, distribuir, descubrir, crear, modificar y hacer cumplir las normas, y mecanismos para delibe-

rar acerca de las normas y detectar la violación o cumplimiento de las mismas" [42]. Así, los SMAN combinan los modelos para sistemas normativos (por ejemplo, sobre obligaciones, permisos y prohibiciones) con modelos de SMA. Por otro lado, una "Sociedad de Agentes", también llamada "Organización de Agentes", se puede entender como "una entidad compleja donde una multitud de agentes interactúa en un ambiente estructurado con miras a un propósito global". Las organizaciones de agentes también se llaman Organizaciones Virtuales (OV) , por las características computacionales de los agentes. En estos sistemas hay una clara separación entre la forma y la función de la OV, lo que requiere la definición de cómo el comportamiento de sus componentes se llevará a cabo. Eso conlleva a la necesidad de desarrollar plataformas para la implantación de agentes para OV.

THOMAS constituye un avance en este sentido. En concreto, implementa abstracciones normativas para una plataforma de agentes específicamente diseñada para la ejecución de OV. Dicha plataforma considera los cuatro aspectos principales de un OV:

- *La dimensión estructural*, que describe los componentes del sistema y sus relaciones.
- *La dimensión funcional*, que detalla la funcionalidad del sistema.
- *La dimensión normativa* , que define los mecanismos empleados por una sociedad con el fin de influir en el comportamiento de su miembros.
- *La dimensión ambiental*, que describe el entorno en términos de sus recursos y cómo los agentes pueden percibir y actuar sobre ellos.

Una OV se define como una tupla,

$$OV = (A, R, OU, N, S, ER, Ac, F, Roles, Juegos)$$

donde:

- $A = \{a_1, \ldots, a_m\}$ es el conjunto de agentes. Cada agente es descrito por etiquetas (a_i), que identifican a los miembros de la OV por un nombre único.
- $R = \{r_1, \ldots, r_n\}$ es el conjunto de etiquetas que identifican los roles que se han definido en el OV.
- $OU = \{OU_0, OU_1, \ldots\}$ es el conjunto de unidades de organización que se ha definido en la OV, donde cada unidad organizativa es una partición de A ($OU_i \in P(A)$).
- N es el contexto normativo de la OV, el conjunto de normas que controlan las operaciones llevadas a cabo.
- $S = \{s_1, \ldots, s_k\}$ es el conjunto de servicios definidos dentro de la OV. Cada servicio es identificado por una etiqueta única (s_i).
- ER es el conjunto de recursos que se encuentran en el medio ambiente de la OV.
- Ac es un conjunto de acciones individuales primarias.
- F es el conjunto de hechos básicos de la OV.

- *Roles*: $OU \Rightarrow R$ es una función que mapea cada unidad organizativa con el conjunto de roles que pueden realizar los agentes pertenecientes a esa unidad.
- *Juegos*: $A \times OU \Rightarrow R$ es una función que asigna a cada agente y unidad con el conjunto de roles desempeñados por el agente en el interior de esta unidad específica.

Esta definición teórica de un OV permite especificar:

- *La dimensión estructural*, es decir, el conjunto de agentes (A), roles (R) y unidades organizativas (OU) que forman una OV. Una unidad organizativa (OU) es una entidad social básica que representa el conjunto mínimo de agentes que realizan actividades o tareas específicas y diferenciadas, siguiendo un patrón predefinido de cooperación y comunicación. Una unidad organizativa incluye un conjunto de funciones (R) que deben realizar sus miembros.
- *La dimensión funcional*, es decir, el conjunto de servicios definidos por la OV. Los servicios representan la funcionalidad que las entidades ofrecen a los demás. La OV permite describir la funcionalidad de los agentes en términos de servicios. En THOMAS usan la ontología *OWL-S* para ello. De esta manera, cualquier entidad, sean agentes o unidades organizativas, son capaces de ofrecer, publicar o solicitar alguna funcionalidad específica. Por lo tanto, la petición (*Request*), publicación (*Register*) y disposición (*Serve*) de servicios son acciones válidas de los agentes en un OV.
- *La dimensión de medio ambiente*, es decir, los recursos (ER), ubicados en el medio ambiente de la OV. El objetivo es la descripción de los recursos ambientales, y la percepción y acción sobre estos elementos. Por lo tanto, la actuación (*Act*) y percepción (*Percept*) de los recursos ambientales pertenecen al conjunto de acciones válidas (Ac).
- *La Dimensión normativa* es un mecanismo de coordinación que trata de: (i) promover comportamientos que son satisfactorios para la organización, es decir, acciones que contribuyan a la consecución de los objetivos globales, y (ii) evitar acciones dañinas, es decir, acciones que hagan al sistema inestable.

En general, las normas se clasifican en tres categorías [42]: las normas constitutivas, regulativas y de procedimiento. *Las normas constitutivas* permiten dar un significado abstracto a los hechos, elementos del medio ambiente, etc. *Las normas regulativas* describen el comportamiento ideal de un sistema por medio de las obligaciones, prohibiciones y permisos. Por último, *las normas procedimentales* son un enfoque instrumental, se definen los mecanismos para hacer cumplir las normas en términos de castigos y recompensas para los agentes de las OVs. Por lo tanto, estas normas definen una conexión práctica entre los reglamentos y sus consecuencias.

Las principales características de una norma están relacionadas con su formalización, distribución, evaluación, ejecución y aprendizaje. Una norma se formaliza si es reconocida como tal por la institución. En cuanto a

la distribución de una norma, representa el grado de conocimiento de la norma, es decir, si la norma se distribuye a todos los miembros de la sociedad, a algún grupo de ellos, o si no se distribuye. El proceso de aprendizaje permite que un agente internaliza una norma, y puede ser: primario (si el agente debe conocer la norma como un resultado de ser un miembro de la sociedad); o secundaria (si el agente debe conocer la norma cuando se asume una posición concreta dentro de la sociedad). La evaluación de una norma determina si el cumplimiento de la norma puede ser observado por todos los miembros de la sociedad (por terceros) o sólo por los agentes afectados por una interacción. Por último, la ejecución tiene dos atributos: los mecanismos de aplicación empleados para la consecución del cumplimiento de normativa, y la entidad de aplicación (entidad a cargo de la aplicación de la norma).

THOMAS es una plataforma que emplea un enfoque basado en servicios. Se basa en FIPA, usando sus componentes principales (AMS y DF), pero lo extiende con un Sistema de Gestión de la Organización y un Manejador de servicios:

- *Manejador de Servicios (SF)*, registra los servicios prestados por entidades externas y facilita el descubrimiento de servicios. Por lo tanto, resuelve las limitaciones del DF tradicional, teniendo en cuenta la información semántica, composición de servicios, así como roles, objetivos y duración de los servicios, siguiendo las directrices del paradigma SOA (*Service-Oriented Architecture*).
- *Sistema de Gestión de la Organización (OMS)*, responsable de la gestión de las OV, tomando el control de su estructura, el papel desempeñado por los agentes dentro de la organización, y las normas que rigen el comportamiento del sistema. Más específicamente, es el encargado de controlar las OU (creación; entidades que participan, roles que desempeñan en el tiempo, etc.).

El proceso de gestión de las normas cubre: (i) el mantenimiento del estado de la organización, (ii) la ejecución de las medidas de sanciones y recompensas, y (iii) la determinación de las acciones permitidas de acuerdo con el marco normativo y el estado de la organización.

Estas funcionalidades se han implementado como servicios internos de THOMAS, por medio de un sistema basado en reglas implementado en JESS. La Figura 4.16 (tomada de [42]), muestra una vista esquemática de este proceso. Este sistema de normas es accedido por los servicios de THOMAS, a través de dos servicios diferentes: un servicio de consulta, que permite determinar si un agente está autorizado a solicitar un servicio, y un servicio de actualización que se encarga de añadir y borrar datos en el sistema de reglas, con el fin de registrar el estado actual de la organización. Además, este sistema controla el comportamiento de los agentes y lleva a cabo las sanciones y beneficios asociados a las normas. Cada uno de sus cuatro componentes principales se explica brevemente a continuación:

- *Interpretación de la norma*: la especificación formal de una norma se traduce en un hecho o instancia normativa. Por lo tanto, es necesario traducir las normas abstractas en normas concretas de acuerdo con la ontología normativa usada, con el fin de hacer a las normas computables.
- *Detección de la activación de la norma*: las normas no están activas en todo momento. La activación de las normas podría ser definida en términos de condiciones de estado y de tiempo. Como consecuencia de ello, se necesita un conjunto de reglas para la detección de la activación y desactivación de las normas. Su instanciación en un momento dado tiene que ver con el proceso de interpretación de la misma en un momento dado.
- *Aplicación de la norma*: el sistema de gestión de normas está informado de lo que sucede en THOMAS: cambios en la organización, ocurrencia de eventos, etc. Por lo tanto, es capaz de controlar la manera en que los agentes interactúan con la infraestructura THOMAS. Para ello, la plataforma THOMAS actúa como promotor y defensor de las normas.
- *Ejecución de la norma*: cuando una nueva solicitud de un servicio es enviada a THOMAS (vía OMS o SF), se registra este hecho en el sistema de normas. Luego, utilizando el servicio de consulta, se analiza si el agente está autorizado a solicitar este servicio de acuerdo al estado de organización. Esta función sigue un criterio de prioridad de normas, estableciendo que la norma más específica precede al resto de las normas. De esta manera, el servicio de consulta comienza con la comprobación de las normas permitidas al agente. Si no hay ninguna norma, se toman en consideración las normas de prohibición. Si no se encuentra ninguna norma, se considera el servicio por omisión.

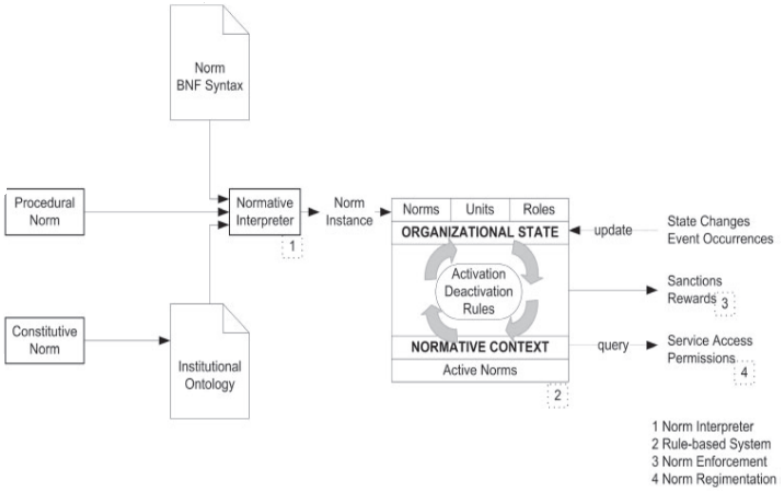

Figura 4.16: Visión general del sistema de Normas de THOMAS.

4.6.2. Plataforma basada en BPEL4WS

Es una plataforma, basada en *BPEL4WS* [57], que permite el desarrollo de flujos de trabajo genéricos para SMA basados en la especificación. La plataforma permite la composición de servicios Web. Parte de un modelo del proceso de negocio que da una visión general de la tarea a realizar. Todos los participantes en el sistema son vistos como agentes genéricos, que no tienen conocimiento de ningún servicio Web particular. El único conocimiento que tienen es la forma de ejecutar el protocolo de interacción y como invocar los servicios Web correctamente. Además, tienen un motor de flujo de trabajo como intérprete de dichos flujos, cuando el motor recibe un flujo de trabajo, ejecuta tal flujo de una manera centralizada. El motor gestiona la interacción de cada servicio Web en el flujo de trabajo, lo que garantiza que todas las operaciones se lleven a cabo según lo especificado en la descripción *BPEL4WS*. La desventaja de este enfoque es que, aunque el motor puede ejecutar estas invocaciones de forma asíncrona (generando así un cierto grado de paralelismo), el proceso es centralizado, sufriendo de las debilidades asociadas a los diseños centralizados. Por otro lado, en ciertos ambientes el enfoque centralizado no es posible, por ejemplo, en un entorno para dispositivos móviles. Puesto que el motor de flujo de trabajo central sólo se utiliza para la lógica del proceso y para controlar las interacciones entre los diferentes servicios Web, según la lógica de proceso, se puede eliminar mediante el uso de SMA.

Tiene básicamente cuatro componentes:

- *La especificación BPEL4WS*: el modelo de proceso de BPEL4WS da una visión general de toda la tarea, y especifica cómo se utilizan los servicios Web.
- *El protocolo de SMA LCC*: se deriva de BPEL4WS, y describe la lógica del proceso. El protocolo LCC (*Lightweight Coordination Calculus*) es utilizado directamente por el SMA que realiza las tareas definidas en la especificación BPEL4WS, de una manera completamente descentralizada mediante la eliminación del motor de flujo de trabajo centralizado.
- *El controlador del flujo de trabajo distribuido*: es responsable de manejar la lógica del proceso y la interacción con los servicios Web. Cada uno de los agentes en el flujo de trabajo representa un rol que se define en la especificación inicial BPEL4WS.
- *El iniciador de agentes*: es un agente que lee el protocolo de interacción multiagente e instancia los roles definidos en el flujo de trabajo distribuido en los agentes.

4.6.3. *AgentCities*

Durante la última década, la investigación sobre agentes ha alcanzado un importante nivel de madurez en teorías, lenguajes y metodologías. A pesar de estos éxitos, la visión de agentes inteligentes como entidades autónomas, sin problemas para interactuar unos con otros en entornos heterogéneos, aún no se ha resuelto. Algunos de los problemas por resolver tienen que ver con temas como la coordinación, la comunicación, el descubrimiento, la confianza, la seguridad y las ontologías; que a pesar de que FIPA presenta algunas propuestas a nivel de estándares, y las plataformas actuales presentan algunas formas de resolverlo, son de forma aislada y no garantizan la interoperabilidad entre esas soluciones. *AgentCities* es una iniciativa para crear una red global abierta de plataformas y servicios heterogéneos para agentes, tal que cualquiera pueda conectar sus agentes [26].

AgentCities se basa en los siguientes principios:

- *Normas consensuales*. La comunicación e interacción en la red se basan en normas de acceso público, tales como las desarrolladas por FIPA y W3C (*World Wide Web Consortium*).
- *Código abierto*. Aunque las tecnologías comerciales no se excluyen, AgentCities promueve la implementación de código abierto de libre acceso, para garantizar el acceso libre y abierto a la red.
- *Acceso abierto*. Cualquier organización o individuo puede crear su propia AgentCity en la red, para alojar sus propios servicios para agentes, facilitar el acceso a ellos, y acceder a los desplegados por los demás.
- *Recursos compartidos*. Los investigadores acceden a servicios compartidos para agentes en la red, tales como servicios de directorio, de nombres, ontologías y aplicaciones.

En particular, AgentCities pretende crear una red en la que los agentes que se ejecutan en diferentes plataformas, propiedad de diferentes organizaciones, implementados de forma diferente y con diversos servicios, puedan interactuar. Aunque existen normas como *DAML+OIL*, *ebXML*, *XML*, *RDF*, etc., la base de la interoperabilidad es el estándar FIPA. Los nodos de la red AgentCities son plataformas de agentes que se ejecutan en una o más máquinas, que pertenecen a una organización o individuo. Los agentes que se ejecutan en una determinada plataforma deberán ser capaces de conectarse a otras plataformas de acceso público y comunicarse directamente con sus agentes. Aplicaciones que combinan servicios de agentes de diferentes AgentCities se pueden crear mediante el uso flexible de este modelo de comunicación entre agentes, y los marcos semánticos, ontologías compartidas, lenguajes de contenido y protocolos de interacción en que se basa.

Algunos nodos de la red proporcionan servicios útiles, tales como servicios de directorio (páginas amarillas y blancas), servicios de ontologías (repositorios para compartir definiciones ontológicas), puertas de enlace (para realizar traducciones entre diferentes protocolos de transportes, lenguajes,

dominios de seguridad, etc.), servicios de pruebas (sistemas para probar la interoperabilidad de las plataformas, dar seguimiento, etc.). La lista de nodos AgentCity crece a medida que nuevos sitios se incorporan.

El despliegue de agentes consiste en instalar una plataforma de agentes en una máquina conectada a Internet, y anunciar la dirección de la nueva plataforma. Los agentes de la plataforma pueden obrar recíprocamente con otros agentes, que ya están en la red, para registrar sus servicios, y para buscar y aprovechar los servicios en el entorno distribuido.

Si bien la infraestructura (mensajería, directorios, etc.) es necesaria para crear la red, el objetivo de AgentCities no es desplegar una infraestructura. El objetivo principal de AgentCities es crear un rico y abierto ambiente para explorar preguntas, entre éstas: ¿cómo los servicios pueden descubrirse?, ¿cómo desarrollar y usar diversas ontologías? y ¿cómo se pueden coordinar agentes en sistemas heterogéneos?. AgentCities apunta a resolver los problemas semánticos, ontológicos y de composición dinámica de servicios en entornos de agentes heterogéneos. Algunas de las aplicaciones de interés sobre AgentCities son:

- *Servicios de viajes, turismo y ocio*: aplicación principal de AgentCities.
- *Servicios de negocios*: sistemas de pagos, servicios de transacción y catálogo.
- *Tecnologías de coordinación*: conjunto de mecanismos para la cooordinación de comunidades de agentes.
- *Servicios médicos y de salud*: servicios distribuidos para el acceso a registros médicos de los pacientes, etc.
- *Integración de cadenas de producción*: uso de la infraestructura AgentCities para la coordinación de procesos de fabricación distribuida y la integración con cadenas de suministro.
- *Servicios de seguridad*: uso de la infraestructura Agentcities como banco de pruebas para abordar las necesidades de seguridad de entornos heterogéneos abiertos.
- *eLearning*: sistemas tutoriales basados en agentes distribuidos.
- *Aplicaciones inalámbricas*: interacción entre los agentes a través de redes de telefonía móvil y cable, para componer dinámicamente servicios basado en la ubicación del usuario.
- *Personalización*: composición dinámica de servicios a usuarios según sus gustos individuales.

Existen proyecto alrededor de AgentCities financiados por la Comisión Europea (*AgentCities.RTD* y *AgentCities.NET*), así como en otras partes del mundo (Japón, Estados Unidos, Australia, entre otros). Aunque los proyectos tienen sus propios objetivos, están unidos en el objetivo de crear una infraestructura de interoperabilidad mundial basada en normas comunes.

En particular, el proyecto AgentCities.RTD (investigación y desarrollo) despliega catorce plataformas AgentCities, cada una de las cuales es atendida por un socio del proyecto. Este proyecto establece las bases para la red

AgentCities: el desarrollo de servicios, la implementación de plataformas, la infraestructura de red, ejemplo de aplicaciones, y realiza investigación fundamental a nivel de marcos de comunicación, la arquitectura de la red y la composición de servicios dinámicos.

El proyecto AgentCities.NET establece una serie de acciones para hacer que la red Agentcities sea accesible a la investigación y a las empresas de la comunidad europea. Las actividades incluyen:

- *Jornadas de información*: se puede acceder fácilmente a la información sobre la forma de explotar y participar en la red.
- *Becas de apoyo para el despliegue*: proporcionar pequeñas donaciones iniciales a las organizaciones que deseen conectar sus propios sistemas a la red.
- *Programa de intercambio de estudiantes*: apoya a los estudiantes mediante la financiación de pequeñas estadías en algunos de los participantes en AgentCities.
- *Concursos de aplicaciones*: fomenta el desarrollo mediante el reconocimiento de las aplicaciones más innovadoras creadas en la red
- *Grupos de trabajo*: estructura para agrupar a los participantes alrededor de ciertos temas.

A pesar que las acciones involucran contribuciones financieras dirigidas a entes europeos, la participación también está abierta a organizaciones de países no pertenecientes a la Unión Europea. Hasta la fecha hay cincuenta organizaciones.

Los "grupos de trabajo" en AgentCities (ACTF), que actúan como un foro abierto para la coordinación global de los diversos esfuerzos relacionados a temas de AgentCities, que actualmente existen, son:

- *De coordinación*: facilita la coordinación entre los diferentes proyectos y actividades que contribuyen al uso de AgentCities.
- *De soporte a la red*: fomenta y da soporte a los componentes de la red, tales como directorios, repositorios de ontologías, etc.
- *De promoción, difusión y enlace*: da a conocer el trabajo que se realiza en la red, y fomenta el interés, la participación y el desarrollo en ella.

AgentCities está teniendo un impacto en el despliegue global de agentes, y proporciona un recurso útil para el desarrollo de la próxima generación de sistemas distribuidos. Sin embargo, AgentCities requiere de mucho esfuerzo a nivel del desarrollo de ontologías, marcos semánticos y lenguajes de contenido, antes que los agentes puedan comunicarse fácilmente. El papel de AgentCities es estimular ese proceso, para proveer sistemas en el contexto de un ambiente mundial. AgentCities está ayudando a crear una verdadera interoperabilidad, no sólo a nivel sintáctico, sino también a nivel semántico.

Referencias

1. http://www.fipa.org/.
2. www.omg.org.
3. http://www.daml.org.
4. www.ontoknowledge.org/oil.
5. http://jade.tilab.com.
6. http://leap.crm-paris.com/.
7. http://www.jessrules.com/jess.
8. www.protege.stanford.edu.
9. http://fipa-os.sourceforge.net/.
10. www.labs.bt.com/projects/agents/zeus.
11. http://www.agent-software.com.au/.
12. http://www.janus-project.org/Documentation.
13. www.intsci.ac.cn/en/research/mage.html.
14. www.madkit.org.
15. www.jcp.orghttp://gti-ia.dsic.upv.es/sma/tools/Magentix/index.php.
16. http://www.java-agent.org/.
17. www.jcp.org.
18. www.zti.polsl.pl/dmrozek/umap.
19. http://sourceforge.net/projects/amine-platform.
20. http://www.cis.ksu.edu/ sdeloach/ai/download-agentool.htm.
21. http://www.agentbuilder.com.
22. http://www.jiac.de/development-tools/jiac-toolipse/agent-world-editor/.
23. http://jason.sourceforge.net.
24. aglets.sourceforge.net/.
25. http://www.grasshopper.de.
26. http://www.agentcities.org.
27. Spade: Smart python multi-agent development environment. http://gti-ia.dsic.upv.es/projects/magentix/.
28. tinymas: a tiny multiagent java platform. www.arakhne.org/tinymas/.
29. Jose Aguilar. *Introducción a los Sistemas Emergentes*. En publicación, 2014.
30. J. Alberola, J. Such, and V. Botti. A scalable multiagent platform for large systems. *ComSIS*, 10(1):51–77, 2013.
31. T. Tsó B. Neuman. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9):33–8, 1994.
32. J. Barthés. Omas v7 platform, features. Technical report, 2009.
33. F. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. John Wiley & Sons, 2007.
34. R. Bordini, L Braubach, M. Dastani, A. El Fallah, J. Gomez, J. Leite, G. OHare, A. Pokahr, and A. Ricci. A survey of programming languages and platforms for multi-agent systems. *Informática*, 30:33–44, 2006.
35. R. Bordini, J.F. Hubner, and M. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley & Sons, 2007.
36. P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. *JACK Intelligent Agents - Components for Intelligent Agents in Java*. AgentLink Newsletter, 2005.
37. V.C. Carrascosa, M. Rebollo, J. Soler, and V. Botti. Simba: an approach for real-time multi-agent systems. In *V Conferencia Catalana dÍntelligncia Articial*. Springer-Verlag, 2002.
38. K. Chouchane, O. Kazar, and A. Aloui. Agent-based approach for mobile learning using jade-leap. Technical report, Computer Science Department, Faculty of Sciences University Hadj Batna, Algeria, 2010.
39. R. Collier. *Agent Factory: A Framework for the Engineering of Agent-Oriented Applications*. PhD thesis, University College Dublin, 2001.

40. J. Collis and D. Ndumu. *The ZEUS Technical Manual*. http://www.labs.bt.com/projects/agents/zeus/.
41. J. Collis, D. Ndumu, and S. Thompson. *ZEUS Methodology Documentation: Role Modelling Guide, Three case studies, Application Realisation Guide, Runtime Guide*. http://www.labs.bt.com/projects/agents/zeus/.
42. N. Criado, E. Argente, and V. Botti. THOMAS: An agent platform for supporting normative multi-agent systems. *J Logic Computation*, 2011.
43. S. DeLoach. Multiagent systems engineering: a methodology and language for designing agent systems. In *Agent Oriented Information Systems*, number 45-57, 1999.
44. S. DeLoach and M. Wood. Developing multiagent systems with agenttool. In *7th. International Workshop Intelligent Agents VII. Agent Theories, Architectures, and Languages*, 2000.
45. O. Dikeneli, R. Erdur, O. Gumus, E. Ekinci, O. Gurcan, G. Kardas, I. Seylan, and A. Tiryaki. SEAGENT: A platform for developing semantic web based multi agent systems. In *AAMAS*, pages 1271–1272. ACM, 2005.
46. M. Escrivá, J. Palanca, G. Aranda, and A. Jabberbased. A jabber based multiagent system platform. In *AAMAS*, pages 1282–1284. ACM, 2006.
47. J. Ferber and O. Gutknecht. A meta-model for analysis and design of multi-agent systems. In *3rd International Conference on Multi-Agent Systems*, pages 155–176. IEEE, 1998.
48. Jabber Software Foundation. *Extensible Messaging and Presence Protocol (XMPP): Core*. http://www.ietf.org/rfc/rfc3920.txt, 2004.
49. E. Friedman-Hill. *Jess in Action: Rule Based Systems in Java*. Manning Publications, 2003.
50. N. Gaud, S. Galland, V. Hilaire, and A. Koukam. An organizational platform for holonic and multiagent systems. In A. Pokahr E. Hindriks and S. Sardina, editors, *Sixth International Workshop on Programming Multi-Agent Systems*, pages 111–126, 2008.
51. J. Graham, D. McHugh, M. Mersic, F. McGeary, M. Windley, D. Cleaver, and K. Decker. Tools for developing and monitoring agents in distributed multi agent systems. Technical report, University of Delaware, 2010.
52. O. Gutknecht and J. Ferber. Vers une méthodologie organisationnelle pour les systèmes multi-agents. In *JFIADSMA*, 1999.
53. O. Gutknecht and J. Ferber. The madkit agent platform architecture agents. In *Workshop on Infrastructure for Multi-Agent Systems*, 2000.
54. A. Kabbaj. Uses, improvements and extensions of prolog+cg: Case studies. In *ICCS'01*, 2001.
55. A. Kabbaj. Amine architecture. In *Conceptual Structures Tool Interoperability Workshop*, 2006.
56. A. Kabbaj, K. Bouzouba, K. ElHachimi, and N. Ourdani. Ontologies in amine platform: Structures and processes. In *ICCS'06*, 2006.
57. L.Guo, D. Roberston, and Y. Chen-Burger. Business process model based multiagent system development. In *Second Workshop On Collaboration Agents: Autonomous Agents for Collaborative Environments*, 2009.
58. E. Ogston and F. Brazier. Agentscope: Multi-agent systems development in focus. Technical report, Delft University of Technology, 2010.
59. E. Ogston and S. Jarvis. Peer sampling with improved accuracy. *Peer-to-peer Networking and Applications*,, 2(1):51–71, 2009.
60. M. Patzlaff and E. Tuguldur. Microjiac 2.0 - the agent framework for constrained devices and beyond. Technical report, DAI Labor, TU Berlin, 2009.
61. Y. Shoham. Agent-oriented programming. *Artificial, Intelligence*, 60(1):51–92, 1993.
62. V. Subrahmanian, P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Özcan, and R. Ross. *Heterogenous Active Agents*. MIT-Press, 2000.

63. S. Sung and H. Youn. Calm: an intelligent agent-based middleware for community computing. In *Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, 2006.
64. K. Szczypiorski, I. Margasinski, and W. Mazurczyk. Steganographic routing in multi agent system environment. *Journal of Information Assurance and Security*, 2(3):235–243, 2007.
65. K. Szczypiorski, I. Margasiski, W. Mazurczyk, K. Cabaj, and P. Radziszewski. Trustmas: Trusted communication platform for multi-agent systems. Technical report, Warsaw University of Technology, Faculty of Electronics and Information Technology, 2012.
66. E. Tuguldur and M. Patzlaff. Microjiac agents in multi-agent programming contest. In *Fifth International Workshop on Programming Multi-Agent Systems*, 2007.
67. E. Tuguldur and M. Patzlaff. Moo - microjiac agents operating oxen. Annals of Mathematics and Artificial Intelligence, 2010.
68. K. Wang, W. Abdulla, and Z. Salcic. Ambient intelligence platform using multi-agent system and mobile ubiquitous hardware. *Pervasive and Mobile Computing*, 5(5):558–573, 2009.
69. M. Winikoff. *JACKTM intelligent agents: An industrial strength platform*, chapter 7, pages 175–193. 2004.

Capítulo 5
Un Medio de Gestión para SMA

Resumen: En este capítulo se detalla la construcción de un Medio de Gestión de Servicios orientado a Sistemas Tiempo Real.

5.1. Introducción

La implantación de agentes en ambientes reales requiere, en un primer paso, la definición de la arquitectura del SMA, que establece las interacciones básicas entre las comunidades de agentes, y luego el modelado (especificación) de los agentes. El producto principal sería un conjunto de plantillas, modelos o listas de agentes y de sus interacciones, las cuales después deberían ser expresadas en algún lenguaje de programación. Estos pasos pueden llevarse a cabo siguiendo alguna metodología específica, como la que se presentó en el Capítulo 3.

Luego, para la operación del SMA es necesario contar con una plataforma básica que provea los servicios fundamentales para el funcionamiento/despliegue de los agentes. Estos servicios, clásicos en los sistemas distribuidos, deben incluir mecanismos para la creación, el nombramiento, la localización, la búsqueda, la comunicación, entre otros. En este sentido, se requiere de un Medio de Gestión de Servicios (MGS) (*Middleware*) que permita la operación de los agentes de una manera segura y eficiente, sin afectar los objetivos propios del SMA.

En la Sección 4.2 se presentó el estándar FIPA, que surge con el objetivo de unificar el esquema de desarrollo y funcionamiento de los sistemas basados en la tecnología de agentes, promoviendo la interoperabilidad con otras tecnologías [5]. Particularmente, FIFA propone una especificación que define al MGS como un sistema constituido por los recursos de hardware y software (sistema operativo, software de comunicaciones, software

de gestión de agentes) necesarios para que los agentes puedan ser ejecutados/desplegados. Si bien es cierto que existen ambientes para el desarrollo e implantación de aplicaciones basadas en agentes que usan el estándar FIPA, como por ejemplo JADE (Sección 4.3.1), este ambiente puede resultar poco atractivo para propósitos prácticos de implantación en ambientes reales, específicamente industriales, debido a aspectos operacionales tales como:

1. Restricciones para respuesta en tiempo real estricta.
2. Dependencia de la máquina virtual de Java.

En general, para los componentes del sistema que requieran procesamiento tiempo real, la tecnología Java presenta limitaciones, tomando en cuenta que los dispositivos donde se ejecutan dichos procesos tienen capacidades de cómputo limitada. Es por ésto que es necesario disponer de una plataforma que permita la implantación de sistemas basados en agentes, que no tenga las limitaciones temporales de las plataformas existentes.

La propuesta que se presenta en este capítulo, corresponde a la conceptualización, diseño orientado a objetos y codificación, del conjunto de módulos de software que proporcionan los servicios del MGS requerido para soportar aplicaciones basadas en SMA para entornos de tiempo real, usando las especificaciones de FIPA. Particularmente, en esta propuesta dichos módulos también son concebidos como agentes y se asume que la implementación usa como base el sistema operativo Linux y algunas bibliotecas particulares para la comunicación.

5.2. Descripción General

El Medio de Gestión de Servicios (MGS) es el conjunto básico de módulos de software que implantan las abstracciones mínimas para la ejecución/despliegue de agentes en un ambiente computacional. El MGS propuesto inicialmente en [4], extendido en trabajos posteriores [3], presentado a continuación, pretende conformidad arquitectural con el estándar FIPA, mostrado en la Figura 4.1. El MGS esta compuesto por 3 niveles:

- **Interfaz**. Define la interfaz entre el SMA y los componentes del sistema distribuido. Esta constituido por cinco agentes: Agente Administrador de Agentes (AAA), Agente Gestor de Recursos (AGR), Agente Gestor de Aplicaciones (AGA), Agente Gestor de Datos (AGD) y Agente de Control de Comunicación (ACC). Los agentes AGR, AGA y AGD son especializaciones del Directorio Facilitador (DF) especificado en FIPA (ver la Figura 4.1). El nivel interfaz se encarga de establecer las pautas de conversación entre los componentes del sistema distribuido y el SMA.
- **Medio**. Constituye el núcleo del sistema distribuido, provee servicios de software que requieren los agentes para poder interactuar entre sí y con

el nodo/sitio de ejecución. Proporciona transparencia y seguridad en las transacciones, interoperabilidad de las aplicaciones y componentes de software, migración de agentes, objetos y/o recursos, comunicación interprocesos, localización de recursos (agentes y objetos), y un sistema de nombramiento para la localización de agentes y/o objetos.

- **Acceso a Recursos**. Esta integrado por el núcleo básico del sistema operativo, el cual maneja las funcionalidades de tiempo real, cuando sean necesarias, y posee manejadores de acceso a hardware específico que requiera el sistema.

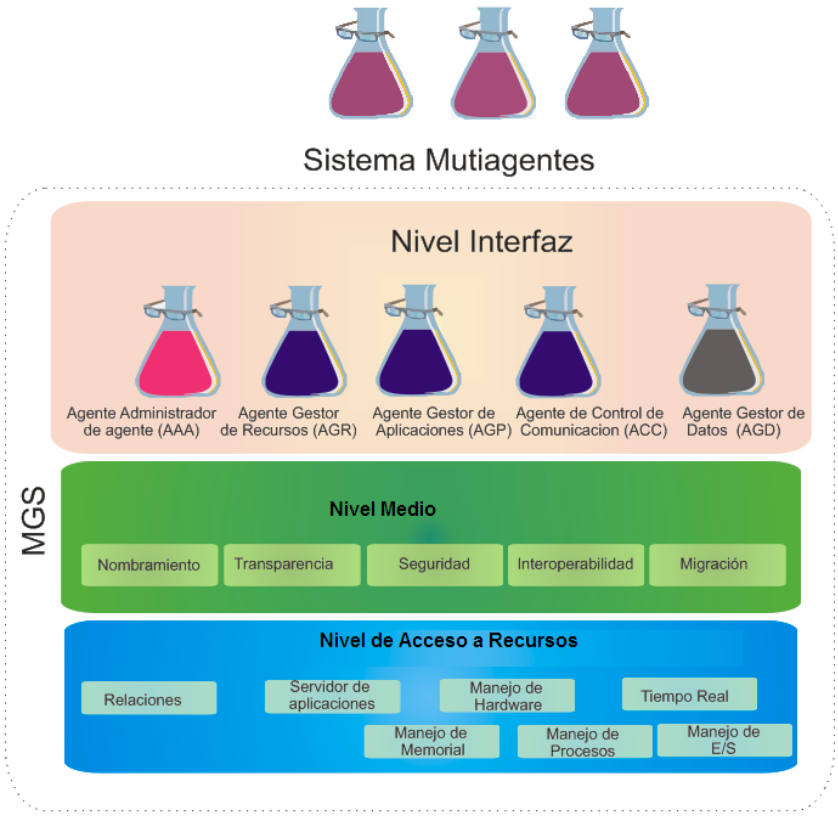

Figura 5.1: Arquitectura básica del MGS.

La Figura 5.1 muestra los tres niveles que componen el MGS y la relación jerárquica entre éstos. En términos operacionales, cada nodo de red de la plataforma computacional, que preste servicios de procesamiento al SMA, deberá estar ejecutando una instancia del MGS. Desde el punto de vista topológico, relativo al funcionamiento dentro de una arquitectura de auto-

matización, los agentes descritos anteriormente se implantan en diferentes niveles y de manera distribuida.

5.3. Servicios del MGS

Como se mencionó anteriormente, el objetivo del MGS es brindar servicios operacionales para que el SMA cumpla con el objetivo para el cual ha sido diseñado. Así, el MGS debe proveer mecanismos que permitan a los agentes [1] (comunidad de agentes del SMA):

- Conocer los servicios disponibles agrupados por niveles, por tipos de servicio, o alguna otra clasificación (tarea que realizan AAA, AGR, AGA, AGD).
- Conocer los agentes que prestan un determinado servicio y dónde están (tarea que realizan AAA, AGR).
- Conocer las formas para acceder a esos servicios. ¿Cómo se invocan?, ¿Qué requerimientos se deben cumplir para accederlos?, entre otros.
- La movilidad, el MGS debe proveer los mecanismos de movilidad y controlar el proceso de migración (tarea que realiza AAA).
- Comunicarse con otros agentes (tarea que realiza ACC).
- Su activación y desactivación, la posibilidad de invocarlos, etc. Es decir, la gestión general de ellos (tarea que realiza AAA).

Por otro lado, el MGS requiere una serie de servicios para asegurar que puede cumplir su cometido. Entre estos servicios están:

- Acceso a las tablas de proceso para mantener control de los agentes en ejecución.
- "Copiado" del estado de un agente para manejo de su migración.
- Mecanismo para controlar las comunicaciones.
- Esquemas de manejo de seguridad (por usuario, por servicio, por recurso, etc.).
- Esquemas de virtualización de recursos.

Particularmente, cuando un SMA es concebido para aplicaciones en tiempo real, como algunas de las aplicaciones en el ámbito de automatización industrial, el MGS debe proveer servicios que garanticen, al menos, comunicación y procesamiento en tiempo real, entre otros. Estos servicios deben estar soportados por el sistema operativo sobre el cual se implementa el MGS.

---

[1] Los agentes del SMA también pueden ser referenciados como agentes de negocio; es decir los agentes que representan abstracciones de los elementos del sistema real que está modelando.

5.4. Conceptualización del MGS

Como pautas para la conceptualización, se ha asumido el sistema operativo Linux, o alguna versión de éste, como base para implementar al MGS. Cuando el SMA requiera servicios en tiempo real, que deben ser provistos por el MGS, entonces se usaría Linux con capacidades de procesamiento tiempo real (RT-Linux) [2]. De esta manera, los procesos del MGS enlazan e invocan bibliotecas de ese sistema operativo.

En base a la arquitectura de referencia de la Figura 5.1, los niveles inferiores del MGS están estructurados de tal manera que deben ser instanciados, obligatoriamente, por los agentes del Nivel Interfaz para la realización de sus actividades. En esta propuesta, el *nivel medio* de la Figura 5.1 va a estar implementado por dos módulos: el ***Agent Manager*** y el ***Communication Manager***, llamando ahora a ese nivel el *Nivel Base.* Así, los niveles del MGS estarán ahora estructurados como se muestran en la Figura 5.2 (el nivel base es equivalente al nivel medio de la Figura 5.1, mientras que el nivel de acceso a recursos es el núcleo del sistema operativo).

El *Agent Manager* es responsable de administrar a los agentes en la plataforma computacional, y está estructurado en tres sub-módulos: Dispacher, Mapper y Locator. El *Communication Manager* se encarga de proveer comunicación confiable a los agentes, en un enfoque de red orientada a invocación.

Así, de manera general, en relación a los aspectos de implementación, como muestra la Figura 5.2, el MGS ha sido especificado a partir de dos capas: Nivel Interfaz y Nivel Base. El nivel interfaz brinda todos los servicios requeridos por las comunidades de agentes para operar como SMA. El nivel base, el cual combina algunas de las funcionalidades requeridas en los niveles medio y de acceso a recursos de la arquitectura básica de referencia del MGS en la Figura 5.1, contiene el núcleo básico del sistema operativo distribuido. Finalmente, el resto de funcionalidades del nivel de acceso a recursos locales es provisto por el núcleo básico del sistema operativo usado.

La conceptualización del MGS se ha realizado siguiendo la metodología MASINA, presentada en el Capítulo 3. En las siguientes secciones presentaremos los productos generados para esta fase de conceptualización.

5.4.1. Conceptualización del Nivel Interfaz

El objetivo esencial de la conceptualización es comprender los requisitos y características que demandan los usuarios, así como hacer el primer

[2] RTLinux proporciona capacidades de ejecución de tareas de tiempo real y manejadores de interrupciones. Cada versión de RT-Linux funciona sobre una versión determinada de Linux estándar.

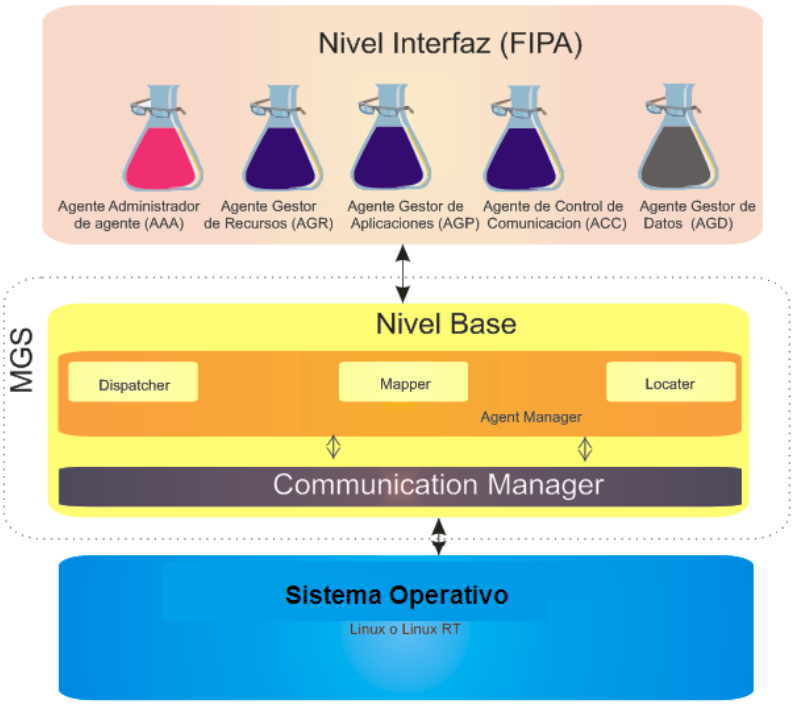

Figura 5.2: Esquema de implantación del MGS.

bosquejo del modelo orientado a agentes. En esta etapa se utiliza a UML, a través de los diagramas de casos de uso y los diagramas de actividades. A continuación, se presentan estos diagramas, asociados a los cinco agentes del nivel interfaz del MGS.

5.4.1.1. Agente Control de Comunicaciones

La Figura 5.3 muestra el diagrama de caso de uso *Gestionar Comunicación*. Este caso de uso involucra a los agentes del sistema que desean comunicarse; para este fin, deben usar los servicios ofrecidos por el ACC. A su vez, este agente utiliza los servicios del nivel base para realizar el proceso de transferencia de mensajes. La Tabla 5.1 describe, formalmente, este caso de uso.

En la Figura 5.4 se observa el diagrama de actividades general del ACC; éste recibe los mensajes con la información del emisor y del destinatario. A continuación, determina la localización del agente (local o remota) y envía el mensaje a dicha localización.

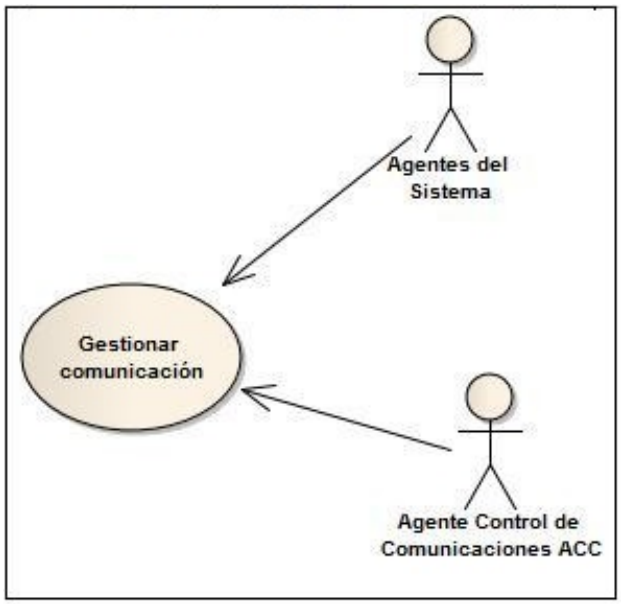

Figura 5.3: Diagrama de caso de uso Gestionar Comunicación.

Caso de Uso	Gestionar Comunicación
Descripción	Se encarga de la comunicación entre los agentes a través del envió y recepción de mensajes
Pre-condición	Existencia de Agentes
Actores	Agentes del SMA, ACC
Condición de fracaso	No enviar ni recibir mensajes entre agentes
Condición de éxito	Envió y recepción de mensajes entre agentes

Tabla 5.1: Especificación del caso de uso Gestionar Comunicación

5.4.1.2. Agente Administrador de Agentes (AAA)

Como se mencionó anteriormente, el AAA gestiona el sistema de agentes (controla, registra, y administra los agentes). Para esto, opera en conjunción con el nivel base, garantizando la operatividad del sistema de agentes. El caso de uso general para el AAA se muestra en la Figura 5.5.

La descripción formal del caso de uso se presenta en la Tabla 5.2. Este caso de uso representa una especificación genérica, el mismo es dividido en varios casos que permitan describir, detalladamente, los diferentes servicios ofrecidos por el AAA.

El diagrama de actividades mostrado en la Figura 5.6 ilustra con más detalles los diferentes servicios ofrecidos por el AAA. Se puede observar que el AAA ofrece 5 servicios básicos: creación, destrucción, movimiento, localización y cambio de estado de los agentes. En cada invocación recibe la solicitud

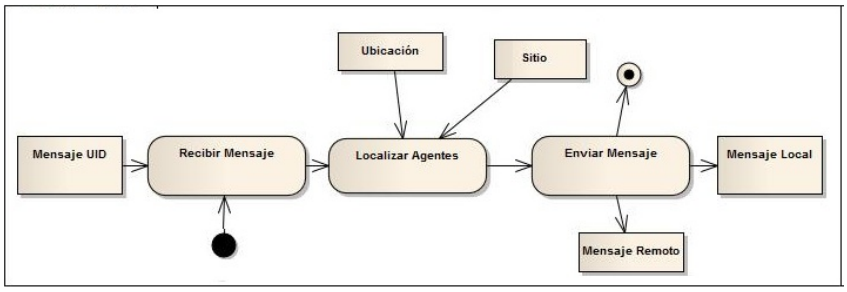

Figura 5.4: Diagrama de actividades del ACC.

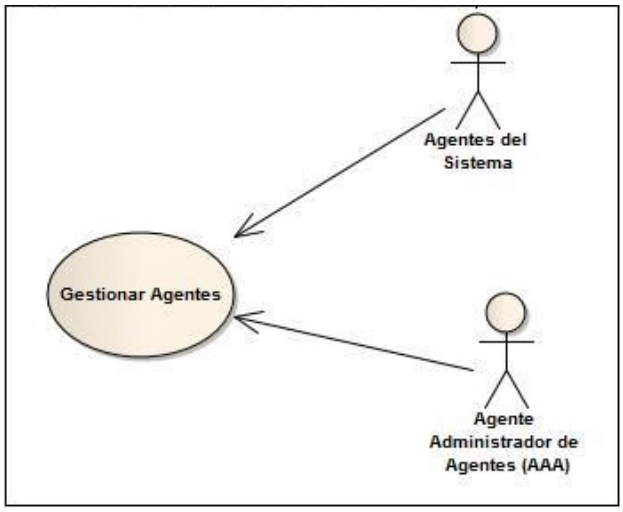

Figura 5.5: Diagrama del caso de uso Gestionar Agentes.

Caso de Uso	Gestionar Agentes
Descripción	El AAA gestiona agentes, de manera que éstos puedan ser usados por el SMA.
Pre-condición	Existencia de Agentes
Actores	Agentes del Sistema, AAA
Condición de fracaso	Gestión no cumplida
Condición de éxito	Invocación cumplida

Tabla 5.2: Especificación del caso de uso Gestionar Agentes

respectiva, y los parámetros requeridos que le permiten dar respuesta a tal invocación. Para cumplir con estos servicios debe establecer un mecanismo de manejo de recursos, y establecer una interacción apropiada con el nivel base.

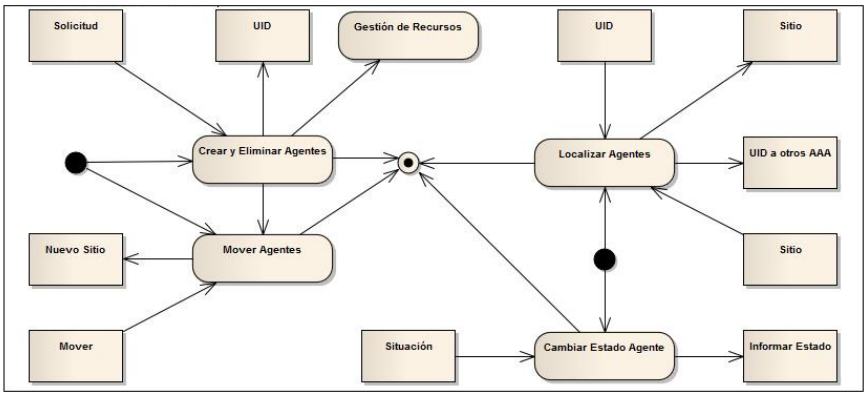

Figura 5.6: Diagrama de actividades del AAA

5.4.1.3. Agente Gestor de Aplicaciones (AGA)

El AGA permite la gestión general de los agentes de aplicaciones especializados; entre los servicios del AGA se encuentran la localización de agentes de aplicaciones (se comporta como un servidor de páginas amarillas). La Figura 5.7 muestra el caso de uso para el AGA.

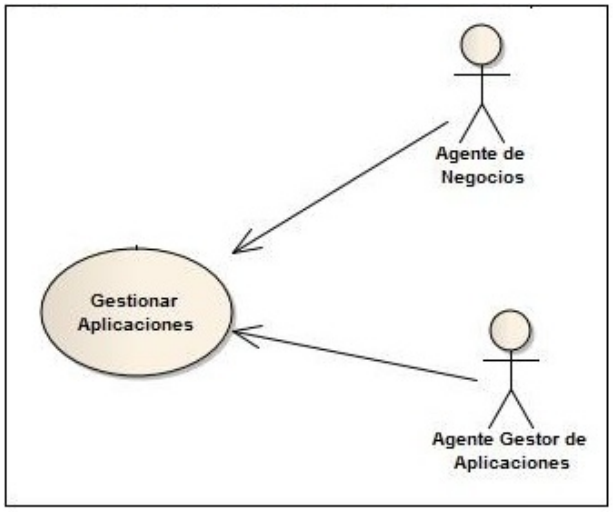

Figura 5.7: Diagrama del caso de uso Gestionar Aplicaciones

En la Tabla 5.3 se especifica el caso de uso para *Gestionar Aplicaciones*. El AGA se encarga de ubicar las aplicaciones y/o agentes especializados que puedan ser requeridos por los Agentes del Sistema.

Caso de Uso	Gestionar Aplicaciones
Descripción	Ubicar las aplicaciones y/o Agentes de aplicaciones que puedan ser requeridos por los Agentes
Pre-condición	Existencia de la Aplicación y/o Agente de Aplicación y su requerimiento
Actores	AGA y Agente del sistema solicitante
Condición de fracaso	No transmitir la aplicación seleccionada
Condición de éxito	Transmitir la aplicación seleccionada para ser utilizada

Tabla 5.3: Especificación del caso de uso Gestionar Aplicaciones.

El AGA ejecuta dos operaciones básicas: la primera es la localización de las aplicaciones, y la otra es la selección, en caso que existan varias aplicaciones que cumplan con los criterios de búsqueda. Para cumplir con su función, el AGA debe llevar un registro de los agentes especializados que prestan servicios de aplicaciones. La Figura 5.8 muestra el diagrama de actividades del AGA.

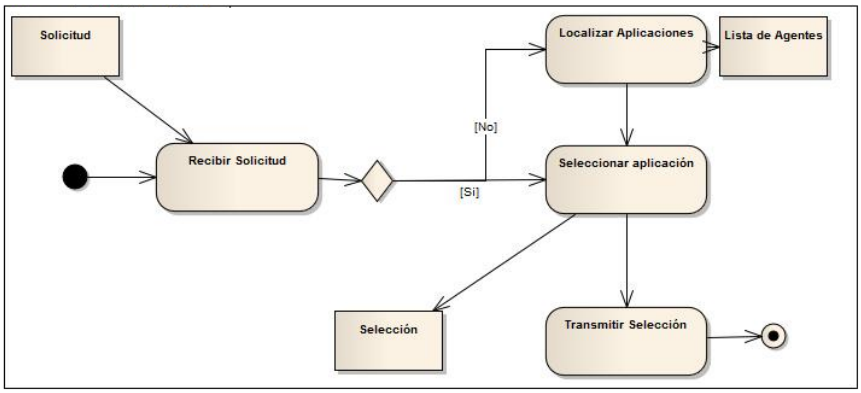

Figura 5.8: Diagrama de actividades del AGA

5.4.1.4. Agente Gestor de Recursos (AGR)

El AGR permite la administración de los recursos que deben ser compartidos entre los agentes del sistema. El AGR gestiona los recursos que sean declarados como compartidos, y lleva a cabo el control del acceso y la asig-

nación de los mismos. En la Figura 5.9 se muestra el caso de uso *Gestionar Recursos*. Este caso de uso se describe en la Tabla 5.4.

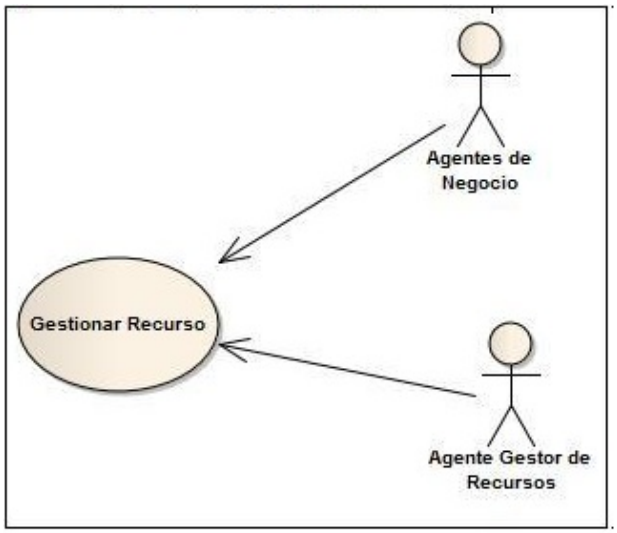

Figura 5.9: Diagrama del caso de uso Gestionar Recursos.

Caso de Uso	Gestionar Recursos
Descripción	Se encarga de ubicar los recursos que puedan ser requeridos por los agentes del sistema
Pre-condición	Existencia del recurso y Agente y su requerimiento
Actores	Agente de Negocio y Agente Gestor de Recursos
Condición de fracaso	No Gestionar correctamente los recursos
Condición de éxito	Gestionar correctamente los recursos

Tabla 5.4: Especificación del caso de uso Gestionar Recursos.

La Figura 5.10 muestra el diagrama de actividades del AGR. Este recibe una solicitud de algún agente del sistema; localiza el recurso solicitado, para lo cual mantiene el registro de los recursos, si está disponible lo asigna, y realiza la actualización del estado del recurso; de igual manera, debe permitir liberar un recurso, en caso de que un agente que lo tiene asignado lo desee liberar.

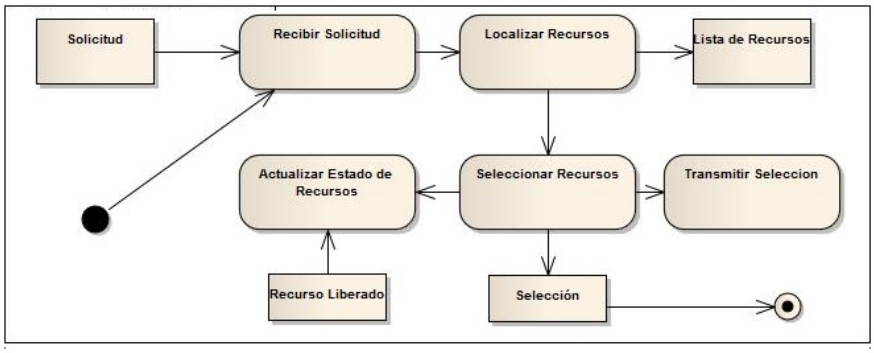

Figura 5.10: Diagrama de actividades del AGR

5.4.1.5. Agente Gestor de Datos (AGD)

Este agente se encarga de establecer el enlace con los lugares donde existan datos de interés para el proceso (agente) que se esté ejecutando, sea que estos datos provengan de bases de datos (relacionales, orientadas a objetos, tiempo real, etc.), de SCADAs, de sensores, o de cualquier otro dispositivo o aplicación que pueda almacenar datos. Además, permite el traslado de los datos entre los diferentes dispositivos y/o aplicaciones de una manera transparente, para lo cual realiza las transformaciones requeridas y el mantenimiento de los medios de almacenamiento de los datos. La Figura 5.11 muestra el caso de uso asociado al AGD.

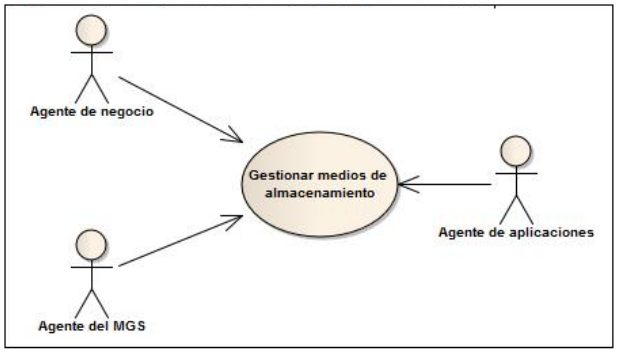

Figura 5.11: Diagrama del caso de uso Gestionar Medios de Almacenamiento

El caso de uso para Gestionar Medios de Almacenamiento se describe en la Tabla 5.5.

Caso de Uso	Gestionar Medios de Almacenamiento
Descripción	Procesa solicitudes de consulta, actualización y de mantenimiento de los medios de almacenamiento de datos. Además, realiza un servicio de auditoría sobre las propiedades de calidad de los servicios ofrecidos
Pre-condición	Existencia de comunicación de datos o solicitudes a procesar, de mantenimiento de los medios de almacenamiento o de auditoría
Actores	Agentes de Aplicaciones, Agentes del sistema y AGD
Condición de fracaso	Error de comunicación, en la recepción de datos, en la recepción de la solicitud, o en la ejecución de la tarea de mantenimiento
Condición de éxito	Datos o solicitud recibidos, datos entregados, o tarea de mantenimiento realizada

Tabla 5.5: Especificación del caso de uso Gestionar Medios de Almacenamiento.

La Figura 5.12 muestra el diagrama de actividades para el AGD. En este diagrama se observan las actividades llevadas a cabo por este agente para cumplir con la funcionalidad asignada.

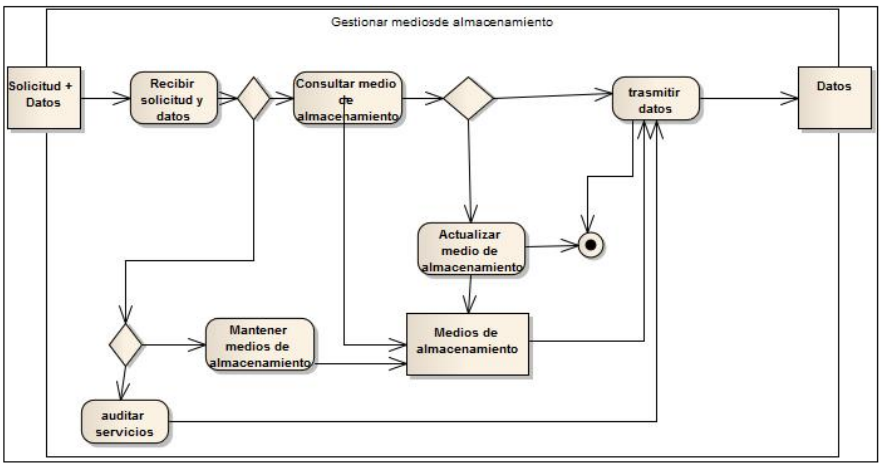

Figura 5.12: Diagrama de actividades del AGD

5.4.2. Conceptualización del Nivel Base

A partir de las funcionalidades del nivel interfaz, tomando en cuenta los servicios que éste requiere, se define el nivel base, tal como se muestra en

la Figura 5.13. Se asume Linux como sistema operativo, en sus versiones tradicionales y su versión tiempo real. Por esa razón, y por simplicidad, se contempla que los agentes residen en procesos Linux. Cada proceso debe enlazar e invocar una biblioteca (*library*) que implanta las llamadas al MGS. Esto se realiza en cada proceso que denominaremos *Ejecutivos*.

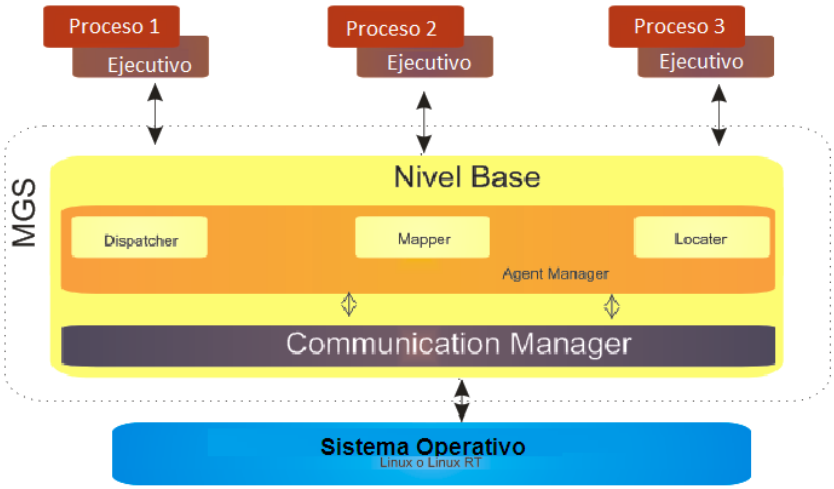

Figura 5.13: Arquitectura del núcleo operacional.

EL MGS en sus niveles inferiores está estructurado en dos módulos que deben ser instanciados, obligatoriamente, por los agentes del Nivel Interfaz (FIPA) para la realización de sus actividades: el manejador de agentes y el manejador de comunicación. Desde el punto de vista de la implantación, estos módulos se corresponden con procesos residentes en un servidor, ó repetidos en varios servidores, para el caso distribuido, tal y como se presentará en la Sección 5.7.

5.4.2.1. Manejador de Agentes (*Agent Manager*)

El Manejador de Agentes se encarga de corresponder agentes con procesos Linux. Contempla funciones como creación, destrucción y manejo de procesos del sistema operativo para la manipulación de agentes. La creación de identificadores únicos también es responsabilidad. Además, el *Agent Manager* debe implantar la invocación de agentes bajo los esquemas estáticos y dinámicos. Los esquemas de migración, intra-sitio o inter-sitio son gestionados por este módulo; de igual manera, la localización de agentes. Este módulo está estructurado en tres sub-módulos:

1. *Despachador (Dispatcher Inter)*: se encarga de despachar invocaciones a los agentes. Del lado superior, recibe invocaciones desde los procesos y las hace llegar al despachador remoto a través del manejador de comunicación. Del lado inferior, recibe invocaciones remotas y se las hace llegar al proceso que contiene el agente. El despachador puede implantar despacho dinámico, en el cual un agente puede construir, en tiempo de ejecución, infraestructuras para recibir invocaciones.
2. *Mapper*: se encarga de otorgar identificadores únicos y de gestionar los recursos del sitio para los agentes y procesos. Este módulo es responsable de la creación y destrucción de agentes. Además, este módulo gestiona la migración de agentes.
3. *Localizador (Locator)*: Se encarga de localizar agentes usando sus identificadores únicos.

5.4.2.2. Manejador de Comunicación (*Communication Manager*)

Este módulo se encarga de proveer comunicación confiable de red orientada a invocación. La semántica queda a decidir entre "*a lo más una vez*" o "*exactamente una vez*", según las suposiciones de fallas que se consideren para los agentes. El *Manejador de Comunicaciones* debe permitir utilizar los esquemas de comunicación síncrona o "$Rendez-Vous$", asíncrona, y semántica "RPC" (*Remote Procedure Call*) [9].

La comunicación $Rendez-Vouz$ es un esquema de sincronización que permite a dos procesos concurrentes intercambiar datos de forma coordinada. Los procesos involucrados pueden quedar en espera inactiva mientras se llega al punto de reencuentro, esto es, el proceso que solicita debe esperar en el punto de reencuentro hasta que el proceso llamado llegue allí o el proceso llamado puede llegar antes que el solicitante y debe esperarlo para poder continuar procesando. En la comunicación *asíncrona*, el proceso emisor continua su ejecución, sin esperar a confirmar que el mensaje sea recibido por el proceso receptor. En el esquema de comunicación remota *RCP* el proceso emisor permanece suspendido hasta que el proceso destino acepta la llamada. Ambos procesos permanecen sincronizados durante la ejecución de un segmento de código común.

A diferencia del Manejador de Agente, el Manejador de Comunicación no está conformado por otros módulos, y en esta propuesta, los componentes del *Manejador de Comunicaciones* podrían estar implantados mediante procesos privilegiados Linux, como se presentará en la Sección 5.5.3.

5.5. Diseño del MGS

Siguiendo la metodología MASINA, presentada en el Capítulo 3, una vez concluida la fase de conceptualización, se desarrolla la fase de análisis, que permite definir en detalle los servicios y tareas para el MGS. Esta fase, descrita en detalle en [1], no es presentada aquí, pasando a detallar a continuación la fase de diseño.

5.5.1. Definiciones generales para el diseño

El diseño del MGS debe garantizar su implementación usando algún paradigma de programación, que en esta propuesta será la orientada a objetos. Para comprender la propuesta de diseño con miras a la implementación, la asociación de agente con objetos y, finalmente, su relación con la especificación FIPA, se utilizan ciertas definiciones que caracterizan el diseño.

1. Objeto: un objeto es una especificación encapsulada de un ente, bajo un tipo de dato abstracto, donde se oculta completamente su implantación y se define claramente su interfaz de operación. Para crear o modelar objetos de algún tipo se definen las *clases*, las cuales, a parte de contener los tipo de dato abstracto (TDA), contienen los métodos (o las operaciones) de manipulación de dichos datos. Un objeto creado a partir de una determinada clase es una *instancia* de esa clase.
2. Agente: un agente es un ente obrante cuyas funciones vislumbrables son implantar un objeto y especificar uno o más fines. En general, se puede considerar que los objetos son *objetivos*, en el sentido de que sólo se considera su interfaz, su cara exterior. Por otro lado, en los agentes se considera tanto lo interior (tareas) como lo exterior (objetivos). De esta manera, el modelo de implementación en esta propuesta asocia a los agentes con objetos, y por ende se le asocia una *clase*, donde sus métodos se corresponden, de alguna manera, a las tareas del agente.
3. Interfaces: los agentes y objetos poseen una interfaz definida por una clase de objeto compatible con la especificación CORBA IDL o CORBA UML, para garantizar la integración de los mismos.
4. Identificadores únicos: un identificador único (UID) es un objeto de identificación de instancias de objetos y de agentes, cuya identidad es única en el espacio y en el tiempo. Todo objeto o agente del sistema posee un identificador único. Un identificador único puede convertirse a una cadena de caracteres ASCII culminada en el valor de byte “0”. El MGS es responsable de generar instancias **UID**.
5. Fin: un fin es una abstracción que representa un flujo de ejecución. Tal flujo se destina a implantar el fin (*goal*) para el cual el agente haya sido programado. Pueden existir agentes que no tengan “fin”, en este caso, el

fin será identificado como $Null\_End$. Un agente cuyo fin no sea $Null\_End$ puede contener más de un fin. Los fines pueden obtenerse mediante el método *get_end*, el cual retorna el fin según el orden de inserción.

6. Comunicación fiable: el MGS maneja la comunicación de mensajes fiable entre agentes. Por fiable se entiende un mensaje que llega a su destino, íntegro, que no se duplica, o una excepción de notificación de falla. En lo que sigue, el agente que origina la comunicación se denomina remitente, mientras que el que la destina (la recibe) destinatario.

5.5.2. *Modelo de diseño de agentes del nivel interfaz*

El modelo de diseño, que apunta a la implementación de agentes para el MGS, viene dado por la jerarquía de clases de la Figura 5.14. Considerando las definiciones dadas en la Sección 5.5.1, dado que una *clase* es una construcción que se utiliza como un modelo para crear objetos, y un agente puede asociarse a un objeto, se puede definir una clase abstracta que implemente al agente fundamental y representa a la clase base. Esta clase contiene los atributos que son comunes a todos los agentes y de la cual deriven directamente el resto de agentes.

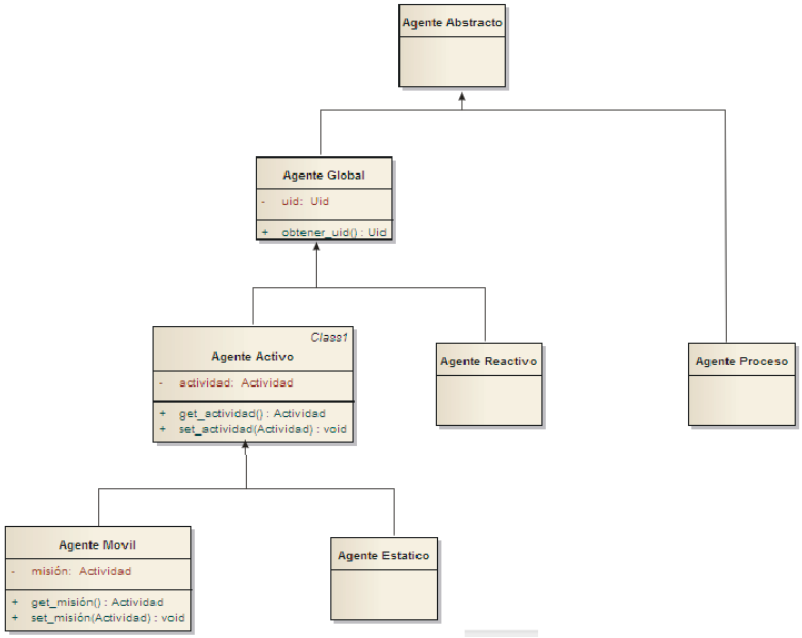

Figura 5.14: Jerarquía de clases en el sistema de agentes.

Se define, entonces, la clase *Agente Abstracto*, que contiene los atributos que son comunes a todos los agentes [2]. Una propuesta de diseño para el Agente Abstracto se muestra en la definición básica de clase, presentada en la Tabla 5.6. La especificación del universo de clases y la definición formal se muestran en las Tablas 5.7 y 5.8, respectivamente, en base a las plantillas TDSO descritas en el Capítulo 3.

AgenteAbstracto
nombreAgente:Cadena estadoAgente:Cadena descripcion:Cadena

Tabla 5.6: Definición básica de la clase Agente Abstracto

10 de Mayo de 2007		Versión 1.0
Universo de clases y TDAs AgenteAbstracto {Colección de clases y TDAs requeridos para implantar el AgenteAbstracto }		
1	AgenteAbstracto()	*AgenteAbstracto()*: Clase que permite crear un agente abstracto
2	Cadena	*Cadena*: TDA cadena de caracteres de longitud variable

Tabla 5.7: Universo de clases del Agente Abstracto

10 de Mayo de 2007		Versión 1.0
1 AgenteAbstracto( ) {Permite crear un AgenteAbstracto}		
	Especificación de atributos:	
1	nombreAgente:Cadena	*nombreAgente*: Nombre del agente dentro del SMA
2	estadoAgente: Cadena	*estadoAgente*: Estado en que se encuentra el agente: Disponible, Ocupado, Bloqueado
3	descripcion:Cadena	*descripcion*: Define el objetivo de un agente
	Especificación sintáctica: No hay	
	Declaraciones No hay	
	Especificación Semántica No hay	

Tabla 5.8: Definición Formal de la clase AgenteAbstracto

De la clase *Agente Abstracto* se derivan directamente dos clases (especializaciones) de agentes: *Agente Global* y *Agente Proceso*. La clase *Agente Global* implementa a los agentes que tienen ámbito global (distribuido) y es la clase genérica para todos los agentes que pueden ser unívocamente identificados en el sistema, y cada instancia de esta clase tendrá asociado un proceso en el sistema operativo. Un agente global es referenciable por cualquier otro

agente del sistema y posee un identificador global único, por lo tanto el mecanismo de acceso es controlado a través de su **UID**. La clase *Agente Proceso* implementa a los agentes de ámbito local. Un agente proceso es aquel cuyo acceso está restringido sólo al proceso que lo instancia. Este agente puede acceder a cualquier agente global del sistema, pero ningún agente fuera del proceso puede accederlo.

De la clase *Agente Global* derivan otras dos clases de agentes: *Agente Activo* y *Agente Reactivo*, que implementan a los agentes proactivos y reactivos, respectivamente. En términos del sistema, un agente reactivo requiere ser activado externamente para cumplir un fin, lo que significa que su comportamiento está determinado por los estímulos (mensajes) que reciba. Un agente proactivo (o agente activo, en el diagrama de clases) se activa autónomamente para cumplir uno o más fines.

Finalmente, de la clase *Agente Activo* se derivan la clase *Agente Móvil* y la clase *Agente Estático*. Un agente móvil es aquél que puede desplazarse entre los procesos de un sistema. La decisión de migrar la debe tomar el propio agente mediante el método *migrate*. Los agentes móviles deben tener una "misión"; esto es, un fin a realizarse a su llegada a destino, bajo su responsabilidad. Así, la misión es obligatoria y debe especificarse en tiempo de instanciación, pero puede modificarse, una sola vez, luego de una migración, mediante el método *set_mission*. Un agente estático identifica a un agente que no se mueve, y representan el complemento de un agente móvil.

Dentro de esta jerarquía de clases, los agentes del nivel interfaz tales como AAA, AGR, AGP, ACC y AGD se implementan a través de clases que heredan de la clase *Agente Estático*. De esta manera, en la fase de diseño de los agentes del nivel interfaz, usando las plantillas TDSO presentadas en el Capítulo 3, se detalla la especificación formal de cada clase, de los métodos y las operaciones a ella asociada [2].

Una propuesta de diseño del AAA se ilustra en la Tabla 5.9, que muestra la definición básica de la clase. El universo de clases con los TDAs, así como la definición formal, sólo con algunos métodos, se muestra en las tablas 5.10 y 5.11, respectivamente. La Tabla 5.12 ilustra el diseño del método *LocalizarAgente*. El resto del diseño del nivel interfaz se detalla en [2].

AgenteAdministradorAgentes
TablaAgentes:Tabla DescripcionPlataforma:Cadena
+agenteAdministradorAgentes(): AgenteAdministradorAgentes +crearAgenteProceso(Cadena: SolicitudACL): AgenteEstatico +registrarAgente(AgenteEstatico:agenteSolicitante):UID Lógico +desregistrarAgente(UID:uid): Lógico
Continua en la próxima página

Tabla 5.9 –Finaliza de la página anterior

+modificarDescripcion(Cadena:descripcion, UID:uid): Lógico
+consultarDescripcion(UID:uid):Cadena
+cambiarEstado(UID:uid,Entero:estado): Lógico
+traducirNombre(Cadena:nombreAgente): UID
+moverAgente(UID:uid,Cadena:destino): Lógico
+localizarAgente(UID:uid): Cadena
+recibirAgente(UID:uid): Lógico
+destruirAgenteAdministradorAgentes(UID:uid)
+destruirAgenteProceso(UID:uid)

Tabla 5.9: Definición básica de la clase AgenteAdministradorAgentes

10 de Mayo de 2007		Versión 1.0
Universo de clases y TDAs AgenteAdministradorAgentes		
{Colección de clases y TDAs requeridos para implantar el AAA }		
1	AgenteAdministradorAgentes()	*AgenteAdministradorAgentes()*: Clase que permite crear un AAA
2	Tabla	*Tabla*: TDA para almacenar objetos paramétricos en función del proceso. Se implantará como una tabla Hash
3	Cadena	*Cadena*: TDA cadena de caracteres de longitud variable
4	Entero	*Entero*: Tipo entero
5	UID	*UID*: Tipo entero que representa un identificador único en el SMA
6	Logico	*Cadena*: Tipo lógico, conformado por los valores cierto y falso

Tabla 5.10: Universo de clases del AAA

10 de Mayo de 2007		Versión 1.0
1 AgenteAdministradorAgentes ( ): AgenteEstático ( )		
{Permite manipular un agente del SMA }		
	Especificación de atributos:	
1	tablaAgentes:Tabla	*tablaAgentes*: Tabla donde se encuentran registrados los agentes del SMA
2	descripcionPlataforma: Cadena	*descripcionPlataforma*: Descripción de las características de la plataforma en donde se está ejecutando el SMA
	Especificación sintáctica:	
1	agenteAdministradorAgentes(): AgenteAdministradorAgentes	*agenteAdministradorAgentes()*: Crea un AAA
2	crearAgenteProceso(Cadena: solicitudACL): AgenteEstático	*crearAgenteProceso()*: Crea un agente
3	registrarAgente(AgenteEstatico: agenteSolicitante): UID	*registrarAgente()*: Proporciona un UID para un agente dentro del SMA y lo registra en su tabla de agentes

Continua en la próxima página

Tabla 5.11 –Finaliza de la página anterior

4	desregistrarAgente(UID: uid): Logico	*desregistrarAgente()*: Elimina el UID de un agente. Devuelve el valor cierto si se realizó la operación y falso si no
5	moverAgente(UID: uid, Cadena: destino): Logico	*moverAgente()*: Mueve un agente a una nueva posición dentro del SMA o a otro SMA. Devuelve el valor cierto si se realizó la operación y falso si no se realizó
6	localizarAgente(UID: uid): Cadena	*localizarAgente()*: Busca un agente dentro del SMA o en otro SMA e informa del sitio donde se encuentra
7	destruirAgenteProceso(UID: uid)	*destruirAgenteProceso()*: elimina un agente
	Declaraciones	
	AgenteAdministradorAgentes x	**x**: Instancia (objeto) de la clase AgenteAdministradorAgente
	AgenteEstático y	**y**: Instancia del agente que solicita un servicio al Agente Administrador de Agentes
	UID uident	**uident**: Identificador único de agente
	Cadena desc, sitio, nomAgente, dest, solACL	**desc**: Descripción de un agente **sitio**: Sitio destino del agente **nomAgente**: Nombre del agente **dest**: Posición destino del agente **solACL**: Especificación de solicitud de creación de un proceso, usando el lenguaje estándar de FIPA *Agente Comunication Languaje* (ACL), que será asociado a un agente del SMA
	Logico msj	**msj**: Valor cierto si la operación se ejecuta con éxito, falso si no
	Entero estatus	**estatus**: Estado de operación de un agente
	Especificación Semántica x.crearAgenteProceso(solACL) → y x.registrarAgente(y) → uident x.desregistrarAgente(uident) → msj x.moverAgente(uident, dest) → msj x.localizarAgente(uident) → sitio x.destruirAgente → (uident)	

Tabla 5.11: Definición Formal de la clase AgenteAdministradorAgentes

10 de Mayo de 2007		Versión 1.0
1,6 (Observador, Público) **localizarAgente(UID: uid): Cadena** {Recibe el UID del agente y devuelve su localización} {**Pre:** El UID está registrado en algún AAA } {**Pos:** Se retorna la localización del Agente }		
1	y=buscarAgente(uid)	*buscarAgente()*: Método que busca un agente localmente (en su tabla de Agentes)
Continua en la próxima página		

Tabla 5.12 –Finaliza de la página anterior

2	Si y ≠ NULL entonces cadena=L.busquedaImplicita(uid) fin si	*busquedaImplicita()*: Solicita la búsqueda de la localización al Agent Manager *L*: Locator
3	Devolver(cadena)	
	Locator L AgenteAdministradorAgentes **x** localizacion=x.localizarAgente(uid)	Verifica que el agente este registrado y solicita al agente Locator la localización del agente solicitado.

Tabla 5.12: Especificación formal del método localizarAgente

5.5.3. Modelo de diseño del nivel base

Las funcionalidades del nivel base son soportadas por la clase *Agent Manager* y la clase *Comunication Manager*, a fin de poder especificar los métodos que permitan ofrecer los servicios de localización y comunicación, entre agentes, descritos en la Sección 5.4.2. A continuación se presentan algunas definiciones formales de estas clases, el resto del diseño ha sido desarrollado en detalle en [2].

AgentManager
+AgentManager(): AgentManager +Mapper(): Mapper +DispatcherInter():DispatcherInter +Locator(): Locator

Tabla 5.13: Definición básica de la clase AgentManager

10 de Mayo de 2007		Versión 1.0
1 AgentManager( ): AgentManager {Permite crear instancias del tipo AgentManager }		
	Especificación de atributos: No Hay	
	Especificación sintáctica:	
1	AgentManager()	*AgentManager()*: Crea un agente de la clase AgentManager
2	Mapper()	*Mapper()*: Esta clase se utiliza para gestionar el registro, desrregistro y la migración de agentes

Continua en la próxima página

Tabla 5.14 –Finaliza de la página anterior

3	DispatcherInter()	*DispatcherInter()*: Esta clase permite la autentificación y el despacho de mensajes
4	Locator()	*Locator()*: Esta clase permite la localización de agentes en el sistema
	Declaraciones AgentManager o	**o**: Instancia (objeto) de la clase AgentManager
	Especificación Semántica AgentManager() → AgentManager Mapper() → Mapper DispatcherInter() → DispatcherInter Locator() → Locator	

Tabla 5.14: Definición Formal de la clase AgentManager

La clase *Agent Manager* es una clase compuesta por las clases *Mapper*, *DispatcherInter* y *Locator*. La definición básica de la clase *Agent Manager* se muestra en la tabla 5.13 y su definición formal en la Tabla 5.14. Es importante observar que la clase *Locator* es usada para implementar el objeto que permite localizar un agente, a solicitud del AAA.

La clase *Comunication Manager* es una clase compuesta por las clases *Sender*, *Receiver*, *RendezVous*, *Invoker*, *Invoked* y *DispatcherIntra*. Para manejar la comunicación se debe instanciar algún agente de comunicación (Comunication Manager) de entre aquellos que se muestran en la Figura 5.15.

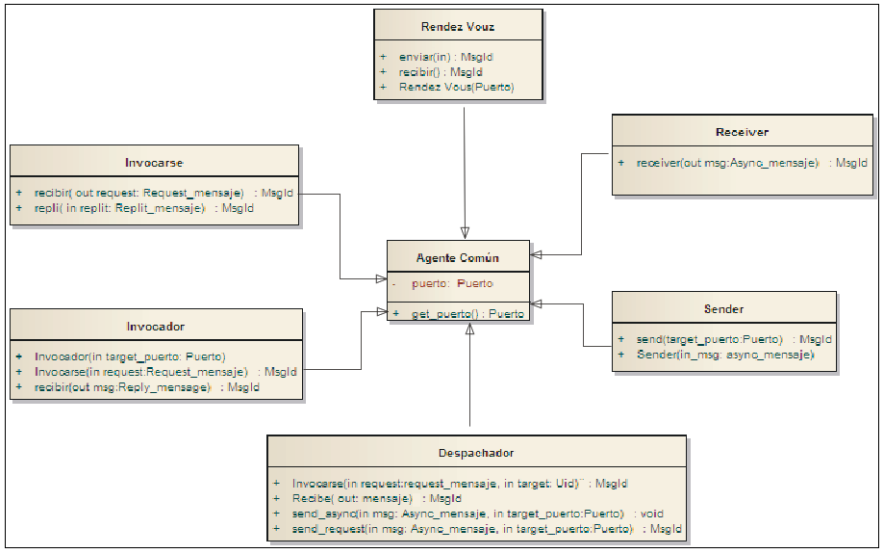

Figura 5.15: Agente de Comunicación.

De modo ilustrativo, la definición formal de las clases *ComunicationManager* y *RendezVouz* se muestran en las Tablas 5.15 y 5.16, respectivamente.

10 de Mayo de 2007		Versión 1.0
1 ComunicationManager(): ComunicationManager {Permite crear instancias del tipo ComunicationManager }		
	Especificación de atributos: No Hay	
	Especificación sintáctica:	
1	CommunicationManager()	*CommunicationManager()*: Crea un agente de la clase CommunicationManager
2	RendezVous()	*RendezVous()*: Esta clase permite la manipulación de mensajes síncronos, para lo cual tanto el remitente como el destinatario deben instanciar un agente del tipo RendezVous
3	Receiver()	*Receiver()*: Esta clase permite la manipulación de mensajes asíncronos, para lo cual el destinatario de un mensaje debe instanciar un agente del tipo Receiver
4	Sender()	*Sender()*: Esta clase permite la manipulación de mensajes asíncronos, para lo cual el remitente de un mensaje debe instanciar un agente del tipo Sender
5	Invoker()	*Invoker()*: Esta clase permite la manipulación de mensajes tipo RCP, para lo cual el remitente del mensaje debe instanciar un agente del tipo Invoker
6	Invoked()	*Invoked()*: Esta clase permite la manipulación de mensajes tipo RCP, para lo cual el destinatario del mensaje debe instanciar un agente del tipo Invoked
7	DispatcherIntra()	*DispatcherIntra()*: Esta clase se utiliza para recibir mensajes del tipo *resquest* y *reply*, asociados ambos a un mismo puerto
	Declaraciones CommunicationManager c	**c**: Instancia (objeto) de la clase CommunicationManager
	Especificación Semántica CommunicationManager() → CommunicationManager RendezVous() → RendezVous Receiver() → Receiver Sender() → Sender	

Continua en la próxima página

Tabla 5.15 –Finaliza de la página anterior

	Invoker() → Invoker Invoked() → Invoked DispatcherIntra() → DispatcherIntra	

Tabla 5.15: Definición Formal de la clase ComunicationManager

10 de Mayo de 2007		Versión 1.0
	2 RendezVouz( ): RendezVouz {Permite crear instancias del tipo RendezVouz }	
	Especificación de atributos: No Hay	
	Especificación sintáctica:	
1	RendezVous(Port: in port): RendezVous	*RendezVouz()*: Crea un agente de la clase RendezVouz
2	send (SyncMessage: in msg): MsgId	*send()*: Este método permite enviar un mensaje síncrono
3	receive (SyncMessage: out msg): MsgId	*receive()*: Este método permite recibir un mensaje síncrono
	Declaraciones	
	RendezVous v	**v**: Instancia (objeto) de la clase RendezVouz
	Port port	**port**: identificador único que abstrae una cola de mensajes según su orden de recepción
	SyncMessage msg	**msg**: mensaje
	MsgId msgid	**msgid**: identificador único de un mensaje
	Especificación Semántica	
	RendezVous(Port: in port) → RendezVous v.send (SyncMessage: in msg) → msgid v.receive (SyncMessage: out msg) → msgid	

Tabla 5.16: Definición Formal de la clase RendezVouz

La definición de clases, presentada anteriormente, permite que el MGS use los esquemas de comunicación siguientes: síncrona o "$Rendez-Vous$", asíncrona, y semántica "RPC" (*Remote Procedure Call*), como se describe a continuación:

1. Comunicación $Rendez-Vous$ (Síncrona). Se utiliza para situaciones en las que haya que garantizar correspondencia de orden entre emisión y recepción. Bajo los otros esquemas, el orden de recepción no necesariamente corresponde con el de emisión. Los mensajes síncronos se manipulan mediante dos instancias del agente $Rendez-Vous$. El constructor de esta clase acepta el **UID** del otro extremo (remitente o destinatario).
2. Comunicación Asíncrona. Los mensajes asíncronos usan dos agentes diferentes. El remitente debe instanciar un agente *Sender* mientras que el destinatario debe instanciar uno *Receiver*.
3. Comunicación RPC. La comunicación tipo RPC se modela mediante los agentes *Invoker* e *Invoked*. El destinatario debe colocarse a la escucha con

un agente *Invoked* y llamar al método *receive*, el cual se bloquea hasta recibir un requerimiento. El remitente instancia un agente *Invoker* y llama al método *invoke* para enviar el requerimiento.

Cuando un agente *Invoked* recibe un requerimiento, éste se desbloquea. Cuando se considere oportuno, el agente *Invoked* responderá mediante el método *reply* indicando el *MsgId* del mensaje al que se está respondiendo.

Puesto que el orden de recepción no necesariamente corresponde con el de emisión, la recepción del *reply* está separada bajo el método *receive*. El *reply* toma el *MsgId* del *request*. De esta manera, el remitente puede identificar el requerimiento del cual está recibiendo respuesta.

Un mensaje tiene dos facetas. La primera, representada por la clase *Message*, concierne al mensaje dentro de la actividad del agente interesado. En este contexto, hay cuatro especializaciones de *Message*:

1. *Sync_Message*: modela un mensaje de requerimiento correspondiente a una comunicación *Rendez – Vous* en la que los flujos del remitente y destinatario se bloquean hasta que no se consuma el mensaje.
2. *Async_Message*: modela un mensaje asíncrono; el remitente entrega el mensaje al sistema de comunicación y se desbloquea.
3. *Request_Message*: concierne a un mensaje de requerimiento (*request*) correspondiente a una comunicación RPC en la que el remitente se bloquea hasta recibir respuesta explícita del destinatario.
4. *Reply_Message*: concierne a un mensaje de respuesta del destinatario a un mensaje previo de requerimiento, para una comunicación RPC.

Una vez que el mensaje es entregado al sistema de comunicación, aparece la segunda faceta de manipulación de mensajes representada bajo la clase *MsgId*. Esta clase modela aspectos de un mensaje que se conocen después de que el sistema de comunicación haya sido informado del mensaje. Lo más representativo e importante de esta clase (*MsgId*) es que ella contiene un identificador único del mensaje. Este identificador ayuda a convalidar el mensaje e identificarlo unívocamente en las situaciones en las que puedan solaparse varios mensajes.

5.6. Codificación y Pruebas

Esta fase corresponde a la última fase de la metodología MASINA del Capítulo 3. Como primer paso es necesario definir la plataforma y los componentes de software que permitirán el desarrollo y ejecución del código que implementa al MGS. En caso de requerir servicios de tiempo real, debe considerarse el sistema operativo base que tenga esta capacidad, como por ejemplo RTLinux, mencionado anteriormente. La versión actual no implementa dentro en el MGS servicios tiempo real para el manejo de agentes o

para el uso director de primitivas de comunicación tiempo real por parte de éstos. Sin embargo, se usa un marco comunicación (ACE) de alto desempeño, extendible para soporte tiempo real. En caso de que un agente, en su lógica de operación, tenga restricciones de tiempo real, la implementación de esa característica se pueden llevar a cabo accediendo a las bibliotecas de servicio específico del sistema operativo tiempo real en uso.

La codificación y ejecución del MGS, diseñado bajo los lineamientos de la secciones 5.5.2 y 5.5.3, usando el lenguaje de programación C++, ha considerado los siguientes requisitos:

- Núcleo *Linux* de 32 bits: disponible en varias distribuciones. Puede usarse cualquier distribución.
- Colección de compiladores GNU: conocida como GCC (GNU Compiler Collection), es una suite de compiladores estándar para diversos sistemas operativos de código propietario ó abierto, particularmente Linux.
- Librería para manejo del lenguaje XML: es un lenguaje utilizado para almacenar datos de manera legible, utilizado para integrar información.
- Librería *OpenSSL*: es un paquete de herramientas de administración y bibliotecas que ayudan a implementar protocolos relacionados con la seguridad, y crear certificados digitales que pueden aplicarse a un servidor.
- Librería *NCurses*: es una biblioteca de programación que provee una interfaz de programación de aplicaciones (*Application Programming Interface*, API) que permite escribir interfaces basadas en texto.
- Librería *ACE 5.5.7*: por su denotación en inglés, *Adaptive Communication Environment*, ACE es un marco de trabajo orientado a objetos de código abierto, que provee un conjunto de clases para la implementación de comunicación concurrente, particularmente para el desarrollo de aplicaciones de alto desempeño y servicios de comunicación en tiempo real.
- Paquete *Libtool*: es una herramienta de programación libre usada para crear bibliotecas de software portables, esto es, que pueden ser compartidas bajo diferentes plataformas.

5.6.1. Implementación del MGS

El diseño del MGS, presentado en la Sección 5.5, fue codificado considerando la plataforma y componentes mencionados en los requisitos [6]. Un ejemplo de codificación en C++ de un archivo cabecera para la creación la clase *Locator*, que compone a la clase *AgentManager*, se muestra en la Tabla 5.17. La clase *Locator* tiene como objetivo proporcionar los métodos necesarios para la búsqueda de agentes. Es importante notar que *Locator* es usado por la clase *AgenteAdministradorAgentes* que implementa al agente AAA, en su método *localizarAgente*, como se especifica en la Tabla 5.12. Este método, a su vez, utiliza el método de búsqueda implícita *implicityFind* de la clase

Locator. El método *implicityFind* genera un mensaje del tipo *piggyback* [3], en el cual se indica la búsqueda de un determinado agente. En caso de darse varias búsquedas fallidas, este mensaje deviene en otro tipo de mensaje, en el cual se indica a los sitios que lo reciben, que detengan al agente buscado, en caso de que éste se encuentre en alguno de estos sitios.

locator.h

```
#ifndef __LOCATOR_H__
#define __LOCATOR_H__

#include <syscall/mgstypes.h>
#include <uid.h>
#include <syscall/syscalllocate.h>
#include <syscall/returncalllocate.h>
#include <syscall/makesyscall.h>
#include <coordenada.h>
#include <mgsbaselib.h>

//Lista de errores devueltos por Locator

# if !defined ( MGS_LOCATOR_NO_ERROR )
#       define MGS_LOCATOR_NO_ERROR 0
# endif /* MGS_LOCATOR_NO_ERROR */

# if !defined ( MGS_LOCATOR_ERROR_IRRECUPERABLE )
#       define MGS_LOCATOR_ERROR_IRRECUPERABLE -1
# endif /* MGS_LOCATOR_ERROR_IRRECUPERABLE */

# if !defined ( MGS_LOCATOR_ERROR_AGENTE_INEXISTENTE )
#       define MGS_LOCATOR_ERROR_AGENTE_INEXISTENTE -2
# endif /* MGS_LOCATOR_ERROR_AGENTE_INEXISTENTE */

class Locator
{
public:

/*Constructores y Destructor*/

Locator ();

~ Locator ();
```

Continua en la próxima página

[3] Transferencia de información que reserva el medio de comunicación a través de un acceso distribuido, para el envío de un mensaje completo.

Tabla 5.17 –Finaliza de la página anterior

```
/*Métodos de Transformación y Acceso */

Coordenada implicityFind ( const Uid& );
Coordenada weakFind ( const Uid& );
Coordenada strongFind ( const Uid& );

int lastError () const;

private:

int lastError_ ;
}

#endif /* __LOCATOR_H__ */
```

Tabla 5.17: Archivo locator.h

El archivo **locator.h** incluye otros archivos cabecera, que a su vez incluyen archivos propios de la biblioteca ACE:

1. El archivo **mgstypes.h**, cuyo objetivo es contener tipos de datos, macros y operaciones generales, creados para las clases de la biblioteca y componentes del Nivel Base del MGS.
2. El archivo **uid.h** es el archivo cabecera de la clase *Uid*, cuyo objetivo es solicitar la generación de identificadores únicos (UIDs).
3. El archivo **syscalllocate.h** es la cabecera de la clase *SysCallLocate*, cuyo objetivo es generar un *SysCall* para obtener una localización, a partir de un Uid. La clase *SysCall* genera una clase interfaz o base para todas aquellas que necesiten generar un mensaje, para ser transmitido a los procesos contenidos en el Nivel Base para que le sea atendido el servicio solicitado.
4. El archivo **returncalllocate.h** es la cabecera de la clase *ReturnCallLocate*, cuyo objetivo es primeramente recibir los parámetros del *SysCall* emitido y atender dicho servicio.
5. El archivo **makesyscall.h** es la cabecera de la clase *MakeSysCall*, su objetivo es ejecutar las llamadas de tipo *SysCall*, manejando estos objetos y enviando su mensaje a la cola SysCallQueue, contenida en un hilo (Thread).
6. El archivo **coordenada.h** es la cabecera de la clase *Coordenada*, cuya función es indicar la localización exacta de un ente en todo el dominio del sistema.
7. El archivo **mgsbaselib.h** es la cabecera de la clase *MgsBaseLib*, cuyo objetivo es inicializar los componentes necesarios para que las bibliotecas pertenecientes al Nivel Base funcionen correctamente.

Un fragmento del archivo **locator.cpp** se presenta en la Tabla 5.18, que muestra la codificación del método de búsqueda implícita *implicityFind()*, de la clase *Locator*.

locator.cpp

```
Coordenada Locator::implicityFind (const Uid& uid)
{
Coordenada coord;
this– >lastError_ = MGS_LOCATOR_NO_ERROR;
if (MGS_BASE_LIB::instance()– >isInit() )
{
    SysCallLocate sysLocate ( uid, MGS_LOCATE_IMPLICITY );
    ReturnCallLocate retLocate;
    MakeSysCall < SysCallLocate, ReturnCallLocate > make (sysLocate, retLocate);
    //Se obtiene el estado del resultado del servicio solicitado
    int result = retLocate.result ().resultado;
    //Si el estado es satisfactorio, se obtiene la Coordenada solicitada
    if ( result> 0 )
      //coord.ipCoordenada ( retLocate.result ().coordAt1 );
      coord = retLocate.result ().coord;
    else
    {
      if(result < 0)
      {
        this– >lastError_ = MGS_LOCATOR_ERROR_IRRECUPERABLE;
      }
      else
      {
        this– >lastError_= MGS_LOCATOR_ERROR_AGENTE_INEXISTENTE;
      }
    }
}
else
{
     ACE_DEBUG((LM_DEBUG, ACE_TEXT ("\n%D\n(%P|%t) Locator::Error Librería
no inicializada \n")));
    this– >lastError_= MGS_LOCATOR_ERROR_IRRECUPERABLE;
}
return coord; //Retorna la Coordenada solicitada
}
```

Tabla 5.18: Fragmento de archivo locator.cpp

5.6.2. Pruebas del MGS

La prueba de implantación del MGS puede hacerse en un ambiente local o distribuido. Para ello es necesario realizar el protocolo de instalación y operación del MGS, el cual se detallará en la Sección 5.7. Para la prueba en un ambiente local. La ejecución de la prueba distribuida se realizó en cuatro equipos, identificados como equipos *A*, *B*, *C* y *D*, configurados para formar una red de área local, cada uno con un sistema operativo *Ubuntu* versión

11.10 de 32 bits y memoria RAM de 1 Gb. También se probó en máquinas con la distribución Gentoo.

Básicamente, las pruebas están orientadas a la verificación de las capacidades de MGS para:

1. Crear agentes con identificadores únicos. Primeramente, se crearon todos los agentes que implementan al MGS, y luego se crearon agentes de un SMA genérico que requiere de los servicios del MGS, para poder cumplir sus objetivos.
2. Localizar agentes del nivel superior, y de cualquier agente por nombre y por UID.
3. Solicitar servicios entre agentes del nivel superior.
4. Solicitar servicios de localización de aplicaciones y de datos.
5. Enviar y recibir mensajes solicitando los servicios de localización anteriores de datos, aplicaciones y agentes.

Para verificar estas pruebas, se necesita diseñar un conjunto de agentes del nivel superior, esto es, los agentes asociados a una aplicación basada en SMA que requiere los servicios del MGS, como se ilustra en la Figura 5.1. En este caso, se diseñan cuatro agentes llamados Negocio, Aplicación, Repositorio y Especializado y se diseñan un conjunto de pruebas que incluye tareas de creación de agentes, localización de agentes, solicitud de servicios entre agentes del nivel superior y servicios de localización de datos y agentes, entre otros.

Algunos fragmentos de los reportes de pruebas se presentan a continuación, para el ambiente distribuido. Todos los detalles sobre las pruebas realizadas para la validación de integración del MGS con el nivel superior se encuentran en [8]:

1. **Plataforma (nodos configurados)**:
   c0a8011e-6763845e-4766ba7b-5b1b3000
   c0a8011e-4353d0cd-4766ba7b-5b725000
   c0a8011e-71f32454-4766ba7b-5bc94000
   c0a8011e-3a95f874-4766ba7b-5c29e000

2. **Creación del ACC (configuración del puerto de comunicación)**:
   DEBUG : Mon Dec 17 14:05:47 2007
   * se cumplio plazo
   * APLICACION: ./interfaz/src/AgenteControlComunicaciones.cpp
   * ACC::registrarPuerto()
   * LINEA : 144

   DEBUG : Mon Dec 17 14:05:47 2007
   * Puerto registrado
   * APLICACION: ./interfaz/src/AgenteControlComunicaciones.cpp
   * ACC::registrarPuerto()
   * LINEA : 161

3. **Pruebas de creación de agentes:**
   Agente Pruebas ejecutándose desde : 192.168.1.30
   Su UID es : c0a8011e-2d1d5ae9-4766ba7b-5b05c000

   + Primera fase de las pruebas : Solicitar la creación de los agentes de Nivel Superior

   Receptor : AAA
   Mensaje : crearAgente
   * Agentes gestores locales :
   - AAA A : c0a8013c-4e6afb66-4728bd9b-1368f000
   - AAA B : c0a80165-2d1d5ae9-38c9e88e-5a6ad000
   - AAA C : c0a8013c-2443a858-47448fde-36cb4000
   - AAA D : c0a8013c-0836c40e-4745a09b-a45a8000
   - AGA : c0a80165-79838cb2-38c9e88e-5c358000
   - AGD : c0a80165-54e49eb4-38c9e88e-5e1b8000
   - AGR : c0a80165-02901d82-38c9e88e-5fe17000

   Se envía mensaje de creación a : c0a8013c-4e6afb66-4728bd9b-1368f000
   303 : Se obtuvo una respuesta al crear el AgenteRepositorio :
   Respuesta : agenteCreado
   Uid : c0a80164-4353d0cd-4766c32f-845cc000

   Se envía mensaje de creación a : c0a80165-2d1d5ae9-38c9e88e-5a6ad000
   333 : Se obtuvo una respuesta al crear el AgenteEspecializado
   Respuesta : agenteCreado
   Uid : c0a8011e-5c482a97-4766ba7d-8f1cd000

   Se envía mensaje de creación a : c0a8013c-2443a858-47448fde-36cb4000
   362 : Se obtuvo una respuesta al crear el AgenteAplicacion
   Respuesta : agenteCreado
   Uid : c0a8011d-79838cb2-4766889e-1021f000

   Se envía mensaje de creación a : c0a8013c-0836c40e-4745a09b-a45a8000
   391 : Se obtuvo una respuesta al crear el AgenteNegocio
   Respuesta : agenteCreado
   Uid : c0a80120-4353d0cd-4766c056-82fe1000

   + Primera fase de las pruebas : Éxito

4. **Pruebas de localización de agentes:**
   + Segunda fase de las pruebas : Solicitar la localización de los agentes de Nivel Superior

   * Localizar agentes
   Receptor : AAA
   Mensaje : localizar

   Linea : 460 :: Se obtuvo una respuesta al localizar el AgenteRepositorio
   Respuesta : c0a80164-4353d0cd-4766c32f-845cc000 : CORRECTO

Linea : 460 :: Se obtuvo una respuesta al localizar el AgenteAplicacion
Respuesta : c0a8011d-79838cb2-4766889e-1021f000 : CORRECTO

Linea : 460 :: Se obtuvo una respuesta al localizar el AgenteEspecializado
Respuesta : c0a8011e-5c482a97-4766ba7d-8f1cd000 : CORRECTO

Linea : 460 :: Se obtuvo una respuesta al localizar el AgenteNegocio
Respuesta : c0a80120-4353d0cd-4766c056-82fe1000 : CORRECTO

+ Segunda fase de las pruebas (localización): Éxito

* Localizar agentes por sus nombres
Receptor : AAA
Mensaje : localizarAgenteNombre

Linea : 429 :: Se obtuvo una respuesta al localizar el AgenteRepositorio
Respuesta : c0a80164-4353d0cd-4766c32f-845cc000 : CORRECTO

+ Segunda fase de las pruebas (localización por nombre): Éxito

* Localizar agentes por Uid
Receptor : AAA
Mensaje : localizarAgenteUid

Linea : 490 :: Se obtuvo una respuesta al localizar el AgenteAplicacion
Respuesta : 192.168.1.29

+ Segunda fase de las pruebas (localización por Uid) : Éxito

5. **Pruebas de solicitud de servicios entre agentes del nivel superior:**
+ Tercera fase de las pruebas : Solicitar servicios a los agentes de Nivel Superior

* Solicitar Repositorio :
- Solicitando variables al AgenteRepositorio :
Linea : 509 :: Se obtuvo una respuesta al leer el AgenteRepositorio
Respuesta DATO1 : 9

* Solicitar Especializado :
Linea : 623 :: Se obtuvo una respuesta al leer el AgenteEspecializado
Respuesta : -1

* Solicitar Aplicación :
- Solicitando operaciones al AgenteAplicacion :
Linea : 567 :: Se obtuvo una respuesta al leer el AgenteAplicacion
Respuesta : 1

* Solicitar Negocio :
Linea : 668 :: Se obtuvo una respuesta al leer el AgenteNegocio
Respuesta : -18

+ Tercera fase de las pruebas : Éxito

6. **Pruebas de solicitud de servicios de localización de agentes:**
   + Cuarta fase de las pruebas : Solicitar servicio de localización:

   * Solicitar localizacion al AGA :
   Linea : 726 :: Se obtuvo una respuesta al Localizar aplicación
   Uid : c0a8011d-79838cb2-4766889e-1021f000
   Nombre : AgenteAplicacion
   Localizacion : 192.168.1.29
   Descripcion : SUMAR

   * Solicitar localizacion al AGD :
   Linea : 684 :: Se obtuvo una respuesta al Localizar DATO1 en repositorio
   Uid : c0a80164-4353d0cd-4766c32f-845cc000
   Nombre : AgenteRepositorio
   Localizacion : 127.0.0.1
   Descripcion : DATO1—DATO2—DATO3

   + Cuarta fase de las pruebas : Éxito

Los resultados de las pruebas del funcionamiento integrado fueron satisfactorios, pudiendo verificar:

- La creación local y remota de agentes con identificadores únicos.
- La comunicación local y remota entre agentes.
- La prestación de servicios de localización local y remota.
- La transferencia de mensajes.
- La interrelación entre los niveles base e interfaz de modo distribuido.
- El intercambio masivo de mensajes.

5.7. Instalación y operación del MGS

La instalación y operación del MGS puede hacerse en un ambiente local ó distribuido. Un esquema de funcionamiento distribuido se ilustra en la Figura 5.16 donde diferentes servidores (*server*) ejecutarían los procesos asociados a los módulos del nivel base y a los agentes del nivel interfaz. En este sentido, se tendría una instancia del MGS en cada servidor(nodo), cuyos servicios pueden ser solicitados por aplicaciones basados en SMA [7].

A continuación se presenta el protocolo de instalación y operación del MSG. Si el ambiente de ejecución es local, se procede a la instalación de las bibliotecas y del código del MGS en la máquina local, y si el ambiente es distribuido dicho procedimiento de instalación debe hacerse en cada nodo de la red, configurada para formar parte de la plataforma de ejecución/despliegue.

1. **Instalación de las bibliotecas.**

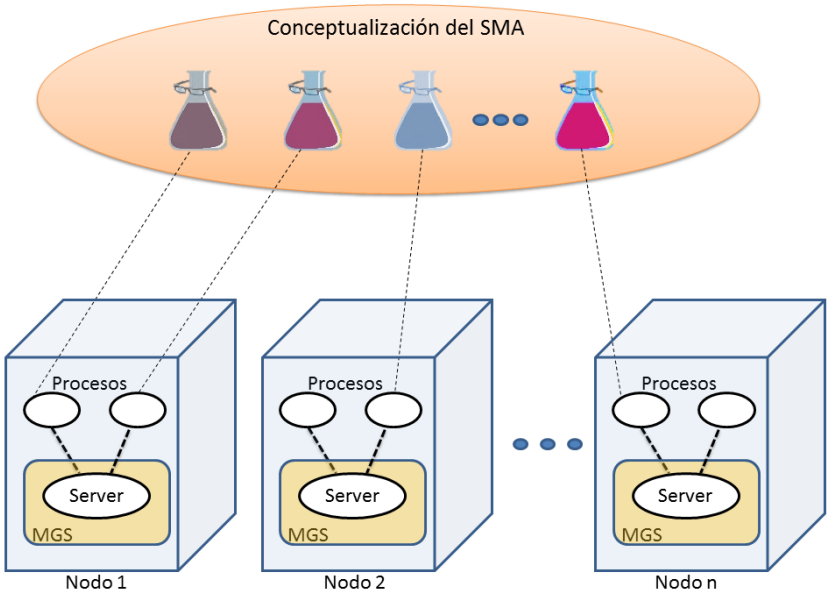

Figura 5.16: Ambiente de ejecución distribuido.

La instalación de las bibliotecas XML, OpenSSL y NCurses depende de la distribución Linux que este siendo usada. Por ejemplo, para la distribución **Gentoo** basta ejecutar:

emerge libxml2 openssl ncurses libtool

Para distribuciones **Ubuntu** y **Debian**, debe ejecutarse:

apt – get install libxml2 libssl-dev ncurses-dev libtool

Para la instalación de *ACE 5.5.7*, se siguen los siguientes pasos:

- Descargar la librería del siguiente enlace http://download.dre.vanderbilt.edu/previous_versions/ACE-5.5.7.tar.bz2
- Descomprimir el archivo ACE-5.5.7.tar.bz2 . Se creará un directorio llamado **ACE wrappers**.
- Dentro del directorio **ACE wrappers/ace/SSL**, modificar el archivo **SSL Context.cpp** cambiando la línea 239:

  SSL_METHOD *method = 0 ;

  por

  const SSL_METHOD *method = 0 ;

- Dentro del directorio **ACE wrappers**, crear una carpeta con cualquier nombre:

```
# mkdir foo
# cd foo
```

- Dentro del directorio recientemente creado, ejecutar:

  ```
  # ../configure
  ```

  Se crearán varios archivos, entre ellos un directorio llamado **ace/**.

- Ingresar al directorio **ace/** y modificar el archivo *Makefile*. Añadir el siguiente código entre las líneas 624 y 625:

  ```
  Select_Reactor.h \
  ```

  y añadir la opción *-fpermissive* a las opciones de compilación (CFLAGS y CXXFLAGS) en las líneas 1897 y 1907:

  ```
  CFLAGS = -g -O2 -pipe -fpermissive
  CXXFLAGS = -W -Wall -Wpointer -arith -g -O2 -pthread -pipe -O3 -fpermissive
  ```

- En el directorio **ace/**, ejecutar:

  ```
  # make && make install
  ```

2. **Instalación del MGS.**
   Una vez instaladas las bibliotecas necesarias para la ejecución de los archivos que implementan el MGS, se procede a la instalación de dichos archivos, los cuales se asumen comprimidos en un archivo *.tar*, siguiendo los siguientes pasos:

   - Descomprimir el archivo *MGS.tar.gz* . Se creará el directorio **Archivos-Fuentes** [4].
   - Una vez descomprimido, modificar los archivos *Makefile* ubicados en los subdirectorios **clientes**/, **lib/**, **mapperp/** y **daemon/** del directorio **Archivos-Fuentes/base/trunk/lib**. Agregar a las macros LDFLAGS y CXXFLAGS la opción - *-tag*. Esto es necesario para las versiones más recientes del compilar GCC.
   - Crear el directorio **/etc/mgs**. Esto debe ser ejecutado como superusuario (*root*):

     ```
     # mkdir/etc /mgs
     ```

   - Dentro del directorio **Archivos-Fuentes/interfaz**, ejecutar:

     ```
     # sh install.sh
     ```

     Este *script* se encarga de compilar el nivel base, crear la estructura de directorios necesaria para instalar el sistema, compilar el nivel interfaz y los agentes de nivel superior. En este caso, los agentes del nivel

[4] La versión de código disponible garantiza la creación de dicho directorio

superior son los asociados a las aplicación basadas en SMA, según la arquitectura básica de referencia de la Figura 5.1. En el versión actual, el MGS contiene los archivos del nivel superior usados para la prueba descrita anteriormente.

3. **Operación del MGS.**
Una vez instalado el MGS, entonces se procede a ponerlo en ejecución. Para esto se ejecuta el programa *server*, en cado nodo que formará parte de la plataforma; este programa se encarga de instanciar los niveles interfaz y base del MGS, ver la Figura 5.16. Este proceso, deja la plataforma operativa, lista para dar servicio a sistemas del nivel superior.
Luego se ejecuta el archivo, ó los archivos, que generan los procesos asociados a los agentes del nivel superior, esto es, los *procesos* de la Figura 5.16.
A continuación se detallan los pasos para la operación local y distribuida, que deben ser llevados a cabo antes de poner en operación el MGS.

Operación local:
Como ya se ha mencionado, se asume que el protocolo de instalación de las bibliotecas, y del MGS, ya ha sido realizado. Se siguen, entonces, los siguientes pasos:

- Modificar el archivo **/etc/mgs/base/sysconfig.xml** la directiva:

  <name> LocalIPAddress </name>
  <value> 192.168.1.60 </value >

  sustituyendo dicha dirección IP, por el valor actual de IP del equipo local.
- Modificar en el archivo **/etc/mgs/AAAConocidos.xml** el valor de la IP por el valor actual sustituido previamente. El archivo *AAAConocidos.xml* es un archivo de configuración que permite al agente AAA del MGS conocer el IP asociado al nodo donde se está ejecutando el MGS.
- Ingresar al directorio **Archivos-Fuentes/base/trunk/lib/bin** y ejecutar el programa que pone en operación al MGS:

  # ./server

- En otra consola, ingresar al directorio **/etc/mgs/agentes**, y ejecutar el archivo que solicita la creación (activación) de todos los agentes del nivel superior [5]:

  # ./AgentePruebas

  Dado que pueden existir diferentes SMAs requiriendo los servicios del mismo MGS, cada uno de estos SMA puede tener su propio archivo

[5] En este caso de la prueba, se ha llamado AgentePruebas

de activación y, en consecuencia, pueden ejecutarse varios archivos de creación de SMAs, sucesivamente.

De esta manera, ya se encuentra en operación tanto el MGS como el SMA, ó los SMAs, que solicita, ó solicitan, los servicios del MGS para cumplir sus objetivos.

Operación distribuida:
Al igual que para el caso local, se asume que el protocolo de instalación de las bibliotecas y del MGS ya ha sido realizado en cada nodo de la plataforma. Se ejecutan los siguientes pasos:

- Actualizar el archivo **/etc/mgs/base/sysconfig.xml**, para cada nodo de la plataforma, con su valor de IP como se indicó en el procedimiento local.
- Copiar los archivos contenidos en:
  el directorio Archivos-Fuentes/interfaz/archivos/EquipoX (donde X es el valor que identifica a cada nodo de la plataforma) al directorio **ArchivosFuentes/interfaz/xml** del nodo correspondiente.
- En cada nodo, se debe ajustar el valor de las IP de los nodos que conforman la plataforma, en el archivo ubicado en el directorio **Archivos-Fuentes/interfaz/xml/AAAConocidos.xml**, como se hizo en el procedimiento local. Es importante resaltar que, en el caso distribuido, este archivo de configuración permite que el agente AAA de cada nodo conozca el IP de todos los nodos donde se estará ejecutando el MGS y, por consiguiente, conozca todos los agentes AAA de la plataforma (Federación de AAAs).
- En cada nodo, ingresar al directorio:
  Archivos-Fuentes/base/trunk/lib/bin y ejecutar:

  ```
  # ./server
  ```

- En cada uno de los nodos, ingresar al directorio **/etc/mgs/agentes**, y ejecutar el arhivo de activación de los agentes que se ejecutarán en él. Cabe resaltar que el MGS provee un servicio de arranque distribuido, mediante el cual se puede iniciar todo el SMA desde un solo nodo. Para ésto, el proceso de inicio (disparador) solicita la ejecución de los agentes necesarios en los nodos de la plataforma. Por ejemplo, en el caso de la prueba presentada, se usa el esquema del disparador para todo los agentes; así se ejecutará sólo en un nodo:

  ```
  # ./AgentePruebas
  ```

Los dos últimos pasos tienen el mismo objetivo que en la operación local, esto es, poner en operación en cada nodo de la plataforma, el MSG y los agentes del nivel superior que se ejecutarán en dicha plataforma con el soporte del MGS. Se resalta nuevamente, que en cada nodo puede

ejecutarse un SMA, o varios SMAs, de nivel superior diferente, cada uno creado a través de su propio archivo de activación.

Referencias

1. J. Aguilar, L. León, A. Ríos, M. Cerrada, F. Hidrobo, F. Narciso, G. Mousalli, D. Hernández, and I. Besembel. Especificación del Medio de Gestión de Servicios. Informe Técnico. Proyecto de desarrollo de la ingeniería conceptual de software sobre la plataforma NetDAS, ULA-FUNDACITE, Venezuela, Septiembre 2005.
2. J. Aguilar, L. León, A. Ríos, M. Cerrada, F. Hidrobo, F. Narciso, G. Mousalli, D. Hernández, and I. Besembel. Ingeniería de Implantación del Medio de Gestión de Servicios. Informe Técnico. Proyecto de desarrollo de la ingeniería conceptual de software sobre la plataforma NetDAS, ULA-FUNDACITE, Venezuela, Octubre 2005.
3. J. Aguilar, A. Ríos, F. Hidrobo, and L. León. An architecture for industrial automation based on intelligent agents. *WSEAS Transactions on Computers*, 4(12):1808–1815, 2005.
4. 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.
5. Foundation for Intelligent Physical Agent. Specification. http://www.fipa.org, 2004.
6. Cooperativa Guatire Gas. Manual de Referencia del Medio de Gestión de Servicios. Informe Técnico. Proyecto de seguimiento, control y evaluación de la fase de desarrollo del Medio de Gestión de Servicios de la plataforma NetDAS 2.0., ULA-FUNDACITE, Venezuela, Diciembre 2007.
7. Morillo H. Reporte de instalación y pruebas del Medio de Gestión de Servicio. Informe Técnico, CEMISID-ULA, Venezuela, Junio 2012.
8. L. León, A. Ríos, F. Hidrobo, J. Barrios, L. Rojas, J. Alvarez, J. Altamiranda, and C. Mejía. Reporte de Validación de Pruebas de Integración del Nivel Interfaz y Nivel Base. Informe Técnico. Proyecto de seguimiento, control y evaluación de la fase de desarrollo del Medio de Gestión de Servicios de la plataforma NetDAS 2.0., ULA-FUNDACITE, Venezuela, Diciembre 2007.
9. Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems: Principles and Paradigms (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

Capítulo 6
EDISMA: Entorno de Desarrollo Integrado para la construcción de SMA

Resumen: En este capítulo se describe un Entorno de Desarrollo Integrado para la construcción de SMA, que se ha identificado con el nombre de EDISMA.

6.1. Introducción

En la Sección 4.1 se describen algunos de los elementos para facilitar el diseño y puesta en operación de SMA. Allí se expresa la necesidad de disponer de herramientas de apoyo en las etapas de construcción e implantación de los SMA.

Tal como ha sido abordado en la Sección 4.5, existen IDEs para el desarrollo de SMA, cada una con sus fortalezas y debilidades. El uso de una de ellas tiene su mayor dificultad en la dependencia de la interpretación de los estándares bajo los cuales está fundamentado el sistema inteligente, y del sistema de gestión de servicios construido para dar soporte al SMA.

En este sentido, en el marco de un proyecto general, que incluya los aspectos metodológicos, presentados en el Capítulo 3, y el soporte de operación de SMA, descrito en el Capítulo 5, se propone la implantación de un IDE que permita crear agentes mediante una interfaz gráfica. Este entorno debe permitir satisfacer los requerimientos metodológicos, usando los servicios disponibles en el MGS. Por consiguiente, este capítulo describe los elementos más importantes del Entorno de Desarrollo Integrado para la construcción de SMA (**EDISMA**).

Para el desarrollo de EDISMA se utilizó el método *White-watch* [4, 12], el cual permite garantizar uniformidad, consistencia, facilidad de integración y calidad de los componentes arquitectónicos del entorno; en un todo de acuerdo con las especificaciones generales del proceso de ingeniería de

software [14]. Con el objetivo de probar las capacidades de EDISMA, se ha desarrollado un caso de estudio que se presenta al final de este Capítulo.

El objetivo de EDISMA es proveer un IDE que permita crear agentes mediante una interfaz gráfica. Para ello, se toma como principio la teoría de SMA presentada en el Capítulo 1, y como estrategia metodológica de base, para el diseño de los agentes, la descrita en el Capítulo 3.

Así, con EDISMA se pretende facilitar la creación de agentes modelados a través de MASINA, partiendo de intentos previos del mismo tipo presentados en [3] y [5]. Pero además, EDISMA despliega los agentes sobre el MGS presentado en el Capítulo 5. De esta manera, se intenta disminuir el tiempo de implantación de los detalles de bajo nivel, necesarios para proveer la operación funcional de un agente. Es decir, los procesos de diseño, desarrollo y despliegue serán transparente hasta cierto punto, obteniéndose un código fuente sobre el cual varias instancias (en este caso, los agentes) son integrados para implantar un SMA. Así, esta herramienta incluye la generación de código fuente en lenguaje C++, con las llamadas a la biblioteca del MGS. Adicionalmente, los códigos generados, una vez adaptados a las particularidades del sistema a implantar, se compilan para ser ejecutados en la plataforma presentada en el Capítulo 5, inicialmente introducida en [1].

Como se comentó, es necesario realizar el diseño de los agentes; como parte de los productos de EDISMA se obtienen las plantillas de la metodología MASINA en un formato digital estándar, generando la documentación de los modelos para todos los agentes del SMA en formato XML. Este formato representa un código intermedio que puede ser usado como insumo para diversas funcionalidades, entre éstas, la verificaciones de modelos y la creación de códigos fuentes en diferentes lenguajes de programación.

Para cada agente que conforma el SMA se generan archivos en lenguaje C++: un archivo cabecera con extensión .h , un archivo con el cuerpo con extensión .cpp y un archivo principal que contiene la función (*main*) que ejecuta al agente. Además, para el SMA se genera un archivo con el guión de compilación (*makefile*), y un disparador que arranca la ejecución del SMA sobre la plataforma computacional.

6.2. Especificaciones Generales

Para la creación de un SMA con EDISMA, se toman en cuenta los elementos descritos en la Sección 3.3, es por ello que inicialmente se debe llevar a cabo el diseño con MASINA, especificando los modelos de agente, tarea, comunicación y coordinación establecidos en ella.

En la infografía mostrada en la Figura 6.1 se describe el proceso general que permite implantar un SMA utilizando EDISMA:

1. *Diseño*: Esta fase comprende las tres primeras etapas de la metodología, descritas en la sección 3.4, de donde se generan los modelos del SMA.

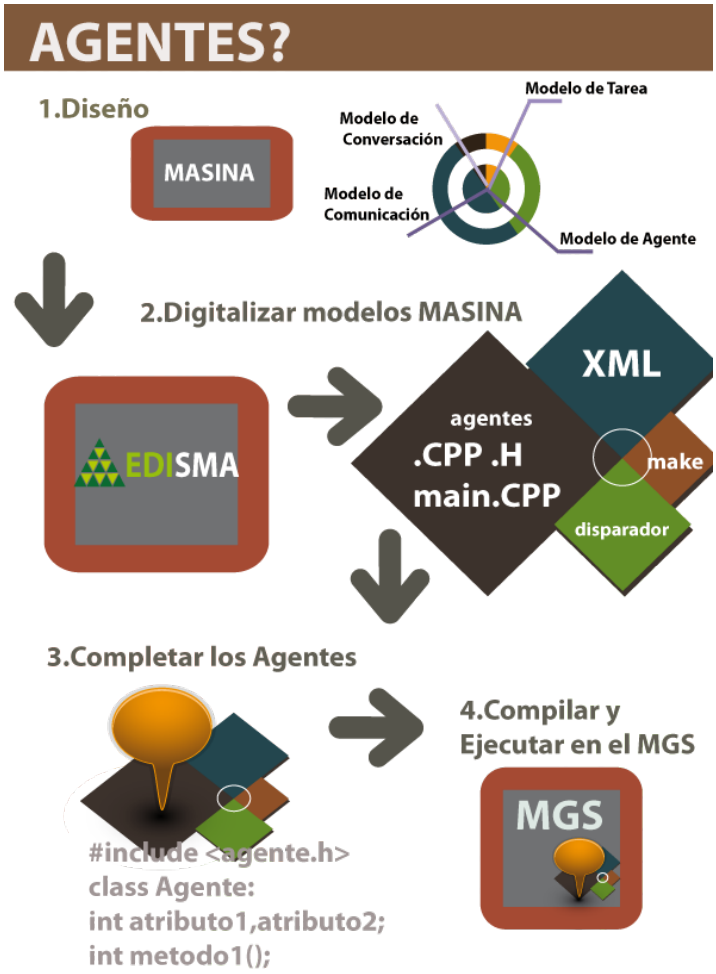

Figura 6.1: Creación de un SMA con EDISMA.

2. *Digitalización de modelos*: se produce la representación intermedia de los modelos en XML y se generan los códigos fuentes que implantan al SMA.
3. *Completar Agentes*: Mediante una módulo especial de EDISMA, un editor, se incorporan características especiales al código fuente.
4. Finalmente, se compilan los códigos fuentes para obtener el conjunto de programas que serán ejecutados en la plataforma, con el soporte del MGS (presentado en la Capítulo 5).

EDISMA, da soporte a los pasos 2 y 3 del proceso de construcción.

EDISMA genera los modelos especificados en MASINA en lenguaje C++, aunque para efectos de documentación y para dar soporte a otras funcionalidades, también genera una representación intermedia en XML. Los archivos generados por EDISMA poseen la especificación de los métodos (archivo cuerpo) y los archivos que ejecutan a cada agente (archivo principal). El código generado contiene la definición y especificación de los métodos que permiten implantar los modelos de la metodología. Por otro lado, estos códigos incluyen los aspectos de integración con la biblioteca del MGS (Sección 5.4.2), y las especificaciones de los puntos donde debe incorporarse algoritmos particulares de cada agente. En este sentido, EDISMA no genera un código final compilable, lo que hace es acercar al usuario a una versión ejecutable del SMA bajo diseño.

Siguiendo la definición de IDE, presentada en la Sección 4.5, el objetivo principal de EDISMA es "*Construir un entorno para implementar SMA*". Los objetivos específicos, son alcanzados mediante la ejecución de los procesos correspondientes:

- Proveer una IGU para modelar los agentes, según MASINA.
- Implementar un componente para especificar los modelos. Con este componente es posible almacenar los modelos en un formato digital estándar, para ser usado tanto en la documentación de éstos, como en la generación de código fuente en un lenguaje de programación de alto nivel.
- Generar la estructura del SMA, integrando los códigos generados con la biblioteca del MGS, para definir los aspectos vinculados a la comunicación y coordinación entre los agentes, lo que permite construir la estructura básica del SMA.
- Proveer un mecanismo que permita modificar los códigos generados.

6.3. Arquitectura de EDISMA

Como se aprecia en la Figura 6.2, EDISMA está conformado por cuatro componentes: la IGU, el generador XML, el editor, y el generador C++. La arquitectura basada en componentes permite lograr un adecuado nivel de independencia entre los procesos básicos que se desarrollan durante la construcción de un SMA. Por otra parte, en la versión actual se cuenta con un editor externo de diagramas UML [16] que permite integrar diagramas de actividades (asociados a tareas de agentes) y diagramas de secuencias (procesos de comunicación) durante la fase de especificación de los modelos correspondientes en la metodología. Además, se observa que existe una estrecha relación de EDISMA con el MGS, puesto que los servicios que presta éste son accedidos por los agentes a través de su biblioteca.

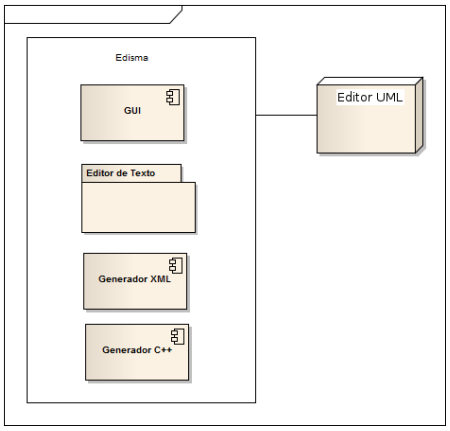

Figura 6.2: Componentes de EDISMA.

6.3.1. *Interfaz Gráfica de Usuario (IGU)*

Es el componente que permite recibir los parámetros necesarios para la construcción de los agentes que forman parte de un SMA. La obtención de los datos se lleva a cabo mediante formularios, que el usuario llena, suministrando secuencialmente los elementos necesarios para la construcción de los modelos de MASINA. Existe una secuencia especificada por la propia metodología: Modelo de Agente, Modelo de Tareas, Modelos de Comunicación y Modelo de Coordinación. Como se puede observar de la sentencia anterior, el Modelo de Inteligencia de MASINA no es considerado en la actual versión de EDISMA. Además, este componente permite invocar la inicialización de la aplicación Umbrello [8, 9], para generar e incorporar los diagramas de actividades y de secuencia empleados en el modelo de tareas y en el modelo de comunicación, respectivamente. Con el fin de garantizar la portabilidad de la herramienta, su facilidad de integración con el lenguaje de desarrollo, y tomando en cuenta aspecto de rendimiento, la IGU ha sido desarrollada en C++ con QT4 [17, 11]. Este componente representa, de modo gráfico, los siguientes elementos:

- SMA: viene representado por el espacio gráfico donde se generan los distintos modelos. Esto permite que el usuario vaya visualizando su sistema, a medida que lo va construyendo. En este espacio gráfico, se muestran los agentes y las interacciones del SMA.
- Agentes: se representan mediante una figura geométrica, la cual permite la configuración de sus elementos gráficos.
- Relaciones entre los agentes: cuando se genera un modelo de conversación, se genera una línea que une a los agentes participantes.

- Acciones: se define como acción, cualquier evento realizado en el espacio gráfico; dichas acciones son especificadas por orden de ejecución, dentro del entorno gráfico.
- Archivos: son los elementos generados por otros componentes de EDISMA, éstos pueden ser visualizados mediante un árbol de gestión de archivos integrado en el entorno gráfico.

La especificación (localización, colores, interacción, etc.) de los elementos gráficos que componen la IGU, se presenta en la Sección 6.5, en la que este componente se describe detalladamente.

6.3.2. Generador de XML

Extensible Markup Language (XML), es un sencillo formato basado en texto para la representación de información estructurada: documentos, datos, libros, transacciones, entre otros. Fue derivado de un formato antiguo estándar llamado SGML (ISO 8879), con el fin de ser más adecuado para uso Web [13, 15].

Una vez obtenidos los parámetros, mediante la IGU, el generador de XML crea archivos con esa extensión para los modelos especificados en MASINA. Además, se crea un archivo índice, el cual está asociado a cada uno de los modelos indicados anteriormente; este índice es de gran utilidad porque contiene todos los elementos que van a conformar el SMA, y se utiliza para la verificación y consistencia del SMA. A través del generador de XML se obtiene un código intermedio, que puede ser usado como insumo para otras funcionalidades, entre éstas, la verificación de los modelos y la generación de código fuente en diversos lenguajes (siempre y cuando éstos provean compatibilidad con el MGS).

6.3.3. Editor de texto

Es el componente que permite modificar los archivos generados por EDISMA, compuestos únicamente por texto sin formato, conocidos comúnmente como archivos de texto o texto plano. Con este componente es posible realizar las operaciones básicas de edición, disponibles en los editores de textos, tales como *emacs* [7, 6] y el *block de notas* [10]. Mediante el editor se pueden incorporar, directamente en los códigos generados, instrucciones específicas que permitan agregar aspectos que la metodología no puede capturar (por ejemplo, funciones particulares dentro de alguna tarea del agente).

6.3.4. Generador de C++

El Generador C++ recibe como insumo los archivos XML, creados con el componente Generador de XML. Los archivos obtenidos en este componente implementan la estructura y los elementos que permiten integrarlos en el MGS, para los agentes que forman parte del SMA. Este componente podría tener tantas versiones como lenguajes destinos se deseen, para la versión actual de EDISMA solamente se generan códigos en C++. Este componente integra los códigos generados con la biblioteca del MGS, haciendo las invocaciones requeridas a esta biblioteca, según lo requerido en cada modelo. Los archivos generados por este componente son:

- Archivo cabecera: es un archivo con extensión .h, y está conformado por una clase compuesta de atributos y procedimientos (solo las declaraciones) que representa al agente.
- Archivo del cuerpo: es un archivo con extensión .cpp, y está conformado por la implementación de cada uno de los procedimientos declarados en el archivo cabecera. EDISMA genera marcas especiales para indicarle al programador los puntos particulares donde se requiere agregar código específico. Esta funcionalidad se incorpora usando las características generales de los diagramas de actividades.
- Archivo principal: es un archivo con extensión .cpp, que está conformado por la implementación del archivo principal de cada uno de los agentes.

Es importante destacar que, tal y como se especificó en el alcance, es necesario que el usuario culmine la implementación de los mismos, de manera particular, lo referido a los métodos que implementan las tareas específicas de cada agente y la estructura particular de los mensajes que intercambian los agentes.

6.4. Esquema de operación

Para el desarrollo de EDISMA se ha utilizado:

- Qt Creator: en su versión 2.4.1 basada en Qt 4.7.4 (32 bit) la cual se encuentra en un repositorio web. (http://qt-project.org/downloadsqt-creator).
- QtXml: este módulo ofrece un lector y escritor de documentos XML. Este módulo se encuentra disponible en el mismo repositorio web de Qt Creator.

Sin embargo, el esquema de operación queda establecido a través de la vista de despliegue, ésta explica cómo los componentes ejecutables (código objeto) y los elementos en tiempo de ejecución se deben instalar en la plataforma de operación. La Figura 6.3 muestra el diagrama de despliegue que

describe la arquitectura física del sistema durante la ejecución, en términos de:

- Nodos: son objetos físicos que representan algún tipo de recurso computacional donde se ejecutan o despliegan los componentes de software.
- Componentes de despliegue: son piezas de software que conforman un sistema ejecutable.

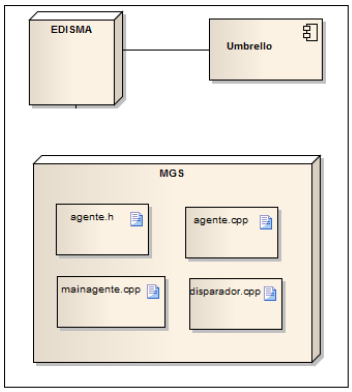

Figura 6.3: Diagrama de despliegue de EDISMA.

EDISMA funciona en un esquema fuera de línea, por lo tanto, el nodo de ejecución de EDISMA no tiene que formar parte de la plataforma de operación del SMA, de hecho, EDISMA puede ejecutarse en diferentes sistemas operativos (tanto en WINDOWS como en LINUX), siempre que se tenga instalado la versión correspondiente de QT (4.7.4); mientras que para la versión actual del MGS solamente funciona en LINUX. Así, los elementos EDISMA y Umbrello, mostrados en la parte superior de la Figura 6.3 se ejecutan en un NODO; mientras los productos generados se ejecutan en diferentes nodos de la plataforma, según la configuración pre-establecida, siguiendo el esquema presentado en la Sección 5.7.

6.4.1. Interacción con el MGS

La Figura 6.4 permite visualizar la relación que existe entre los productos que genera EDISMA con la plataforma de implantación del SMA. Los elementos que EDISMA genera están conformados por una serie de archivos, que pueden ser compilados utilizando la biblioteca del MGS. Posteriormente deben ser distribuidos y ejecutados sobre la plataforma de despliegue.

Como se establece en la Sección 5.7, la plataforma debe ser configurada, y en cada nodo debe estar en ejecución el programa ejecutivo del MGS.

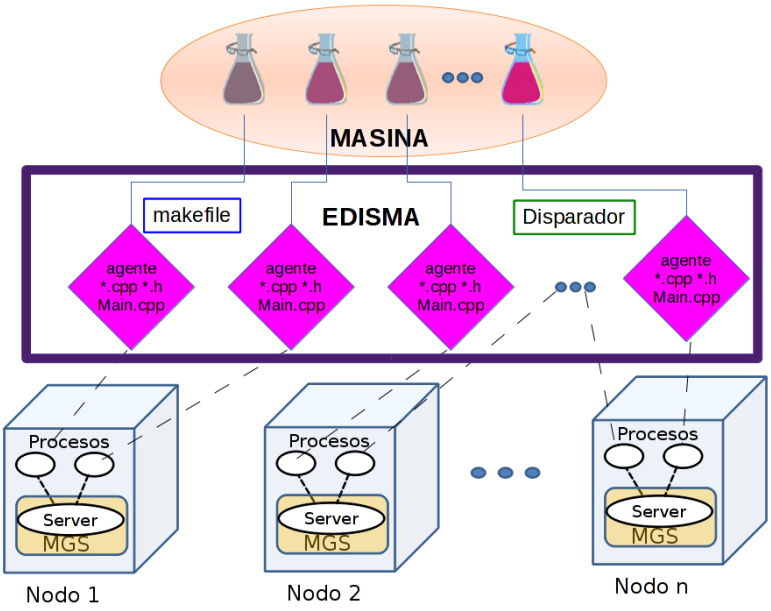

Figura 6.4: Productos de EDISMA y relación con la plataforma

6.4.2. *Manejo de Datos*

EDISMA, a pesar de ser un sistema con diversas funcionalidades, formalmente no requiere un manejador de base de datos para la ejecución de sus tareas. Sin embargo, para el manejo de los datos posee un repositorio de archivos que permite llevar a cabo la implementación de los agentes descritos a continuación:

- archivoCabecera: proporciona la estructura que deben tener los archivos con extensión .h, que permiten realizar la definición de las clases que caracterizan a los agentes.
- archivoCuerpo: provee la estructura que deben tener los archivos con extensión .cpp, que posee la especificación de los métodos definidos en el archivo cabecera.
- archivoPrincipal: proporciona la estructura que deben tener los archivos con extensión .cpp, pero que contienen el método **main**.

Para la implantación del makefile y del disparador, al igual que para los agentes, se dispone de los archivos necesarios que poseen la estructura básica de éstos (plantillas).

6.5. Interfaz de EDISMA

A continuación se presentan las características estructurales y de percepción visual de la IGU de EDISMA, ya que constituye el único perfil que ésta posee para el usuario. También, se describe el comportamiento de la interfaz, presentando además, las pantallas de captura de datos para cada una de las funcionalidades contempladas.

6.5.1. Funcionalidades

La funcionalidades de la IGU se especifican como servicios o acciones que el usuario puede ejecutar a través de ésta, estos servicios son:

- Crear un SMA. En el contexto de la IGU, se asocia un SMA con un proyecto de software. Así, en la IGU cada vez que se hace referencia a un proyecto se está hablando de un SMA; crear un SMA significa crear un nuevo proyecto.
- Crear un agente.
- Crear un modelo de tarea a un agente.
- Crear un modelo de conversación entre agentes.
- Crear un modelo de comunicación entre agentes.
- Abrir un SMA elaborado creado previamente (Abrir un proyecto existente).
- Modificar posición, tamaño y color de los elementos gráficos que representan a los agentes.
- Crear los archivos que permiten la implantación de los agentes en el MGS.
- Editar los archivos creados por EDISMA.

6.5.2. Estructura de la IGU

La interfaz de EDISMA está conformada por los siguientes elementos (Ver Figura 6.5):

- Área 1 - Menú Principal: se encuentra en la sección superior de la IGU de EDISMA, permite llevar a cabo acciones sobre los proyectos.

- Área 2 - Menú Secundario: está debajo del Área 1, en la sección superior de la IGU de EDISMA, y permite llevar a cabo acciones básicas sobre los proyectos: crear proyecto, abrir proyecto, generar archivos XML, etc. La diferencia es que a este menú, con el menú principal, es que se accede a través de iconos.
- Área 3 - Pila de Acciones: es un elemento gráfico donde se puede observar la serie de acciones llevadas a cabo en el entorno de EDISMA; por ejemplo, sí se agrega un agente en dicha pila se mostrará un mensaje indicando la acción.
- Área 4 - Gestor de Archivos: es el área situada en la sección inferior derecha de la IGU, permite visualizar los archivos que se generan de los modelos de MASINA y/ó de la implementación de los agentes.
- Área 5 - Entorno Gráfico: está conformado por un espacio que permite visualizar los elementos gráficos que conforman al SMA, a partir del cual se puede observar tanto los agentes como las relaciones entre éstos. Esta área se encuentra ubicada en el centro de la IGU.
- Área 6 - Barra Lateral: localizada en la izquierda de la IGU principal. Está compuesta por una serie de iconos que representan las funcionalidades de: crear Agente, crear Modelo de Tarea, crear Modelo de Conversación, crear Modelo de Comunicación, eliminar un agente y manipular el color de los elementos gráficos que representan a los agentes. Además, de manera gráfica, es posible seleccionar un agente existente y copiarlo para duplicarlo, luego se puede editar para cambiar las propiedades.

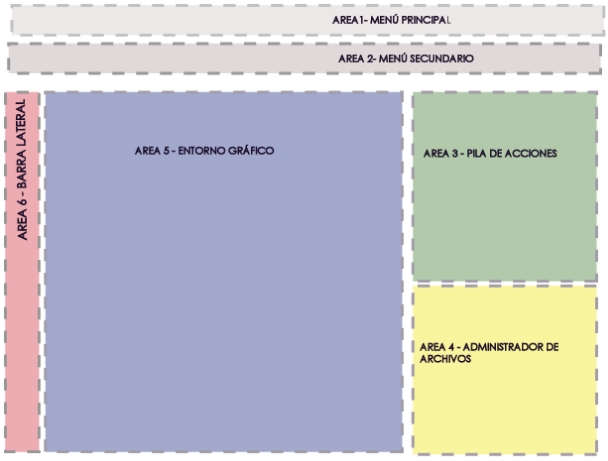

Figura 6.5: Estructura de la interfaz.

6.5.3. *Interacción gráfica*

La IGU de EDISMA tiene la particularidad de desglosar la información de un modo progresivo, es decir, inicialmente el usuario podrá acceder a funcionalidades básicas sobre las gestiones de proyectos. Después, en caso de iniciar uno nuevo se presentará la opción de crear un agente, que es el proceso inicial cuando se crea un SMA. En el caso de abrir uno existente se despliega en el área correspondiente los elementos de ese proyecto, para tomar acciones de modificar, eliminar o agregar elementos (agentes, interacciones, etc.)

La Figura 6.6 muestra la interfaz de EDISMA, al inicio de su ejecución, puede observarse que sólo se encuentran activas las opciones de generar un nuevo proyecto, abrir un proyecto, salir ó iniciar la creación de un agente (aparecen señaladas en la Figura 6.6). Las opciones activas están a color y las inactivas en gris.

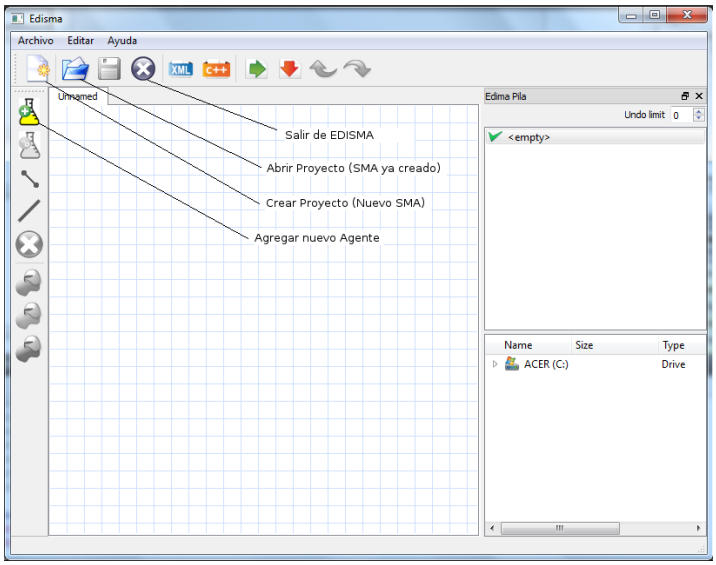

Figura 6.6: Interfaz principal de EDISMA.

Al iniciar la creación de un agente, se despliega una ventana, como la que se muestra en la Figura 6.7, que permite al usuario proporcionar los elementos que conforman el Modelo de Agente, de un modo secuencial. El usuario debe ir llenando el formulario con los aspectos básicos del agente, para luego continuar con los servicios, objetivos, restricciones y capacidad, de acuerdo a MASINA, a los cuales se accesan a través de las pestañas de esa ventana.

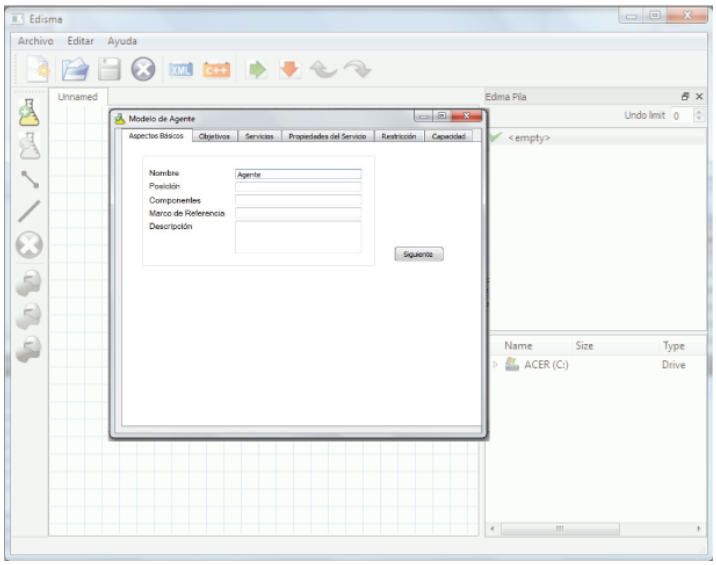

Figura 6.7: Interfaz para crear un Modelo de Agente.

El modelo de tarea puede especificarse a través de la opción respectiva, presionando el segundo icono de la barra lateral izquierda. Esa acción desplegará una ventana, como se señala en la Figura 6.8, donde el usuario proporciona, a través de un formulario, cada uno de los parámetros necesarios para crear dicho modelo. Esa ventana tiene dos pestañas, una para crear una nueva tarea (Agregar Tarea) y otra para acceder a las tareas que ya han sido creadas (Tarea).

Las Figuras 6.9 y 6.10 muestran las ventanas de los modelos de conversación y de comunicación, respectivamente. Al igual que en los modelos anteriores, en ellos se introducen, mediante formularios, los parámetros que los conforman. En ambos casos, la ventana que se despliega tiene dos pestañas, una para crear un nuevo modelo y otra para acceder a los modelos que ya han sido creados, en caso de existir.

6.6. Usando EDISMA

La Figura 6.11 describe los pasos que se siguen para usar EDSIMA en la implantación de un SMA. En cada paso se generan productos que sirven de insumos al paso siguiente.

A modo de ilustrar como se realiza la construcción de un SMA con EDISMA, se especifica a continuación el conjunto de pasos que son necesarios

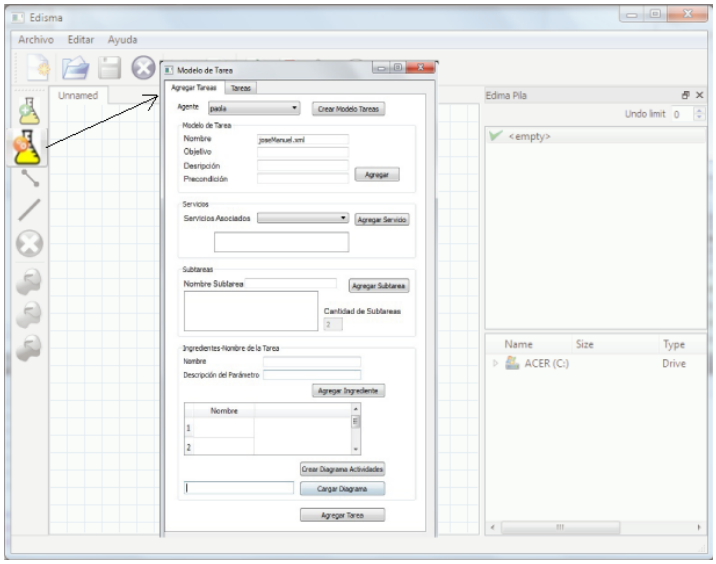

Figura 6.8: Interfaz para crear un Modelo de Tarea.

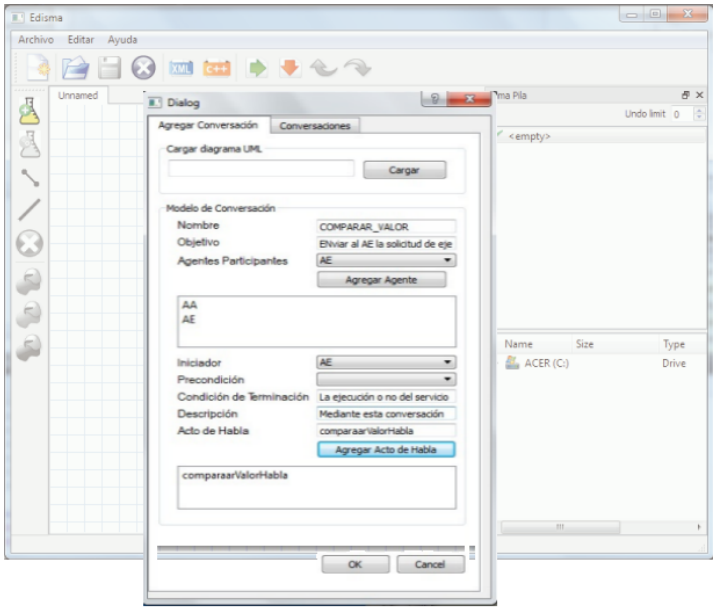

Figura 6.9: Interfaz para crear un Modelo de Conversación.

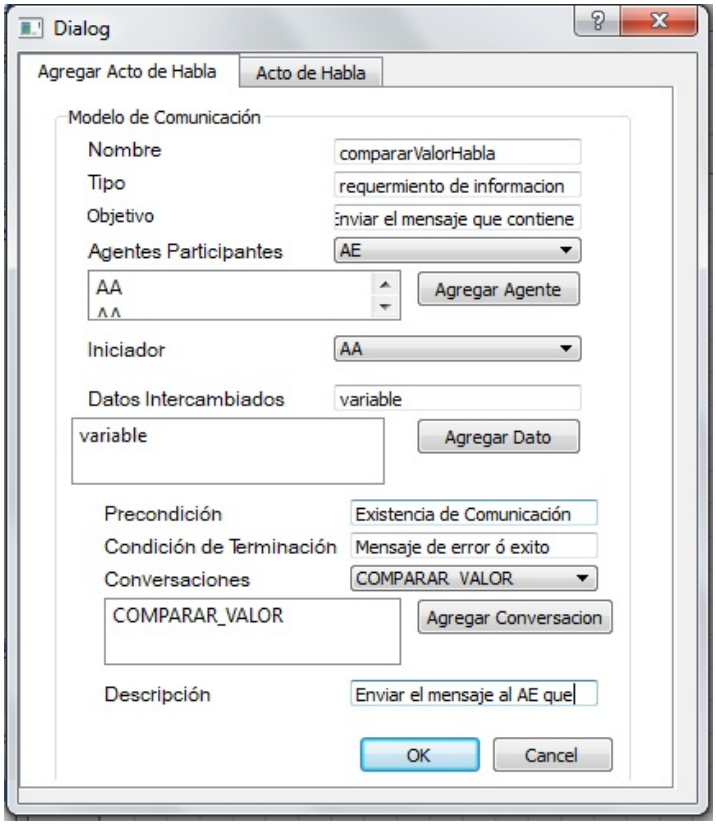

Figura 6.10: Interfaz para crear un Modelo de Comunicación.

realizar, que van desde la especificación de los modelos de MASINA hasta la implantación del sistema.

1.Crear un SMA: al iniciar la creación de un SMA, se presiona el icono correspondiente en el menú secundario. (ver Figura 6.12)

1.1 Crear un Agente: se selecciona la opción "Agregar Modelo de Agente", activando el primer matraz de la barra lateral izquierda (ver Figura 6.13), o accediendo por el menú superior en la opción "Editar". Luego se despliega una ventana, como la que se muestra en la Figura 6.7, en la cual se deben introducir los parámetros especificados en el modelo de agente.

1.2 Crear una tarea: se selecciona la opción "Modelo Tareas" activando el segundo matraz de la barra lateral izquierda (Ver Figura 6.14), o por el menú superior en la opción editar, seleccionando "Agregar Modelo de Tarea". Luego se despliega una ventana, como la que muestra la Figura 6.8, en la cual se introducen los parámetros especificados en el modelo. En la Figura 6.8 se observa que existe un botón con la opción "cargar diagrama", éste permite importar un diagrama de actividades creado en Umbrello que

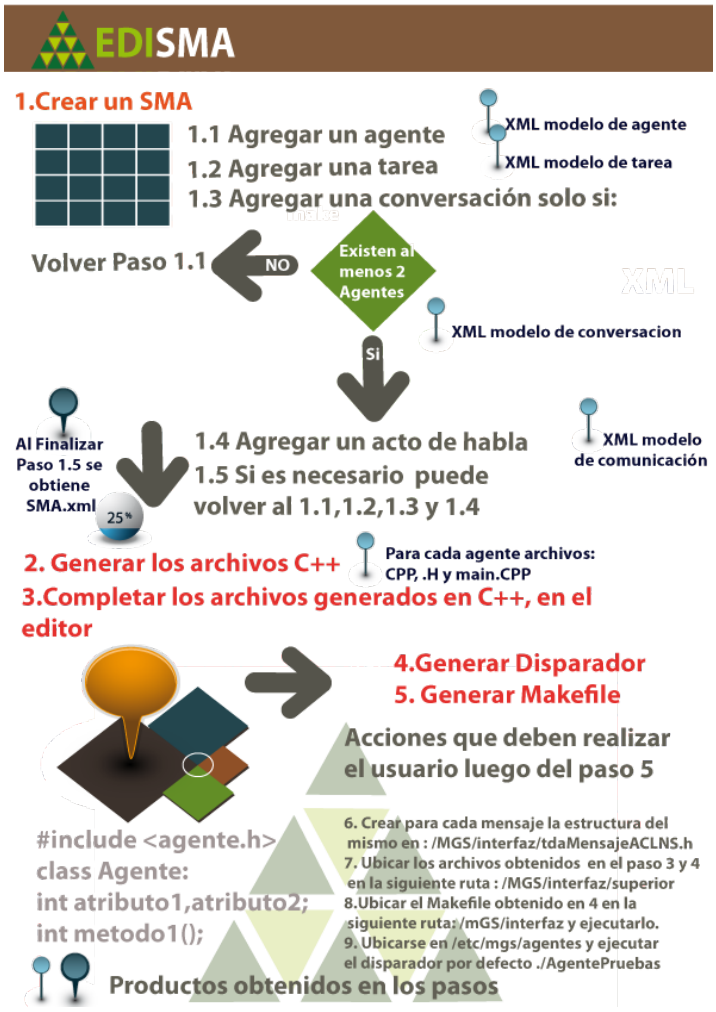

Figura 6.11: Funcionamiento de EDISMA.

se corresponde con la tarea y con el cual se genera un *pseudo* algoritmo, que posteriormente es utilizado por el generador C++.

1.3 Crear una conversación: una vez que se han creado al menos dos agentes es posible generar una conversación. Para ello se selecciona la opción "Modelo de Conversación", representada gráficamente en la barra lateral izquierda por una linea negra con círculos en sus dos extremos (Ver Figura 6.15), o también se puede acceder en el menú superior con la opción "Editar", seleccionando "Agregar Modelo de Conversación". Esta acción despliega una ventana en la cual se introducen los parámetros especificados del modelo (ver Figura 6.9). En la Figura 6.9 se observa que existe un botón con

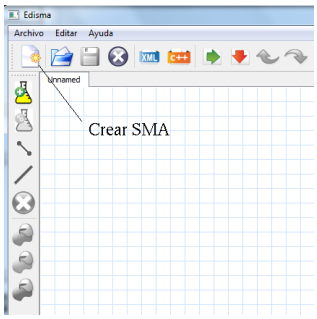

Figura 6.12: Crear un SMA

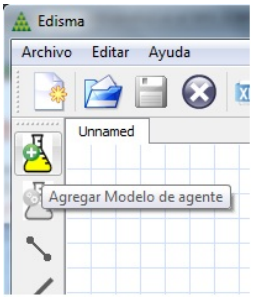

Figura 6.13: Crear un agente de un SMA

la opción "Cargar Diagrama". Esta botón permite importar un diagrama de secuencia creado en Umbrello, que es usado para capturar los atributos de campo Acto de habla, Agentes participantes y Agente iniciador.

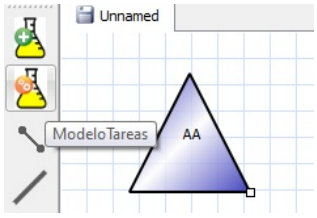

Figura 6.14: Crear una tarea en EDISMA.

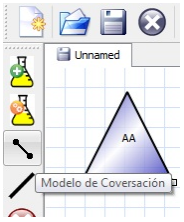

Figura 6.15: Crear una conversación en EDISMA.

1.4 Crear un acto de habla: una vez que se ha creado al menos una conversación, es posible generar un acto de habla. Para ello se selecciona la opción "Modelo de Comunicación, representada gráficamente en la barra lateral izquierda por una línea negra (Ver Figura 6.16), también se puede acceder por el menú superior en la opción "Editar", seleccionando "Agregar Modelo de Comunicación". Esta acción despliega una ventana que permite introducir los parámetros especificados en el Modelo de Comunicación (ver Figura 6.10).

2. Generar los Archivos C++: cuando se han creado todos los modelos del SMA se procede a generar los archivos C++ que lo implantan. Para ello se selecciona la opción C++, representada gráficamente por un rectángulo verde ubicado en la barra superior de iconos (ver Figura 6.17). Esto despliega, en la sección del árbol de archivos de EDISMA, los archivos respectivos (un ejemplo se muestra la Figura 6.18), compuestos por el archivo .cpp, el .h y el principal para cada agente. Por ejemplo, si el nombre del agente es "*AgentePrueba*", se generaran los archivos "AgentePrueba.cpp" (los métodos

del Agente), “AgentePrueba.h” (el archivo de definiciones para AgentePrueba) y mainAgentePrueba.cpp (el archivo principal de AgentePrueba).

Figura 6.16: Crear un acto de habla en EDISMA.

Figura 6.17: Crear archivos C++.

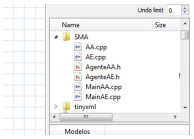

Figura 6.18: Archivos Generados.

3. Completar los archivos generados en C++ en el editor: una vez creados los archivos C++ con los códigos fuentes del SMA, se procede a completar dichos archivos para que los mismos puedan ser compilados y posteriormente ejecutados. Los aspectos que deben completarse en el código se refieren a las acciones específicas en las subtareas, que no pueden describirse en un diagrama de actividades UML, y al cuerpo de los mensajes que tampoco puede detallarse en un diagrama de secuencias. Para ello se selecciona, en el árbol de archivos que se muestra en la Figura 6.18, el archivo que se desea modificar, y con el evento doble *clic* se despliega dicho archivo en el editor, la Figura 6.19 muestra el Editor con un archivo abierto.

Cuando se ha completado el código, como se ilustra en la Figura 6.20, se procede a guardar los cambios del mismo usando una opción similar a la que ofrecen los editores comunes.

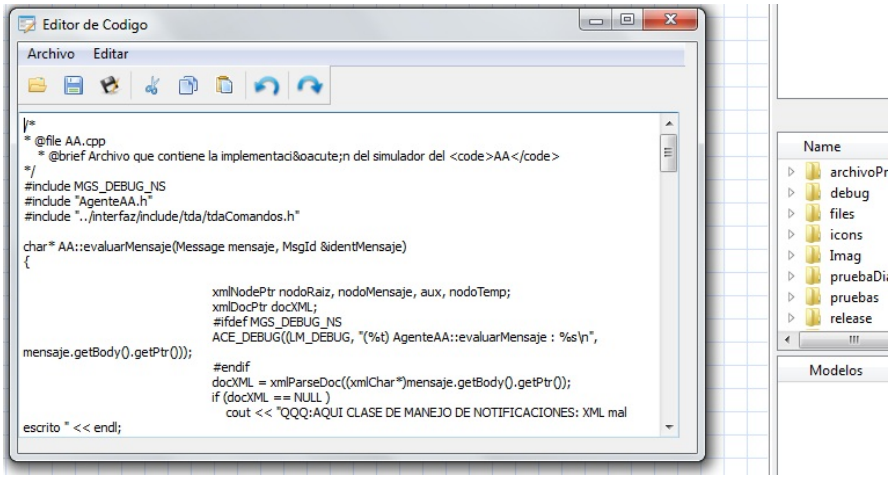

Figura 6.19: Editor de archivos C++ en EDISMA.

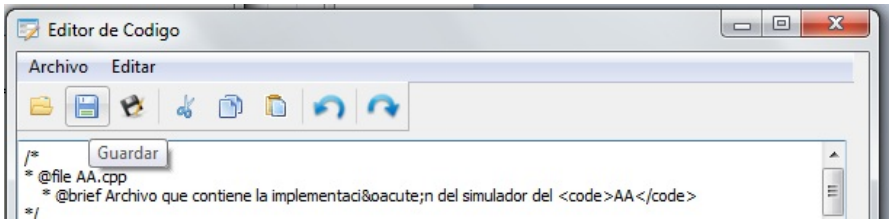

Figura 6.20: Funcionalidad guardar del editor de archivos C++.

4. Generar Disparador: teniendo los códigos creados y modificados según lo requerido, se procede a generar el disparador que permite crear las instancias de los agentes. Para ello se selecciona en la opción Editar, Crear Disparador (ver Figura 6.21).

5. Generar MakeFile: una vez que se ha creado el disparador, se procede a generar el *makefile* que permite, mediante un *script* (guión), compilar los agentes del SMA y el Disparador. Para ello se selecciona en la opción Editar, Crear MakeFile (ver Figura 6.21).

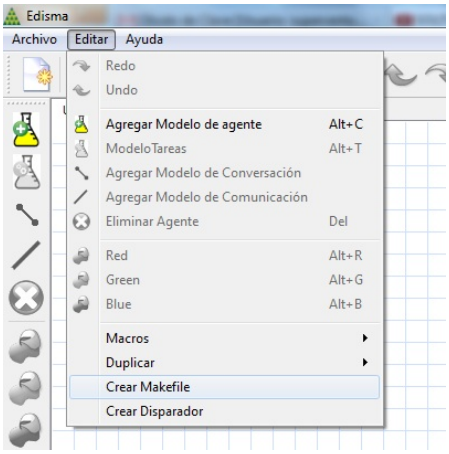

Figura 6.21: Funcionalidad crear Disparador y MakeFile.

6. Reubicar los archivos obtenidos en el paso 3 y 4: con todos los archivos fuentes, se procede a ubicar los archivos .h y .cpp del SMA en la ruta MGS/interfaz/superior del nodo de compilación, según lo especifica el uso del MGS 5.7. Para la versión actual, esta acción queda fuera del alcance de EDISMA, se debe usar usando los mecanismos/comandos básicos de transferencia de archivos.

7. Reubicar el MakeFile obtenido en el paso 5: una vez que se han reubicado los archivos fuentes, se copia el MakeFile en la siguiente ruta

MGS/interfaz/ del nodo de compilación. Luego se compila el SMA, usando la herramienta de compilación del nodo y los archivos copiados.

8. Ejecutar el Disparador: el usuario debe garantizar que el MGS esté en ejecución en cada nodo, y que la configuración es correcta, para esto debe seguir los pasos descritos en la Sección 5.7. Luego es necesario posicionarse en la ruta /etc/mgs/agente y ejecutar el programa disparador que hará que el SMA inicie su ejecución.

6.7. Caso de estudio: Entorno de automatización básico

El objetivo del caso de estudio es comprobar que los archivos generados con EDISMA facilitan la implantación del SMA, dejando al programador la actividad de completar el código que contiene aspectos específicos del modelo de tarea, y la estructura de los mensajes (actos de habla). Para ello es necesario previamente tener especificado formalmente el sistema de acuerdo a MASINA en EDISMA.

6.7.1. Descripción general

Tomando en cuenta que el MGS posee una prueba de instanciación de un SMA en [2], se ha realizado el proceso de ingeniería inversa sobre dicha prueba, que consiste en analizar los archivos que la conforman, para luego determinar y especificar el SMA usando la metodología MASINA, descrita en el Capítulo 3, y proceder a generarlo con EDISMA.

El sistema en estudio pretende simular un escenario simplificado de operación en un entorno SCADA, donde existe un proceso que genera datos, los cuales se están supervisando constantemente. Estos datos son procesados mediante un modelo analítico simple asociado al proceso, y el valor generado por el modelo se compara con algún patrón de operación pre-establecido. Esta prueba se diseñó para probar la operatividad de la plataforma con el MGS, y ahora se usa para comprobar la utilidad de EDISMA; no pretende ser una prueba de las potencialidades de los SMA, dada la simplicidad de los procesos involucrados.

De la fase de conceptualización, según la Sección 3.4.1, se generan cuatro agentes:

- Agente Aplicación (AA): realiza la suma algebraica de dos variables cuyos valores son tipo entero, y asigna dicho resultado a una variable. Este agente emula la operación de un modelo de comportamiento de un proceso físico.
- Agente Especializado (AE): realiza la comparación del valor de una variable con un valor fijo, para identificar si dicha variable posee un valor

mayor, menor o igual. Este agente sirve como un supervisor del estado de operación.
- Agente Negocio (AN): solicita la búsqueda del Agente Repositorio y del AA, así como los servicios de los mismos. Representa al proceso físico, es una instancia lógica del proceso.
- Agente Repositorio (AR): se encarga de almacenar el valor de tres variables, este agente presta su servicio al AA y al AE. Mediante este agente se simula un colector de datos, análogo al que existe en los sistemas SCADA.

Para cada uno de los agentes, y siguiendo los pasos especificados en la metodología, se generaron los modelos respectivos. Desde el punto de vista de operación, la aplicación que representa al SMA es distribuida, por lo cual los procesos que implantan a los agentes se ejecutan en diferentes nodos.

6.7.2. Construcción del caso de estudio con EDISMA

A continuación se presenta el esquema de construcción del SMA para el caso de estudio con EDISMA, siguiendo los pasos descritos en la Sección 6.6. Para ilustrar el proceso, se muestran algunas de la pantallas de interacción con EDSIMA.

1.Crear el SMA: se selecciona la opción de crear SMA y se especifica el nombre y la ruta en la cual se creará el proyecto(ver Figura 6.22).

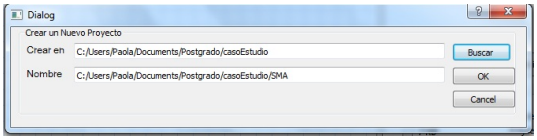

Figura 6.22: Crear el caso de estudio con EDISMA.

1.1 Crear los agentes: se selecciona la opción "Agregar Modelo de Agente". Luego, en la ventana desplegada, se inicia especificando los parámetros para el AA, ver Figuras 6.23, 6.24, 6.25 y 6.26. Luego se crean AE, AR y AN, siguiendo el mismo procedimiento (especificando sus modelos de agente según la metodología).

1.2 Crear las tareas: se selecciona la opción "Modelo de Tarea", luego se introducen los parámetros especificados en el "Modelo de Tarea" del AA. Luego se importa el diagrama de actividades (ver Figura 6.27) hecho con Umbrello, que permite generar un *pseudo* algoritmo en el código fuente que implanta dicha tarea. Este paso se repite para el resto de los agentes del SMA, para cada una de sus tareas.

1.3 Crear las conversaciones: para ello se selecciona la opción "Modelo de Conversación"; en la ventana desplegada se introducen los parámetros

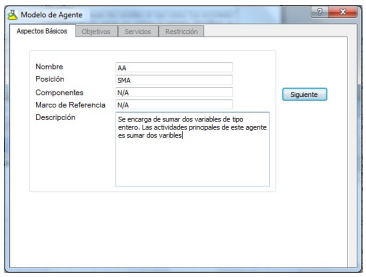

Figura 6.23: Crear AA, pestaña de aspectos básicos.

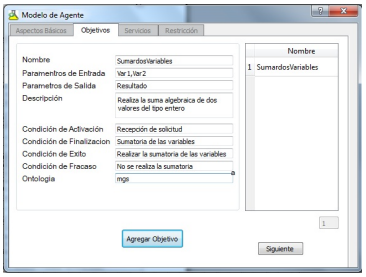

Figura 6.24: Crear AA, pestaña de objetivos.

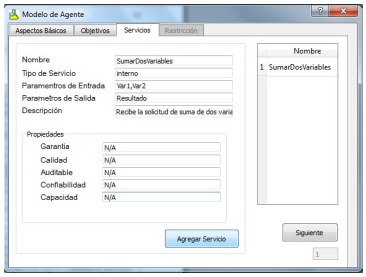

Figura 6.25: Crear AA, pestaña de servicios.

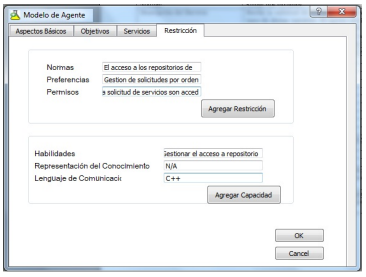

Figura 6.26: Crear AA, pestaña de restricciones.

de este modelo. La Figura 6.30 muestra la ventana para el caso de la conversación LOCALIZAR_APLICACION. Además, en la Figura 6.28 se observa el diagrama de interacción para esta conversación, elaborado en Umbrello, que fue importado a EDISMA. Todas las conversaciones se especifican de la misma forma.

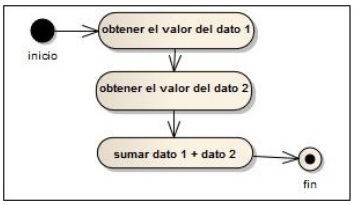

Figura 6.27: Diagrama de actividades para la tarea SumarDosVariables.

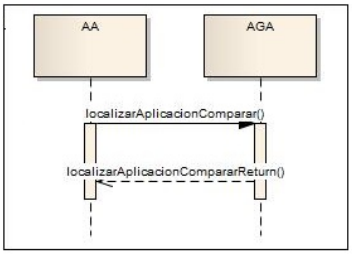

Figura 6.28: Diagrama de interacción para la conversacion LOCALIZAR_APLICACION..

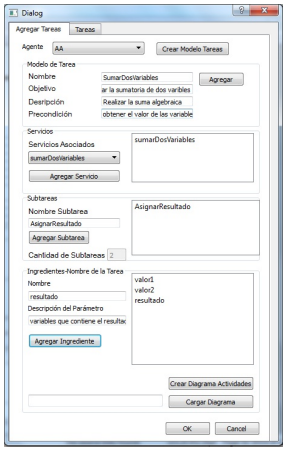

Figura 6.29: Modelo de tarea en EDISMA para el AA.

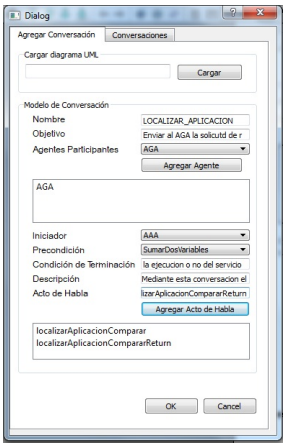

Figura 6.30: Modelo de conversación en EDISMA para el SMA.

1.4 Crear actos de habla: los actos de habla de las conversaciones se crean automáticamente cuando se importa el diagrama de interacción respectivo.

2. Generar los Archivos C++: se selecciona la opción C++. Como muestra la Figura 6.31, cuando los archivos han sido generados, éstos se despliega en la sección del árbol de archivos.

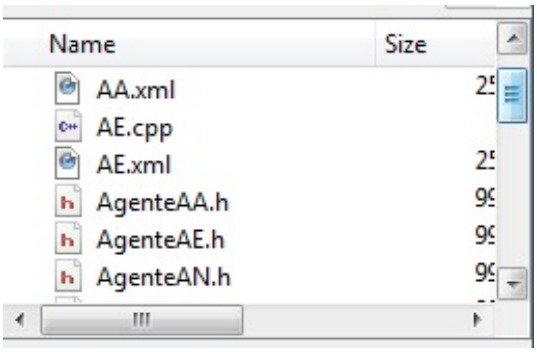

Figura 6.31: Archivos generados para el SMA.

3. Modificar los archivos generados: una vez que se ha creado el código fuente del SMA, se procede a modificarlo para agregar elementos específicos de las tareas y de los mensajes. A modo de ejemplo, en la Figura 6.32 se muestra en el editor el contenido del archivo AA.cpp, donde es posible observar, en color amarillo, el lugar donde debe completarse el código, para poder implementar el AA. La etiqueta \\<EDITAR> la crea automáticamente el generador C++, permite que Editor la resalte en color amarillo para in-

dicar al usuario donde debe modificar el código generado. Una vez que los cambios han sido introducidos, se procede a guardar el archivo (ver Figura 6.33, con los cambios introducidos para AA).

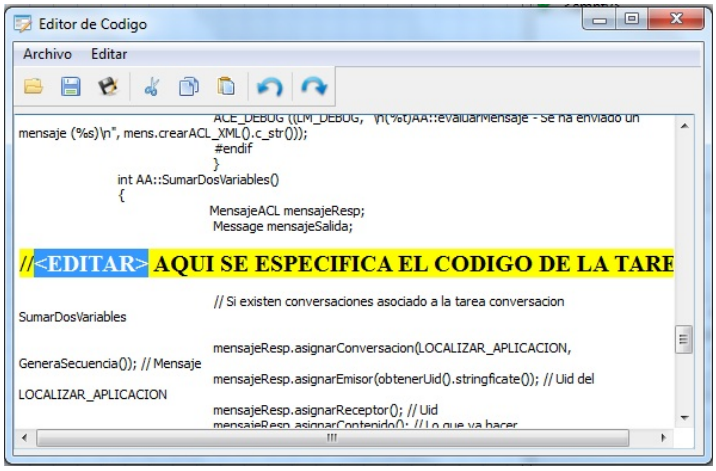

Figura 6.32: Editor de archivos C++ en EDISMA, AA.cpp sin edición

.

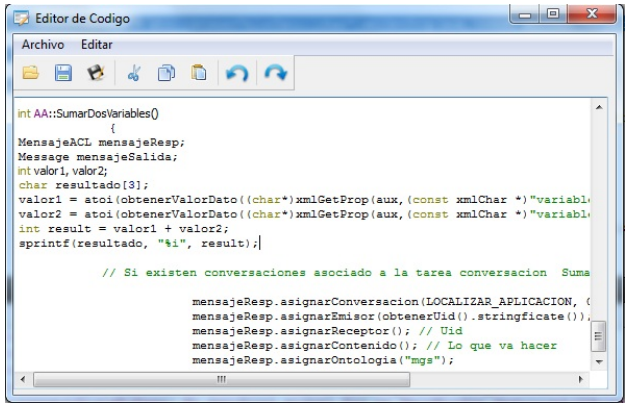

Figura 6.33: Editor de archivos C++ en EDISMA, AA.cpp con edición.

4. Generar Disparador: se selecciona la opción Editar, Crear Disparador.

5. Generar MakeFile: se selecciona la opción Editar, Crear MakeFile.

6. Reubicar los archivos obtenidos en el paso 3 y 4: una vez que se ha creado el MakeFile, se procede a ubicar los archivos .h y .cpp del SMA en la ruta MGS/interfaz/superior.

7. Reubicar el MakeFile obtenido en el paso 5: una vez que se han reubicado los archivos C++ generados en EDISMA, se procede a reubicar el MakeFile en la siguiente ruta MGS/interfaz/. Luego se compila el SMA, usando el makeFile.

8. Ejecutar el Disparador: una vez que se ha compilado el makeFile, el usuario debe garantizar que el MGS esté en ejecución en cada nodo, y que la configuración es correcta, para esto debe seguir los pasos especificados en la Sección 5.7.

EDISMA generó la estructura de los agentes del SMA y los aspectos de comunicación y coordinación. Solo fue necesario completar los detalles de las tareas de los agentes, adicionando para ello el código necesario en cada archivo cuerpo. También, fue necesario especificar la estructura de los mensajes utilizados en el modelo de comunicación.

Referencias

1. J. Aguilar, M Cerrada, A. Ríos, and F. Hidrobo. Diseño y Construcción de una Plataforma para el desarrollo e implantación de sistemas multiagentes. Technical report, ULA, CDCHT, 07 2010.
2. J. Aguilar, L. Leon, A. Ríos, F. Hidrobo, J. BarRíos, and O. Teran. Reporte de validacion de pruebas de integracion del invel interfaz y base. In *Seguimiento, control y evaluacion de la fase de desarrollo del Medio de Gestión de Servicios de la plataforma Net-DAS 2.0*, pages 1–9, Merida, Diciembre 2007.
3. J. Aguilar and W. Zayas. Sistema multi-agente para crear agentes de control para el scdia. *Avances en Sistemas e Informática*, 6(2):47–58, 2009.
4. J. Barrios, J. Montilva. Método white_watch para el desarrollo de proyectos pequeños de software. Versión 1.2, Universidad de Los Andes, Venezuela, Enero 2010.
5. R. Bravo. Sistema generador de codigo para la metodología masina. Tesis de pregrado, Universidad de Los Andes, Mérida, Venezuela, 2009.
6. Debra Cameron, Bill Rosenblatt, and Eric Raymond. *Learning GNU Emacs (2nd ed.)*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1996.
7. Inc Free Software Foundation. Tgnu emacs manual. http://www.gnu.org/software/emacs/manual/text/emacs.txt, 1993-2012.
8. P .Hensgen. Manual de umbrello uml modeller. Umbrello UML Modeller Autores, 2003.
9. P .Hensgen. Umbrello 2.0. http://uml.sourceforge.net/download.php, 2013.
10. Microsoft. Block de notas. http://support.microsoft.com/kb/260563/es, 2013.
11. D. Molkentin. *The Book of Qt 4: The Art of Building Qt Applications*. No Starch Press Series. Open Source Press, 2007.
12. J. Montilva. Desarrollo de aplicaciones empresariales. Informe técnico, Grupo de Investigación en Ingeniería de Datos y Conocimiento. Universidad de Los Andes, Mérida, Venezuela, 2004.
13. W3 org. Extensible markup language. http://www.w3.org/XML/, 2013.
14. R. Pressman. *Ingenieria del Software*. Mc Graw Hill, 2006.
15. E.T. Ray. *Learning XML*. Safari electronic books. O'Reilly Media, 2009.
16. S. Schach. *Análisis y diseño orientado a objetos con UML y el proceso unificado*. Mc Graw Hill, 01 edition, 2005.
17. Trolltech. Qt4. http://qt-project.org/downloads, 2013.

Parte III
Automatización Integrada de Procesos

Preámbulo

Desde su concepción, el control de procesos técnicos y su evolución a la automatización integrada de medios y mecanismos de producción ha estado dirigido por criterios de seguridad y productividad. Si bien, en sus etapas incipientes, la aplicación de métodos y técnicas para asegurar la operatividad de los sistemas productivos con la mínima intervención de factor humano se centralizada directamente en los procesos controlados, el desarrollo tecnológico ha permitido crecer en la automatización industrial, hasta alcanzar lo que hoy se conoce como los sistemas de control distribuido.

Desde una perspectiva histórica, entonces la automatización se inicia con la incorporación de tecnologías para garantizar una operación segura de máquinas y herramientas, tal como fue el caso del uso de relés. Ese nivel de automatización fue, además, soportado con técnicas de control de procesos de poca complejidad, por ejemplo, los tipos PID. Con el desarrollo del transistor, esta filosofía de automatización se mantiene ganándose en continuidad operacional, reducción de costos por mantenimiento, como aspectos mas resaltantes. En esta primera era de la automatización, los criterios de productividad tenían una importancia relativa. La seguridad operacional marcaba los criterios para la automatización de los procesos.

Un primer gran salto en la automatización surge con el devenir de la era del microprocesador, a partir de lo cual se establecen nuevos paradigmas y criterios para el control y supervisión de los procesos. Con los computadores personales aparecen los controladores lógicos programables (PLC), y las filosofías de automatización centradas en la organización empresarial, por ejemplo los modelos jerárquicos. Es así como, entonces, las tendencias en el desarrollo informático moldean las tendencias en automatización industrial. A partir de los PLCs, se pasa a los sistemas de adquisición de datos y control supervisorio (SCADA) y los sistemas de control distribuido (DCS), donde los paradigmas de control multivariable robusto, control óptimo, entre otros, se consideran objetos de implementación. Al mismo tiempo, los criterios para la automatización basados en la productividad son acrecentados, obviamente, sin desviar los criterios primarios basados en la seguridad y confiabilidad operacional. Además, el desarrollo de las tecnologías de información y comunicación (TIC) configuran los paradigmas de la automatización de procesos.

Siguiendo la evolución y tendencias en los sistemas informáticos, con la Internet emerge otro gran salto para la automatización de los procesos, la cual permite implantar arquitecturas y modelos de automatización que, en un principio, se consideraban esquemas de trabajo, como ha sido el caso del modelo ISO/OSI. Entonces, las TICs, conjuntamente con la mecatrónica, comienzan a imponer los requerimientos para que los métodos, criterios, esquemas y estructuración de los procesos productivos sean automatizados.

Con esa visión, la mayor dificultad en la implantación de una automatización plena ha sido la integración de sistemas, procesos y datos.

Con el desarrollo de las TICs y la Internet surge la sociedad de la información, donde el conocimiento producido mediante la adquisición, procesamiento y tratamiento de datos, conjuntamente con toda la estructura de hardware, imponen nuevos métodos, estrategias y paradigmas de organización y producción en los procesos técnicos. De tal manera que los criterios para la automatización se acompañan con la visión de la economía global, las tendencias económicas, el desarrollo sustentable, la organización del negocio, la fabricación y producción reconfigurable, la seguridad ambiental, la eficiencia y disponibilidad de las plantas, la producción masiva de bienes, entre varios aspectos novedosos. Por lo tanto, a la par de los requerimientos de seguridad, confiabilidad y productividad; el factor humano, las determinaciones culturales y las tradiciones empiezan a ser decisivas en la manera en que las tecnologías para los sistemas automatizados son diseñados y aplicados. De manera que el éxito de la implantación de tecnologías de puntas y la investigación en automatización, dependen fuertemente del ingrediente y la identidad cultural, en un mundo tan competitivo como el actual.

Para la producción industrial, la sociedad de la información establece importantes retos haciendo que los procesos sean más complejos, por lo que demandan mejores y mayores sistemas para la automatización, debido a criterios en función de la orientación humano-sociológica, la flexibilidad y estructura organizacional, las capacidades de adaptación y los rápidos cambios en la economía global. De este modo, la importancia de la automatización en los procesos industriales se incrementa cada día mas, tomando mayor protagonismo.

La convergencia de nuevas tecnologías en el ámbito de los sistemas informáticos, como la inteligencia artificial distribuida, y en los métodos y medios de producción, como el paradigma de producción inteligente, imponen nuevos retos que permiten reducir la brecha entre información, comunicación y automatización, de modo que la automatización integrada inteligente de los procesos pueda hacerse realidad. Esos retos, a partir de las tecnologías claves, han definido líneas de trabajo en investigación y desarrollo que en un futuro próximo puedan generar productos tecnológicos para la automatización inteligente, que permita alcanzar mayores exigencias en los procesos productivos. En ese contexto, los sistemas distribuidos inteligentes se constituyen en un paradigma relevante para la construcción de mecanismos de automatización integrada. De allí surgen los *Agentes Inteligentes* como un paradigma para implementar métodos integradores, de control y supervisión y de información y conocimiento, en procesos complejos como los de producción industrial. De igual manera se establecen nuevas arquitecturas y modelos para la *Automatización Integrada Inteligente* (AII), que efectivamente garanticen las tareas y procesos de la automatización actual: a) soporte en la gerencia e ingeniería durante la operación; b) rápido desarrollo, modificación y adaptación; c) soporte en la planificación, imple-

mentación y supervisión; d) integración e interoperabilidad de diferentes sistemas heterogéneos.

Con esa visión, la automatización industrial estará regida por un *Sistema Integrado* basado en agentes inteligentes, construido a partir de las TICs y de las tecnologías de control inteligente de procesos, empleando todas las fortalezas de las técnicas de inteligencia artificial distribuida y de optimización. En este sistema integrado se dispondrán de redes de campo, redes de unidades de producción, redes corporativas, etc., las cuales dan soporte a todas las operaciones del complejo productivo donde la inteligencia estará disponible, funcionalmente en todos los procesos, según el modelo de negocio. Esto conlleva a un *aplanamiento* de las arquitecturas jerárquicas hasta alcanzar una automatización heterárquica holónica, donde se disponen de sistemas que operan coordinamente en forma cooperativa e integral, para alcanzar las metas de producción.

En este contexto, en esta parte se aborda el tema de automatización integrada de procesos productivos, comenzando con el Capítulo 7 donde, a partir de una perspectiva histórica, se presentan las bases teóricas de la automatización industrial de procesos. Seguidamente, en el Capítulo 8 se describen las diferentes arquitecturas y modelos para la automatización de procesos técnicos, estableciendo ciertas comparaciones de los mismos. Finalmente, en el Capítulo 9 se analiza el problema de integración, evaluándose algunas vías de solución.

Capítulo 7
Automatización de Procesos

Resumen: En este capítulo se presenta todo lo referente a Automatización de Procesos Técnicos.

7.1. Introducción

La sustitución, en los procesos técnicos, del quehacer humano por equipos mecánicos, eléctricos, electrónicos o sistemas, con el fin de optimizar el uso de los recursos existentes y la continuidad de los procesos, recibe el nombre de *automatización*.

En la actualidad, los desarrollos tecnológicos y la existencia de un mundo de información al alcance de todos, ha llevado a la humanidad a idear metodologías y prácticas para la estructuración de las organizaciones empresariales, de la comunicación y manejo de datos desarrollados en diferentes plataformas, así como técnicas y arquitecturas de automatización.

Inicialmente, para el modelado de procesos industriales se realizan representaciones conceptuales de la automatización y se definen los requerimientos comunes para todas las implementaciones, independientemente de los requisitos específicos del proceso particular. La automatización debe proveer una infraestructura que permita cubrir todas las fases y aspectos del proceso productivo con el fin de lograr un incremento de la producción, manteniendo o reduciendo los costos, por esta razón en estos modelos se encuentran presentes las actividades inherentes de la automatización:

- Optimización
- Planificación
- Supervisión
- Control

En función del proceso industrial, se organizan e integran estas actividades definiendo diferentes arquitecturas jerárquicas, heterárquicas e inclusive híbridas, donde se toman las características que se consideren convenientes de algunos modelos referenciales de automatización.

La *Automatización Industrial* (AI) se caracteriza por una relación intrínseca de varias ingenierías: eléctrica, electrónica, mecánica, química, de comunicaciones, computacional y de software. Constituye un conjunto de técnicas que involucran la aplicación e integración de sistemas industriales de forma autónoma, siendo un área en la que confluyen esas diferentes disciplinas para la solución de los problemas propios de los procesos productivos tales como eficiencia, productividad, calidad, decisiones estratégicas y diseño de procesos a nivel de producción, planta y a nivel gerencial.

Por lo tanto, la AI aborda los problemas técnicos y operacionales, en todos los niveles, de los sistemas productivos. Así, la AI proporciona mejoras en la producción por adición de valor, y permite mejorar la productividad y calidad, reducir tiempos de paradas e incrementar la eficiencia y la seguridad.

7.2. Automatización Industrial

El proceso de automatización consiste en diseñar sistemas capaces de ejecutar tareas repetitivas y de controlar operaciones con la mínima intervención de un operador humano, permitiendo, a su vez, la disposición de elementos para supervisar y controlar cada uno de los procesos dentro de las cadenas de producción, almacenamiento y distribución, así como los diferentes servicios ofrecidos a los clientes por parte de las fábricas, empresas y proveedores; implica la implantación de instrumentación, dispositivos de maniobra, máquinas mecánicas o hidráulicas móviles, autómatas, displays o paneles, SCADAs y redes de comunicación que enlacen cada uno de los elementos que componen el sistema de automatización.

Definición 7.1. La AI es un conjunto de técnicas que involucran la aplicación e integración de sistemas mecánicos, eléctricos-electrónicos, fluidos, soportados por tecnologías de información y comunicación, todos ellos unidos para operar, supervisar y controlar diferentes tipos de procesos técnicos de forma autónoma y con la mínima intervención humana.

Mediante la implementación de sistemas automatizados se logra:

- Mejorar la productividad de la empresa, reduciendo los costes de la producción y mejorando la calidad de la misma.

- Mejorar las condiciones de trabajo del personal, suprimiendo los trabajos penosos e incrementando la seguridad
- Realizar las operaciones imposibles de controlar intelectual o manualmente.
- Mejorar la disponibilidad de los productos, pudiendo proveer las cantidades necesarias en el momento preciso.
- Simplificar el mantenimiento de forma que el operario no requiera grandes conocimientos para la manipulación del proceso productivo.
- Integrar los sistemas de gestión y los sistemas de apoyo a la producción.

Independientemente de la arquitectura utilizada, la automatización debe permitir implantar un sistema capaz de realizar la captación, adquisición, control y supervisión de todos los elementos, procesos y servicios llevados a cabo por las industrias, fábricas y empresas, posibilitando al personal encargado la monitorización de los procesos. De esta forma, la automatización debe proveer una infraestructura que permita cubrir todas las fases y aspectos del proceso productivo con el fin de lograr un incremento de la producción, manteniendo o reduciendo los costos.

El desarrollo de la AI ha estado ligado al desarrollo de los sistemas computacionales. Por lo tanto, el desarrollo tecnológico de los últimos años ha ocasionado un incremento en el número de actividades desempeñadas por sistemas automatizados. Inicialmente, la automatización sólo ocurría a nivel de máquinas y supervisión, siendo incorporada posteriormente al proceso de planificación y optimización, para la creación de nuevos modelos de automatización, de acuerdo a las estructuras del sistema de producción.

7.2.1. La Automatización basada en computadores

Con el desarrollo de las minicomputadoras a mediados de las 60's y el microcomputador en los 70's, las tareas relativas al control de procesos industriales cambiaron y siguen cambiando vertiginosamente a favor de implementar técnicas muy depuradas y con diversos índices de optimización, en función de la potencialidad de cálculos de los dispositivos microprocesadores y microcontroladores.

Definición 7.2. El *control de procesos* se refiere a los sistemas de control de regulación automática, en los cuales las variables que se manejan son variables físicas o químicas como temperatura, presión, flujo, nivel, PH, alguna composición, etc.

El computador, en un ambiente de control de procesos, cumple las siguientes funciones:

1. Adquisición de datos.
2. Control digital directo.
3. Control supervisorio.
4. Control jerárquico.

7.2.1.1. Adquisición de datos

La adquisición de datos es un proceso donde un conjunto de datos del mundo físico son recolectados, procesados, almacenados y usados. Los Sistemas de Adquisición de Datos (SAD) son ampliamente usados en la industria. Ellos se utilizan en investigación, desarrollo, producción, control de procesos, control de calidad, procesos de prueba, gerencia, etc.

Cuando el computador se utiliza para una adquisición de datos pura, él no participa en forma directa en la función de control, es decir, el computador opera en línea y en tiempo real con el proceso, pero no cierra ningún lazo de control. De manera que a través de sistemas de entrada de señales y de salida de señales, el computador mantiene actualizada la información de lo que está ocurriendo en el proceso y la presenta cuando y en la forma que se requiere. Las señales del proceso pueden ser analógicas o digitales, las cuales se adecúan por medio de acondicionamiento de señales, a los requerimientos del computador.

Sistemas de Adquisición de Datos

En virtud de que en su mayoría los procesos industriales son continuos, para que el computador digital pueda manejar adecuadamente las variables del proceso y los elementos finales de control, se debe realizar un proceso de adquisición, acondicionamiento y conversión de señales continuas. Un sistema que permite realizar estas tareas, se denomina SAD.

Un SAD típico se muestra en la Figura 7.1. En el se distinguen los siguientes elementos:

- *Transductor*: son elementos que permiten obtener una señal eléctrica equivalente a alguna variable física.
- *Multiplexor*: un dispositivo utilizado para la recolección de datos a través de diferentes canales hacia un único Convertidor Analógico-Digital (ADC).
- *Acondicionador de señales*: son elementos de filtrado y adecuación de los niveles de voltaje o de corriente de la señal eléctrica de acuerdo a las características del ADC.
- *Convertidor Analógico-Digital*: es un dispositivo que convierte señales analógicas en digitales. Las características principales de éste elemento son la longitud de la palabra digital (resolución en *bits* y la frecuencia de muestreo.

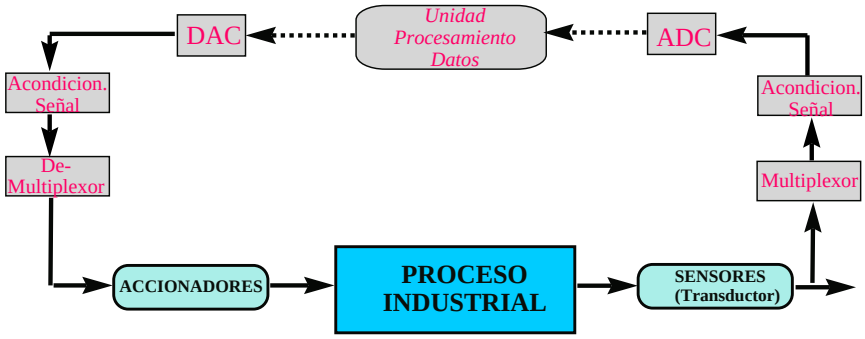

Figura 7.1: SAD típico.

- *Unidad de procesamiento de datos*: el dato digital proveniente del ADC irá a una unidad de procesamiento de datos. Dependiendo del tipo de SAD empleado, esta unidad puede ser un microprocesador o un computador digital separado, por ejemplo una Unidad de Transmisión Remota (RTU), un Maestro de Transmisión Remota (MTU), etc. Los datos procesados se pueden enrutar a cualquiera de los siguientes destinos:

  1. Memoria.
  2. Unidad de presentación visual.
  3. Sistema de alarma.

Para garantizar que el equivalente digital de las señales continuas contenga la plenitud de la información, los dispositivos ADC y DAC (convertidor digital-analógico), deben operar con una rata de muestreo de al menos el doble del ancho de banda de la señal continua. La frecuencia mínima de muestreo se llama *Frecuencia de Nyquist*, [6, 7, 5]. Otro de los factores a considerar durante el proceso de conversión es la longitud de la palabra digital equivalente, ya que de ello dependerá la precisión de los datos en base a lo que se conoce como *cuantización*. Es importante destacar que a medida que se utilize convertidores con mayor número de bits, mayor será su precisión al igual que su costo y tiempo de procesamiento. Además, es de vital importancia para el buen análisis y síntesis de estructuras de control para los sistemas supervisados, considerar cual es la influencia de los convertidores en cuanto al tamaño de la palabra digital y por supuesto la frecuencia de muestreo.

En la Figura 7.2 se muestra la configuración de un sistema donde el computador realiza las tareas de monitoreo de señales del proceso.

Las salidas de un SAD, a través de un computador, pueden ser:

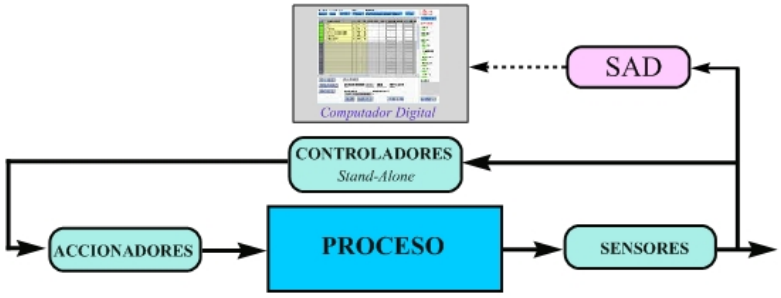

Figura 7.2: El computador en adquisición y manejo de datos.

- *Estado actual del proceso*: el computador muestra a través de terminales en pantallas o mímicos del proceso, los valores actuales de las diferentes variables de proceso. La información se actualiza periódicamente.
- *Alarmas*: el sistema debe estar diseñado para indicar rápidamente cualquier condición anormal en la operación del proceso. Existen dos formas alternativas para la programación de la detección y el manejo de las condiciones de alarma. En la primera, la detección se incluye dentro de la secuencia programada para la adquisición de datos; la segunda, se basa en la instalación de tarjetas sensoras de eventos diseñadas para detectar cambios de estado.
- *Registro y gráficos de tendencias*: en la medida que se van produciendo barridos sucesivos, el computador puede ser programado para que la información obtenida se transfiera a archivos; obteniéndose de esta forma, las curvas de tendencia correspondiente a las diferentes variables del proceso. Estas pueden ser visualizadas por pantallas o a través de la impresión.
- *Registro histórico*: los archivos anteriores se pueden compilar estadísticamente (reconciliación de los datos) para disponer de parámetros que indiquen el comportamiento del proceso en forma de registros temporales. Además, el computador se puede programar para que funcione como un registrador de los eventos ocurridos en condiciones de emergencia. Estas son herramientas poderosas para la auditoría técnica del proceso.
- *Reporte diario de la operación*: el sistema puede generar los reportes diarios o por turnos de la operación, conforme a las especificaciones de los usuarios.
- *Conversión a unidades físicas*: el computador lo que lee son valores digitalizados correspondientes a las señales del proceso, proveniente de los transmisores. Estos valores se pueden convertir a sus unidades físicas originales a través del computador. Además, se pueden realizar funciones de compensación de no linealidades.
- *Estimación de variables*: mediante la aplicación de algoritmos de estimación de parámetros u observadores de estado; dado un conjunto de va-

riables medidas en un proceso, se pueden estimar otras variables cuyas mediciones no se realizan por ciertas dificultades tecnológicas y/o físicas. En la actualidad, ésto se refiere a la instrumentación virtual.

- *Detección de fallas en la instrumentación*: en el caso de variables cuya medición sea crítica para la supervisión del proceso, puede resultar interesante dotar al computador de la capacidad de detectar fallas en la instrumentación asociada a esas mediciones. Esto se puede lograr por estimación y comparación.
- *Procesamiento digital de señales*: es muy usual que varias de las señales provenientes del proceso, estén corruptas por ruidos; para resolver este problema además del uso de filtro analógicos, se pueden emplear algoritmos para el filtrado digital, de esta forma es el computador quien se encarga de filtrar el ruido.

7.2.1.2. Control Digital Directo (CDD)

Tal como se muestra en la Figura 7.3, el computador es el responsable de generar las señales de control sobre el proceso a través de las variables medidas y las especificaciones de operación. Desde el punto de vista del proceso, el control así ejecutado es operacional por cuanto se realizan tareas a tiempo real que involucran directamente las operaciones sobre el proceso técnico. Así, este control también es referido como *control operacional*.

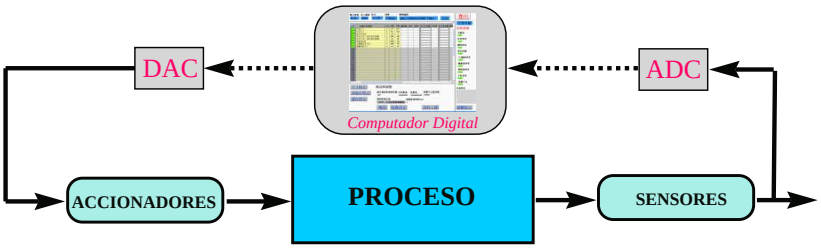

Figura 7.3: Computador digital dentro del lazo de control.

En un esquema de control básico se distinguen las siguientes variables, según la Figura 7.4:

1. *Variable controlada*: es la variable que se desea mantener constante a objeto de garantizar la operación estable del proceso.
2. *Variable manipulada*: es la variable que se maneja en el proceso con el fin de mantener regulada a la variable anterior.
3. *Variable de perturbación o carga*: son variables que influyen sobre la variable controlada, pero que no son manipuladas a los fines de ejercer

el control. El objetivo del sistema de control es el mantener la variable controlada en su valor deseado, independientemente de las variaciones o perturbaciones que se puedan producir en las variables de carga.

4. *Variables medidas*: son las que se miden para poder calcular la señal de control. Su selección depende del esquema de control a utilizar.

7.2.1.3. Diseño de sistemas de control para CDD

En general, los sistemas de control, y en particular el control de procesos, se basan en el principio de realimentación, donde las señales a controlar se comparan con las señales de referencias y la discrepancia (error) se utiliza para determinar la acción de control correctiva. Este modo de operación se muestra esquemáticamente en la Figura 7.4.

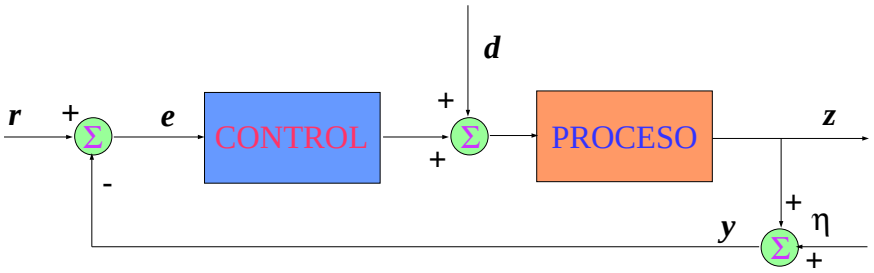

Figura 7.4: Sistema de control realimentado.

El problema se traduce en diseñar el *control* dado un *proceso* y en base a ciertas especificaciones que se indican a través de las señales de referencias, características de las respuesta transitoria, condiciones sobre las variables de control, rechazo a perturbaciones, etc.

La tarea de generar las señales de control a través de un computador se basa en un algoritmo de control, el cual se define como:

Definición 7.3. Un Algoritmo de Control es el conjunto de instrucciones responsables de calcular el valor adecuado para las variables manipuladas, en función de las variables medidas del proceso y los puntos de ajuste (*Set-Points*) para las variables controladas, sobre la base de las especificaciones de desempeño del proceso.

Siguiendo el principio de realimentación, generalmente el proceso de diseño de un sistema de control se lleva a cabo según el siguiente escenario:

1. Estudiar el sistema a controlar y decidir los tipos de sensores, transmisores y actuadores que se emplearán y donde serán ubicados.
2. Modelar y/o identificar el sistema resultante a controlar.
3. De ser necesario, simplificar el modelo a fin de hacerlo útil para su manejo.
4. Analizar el modelo resultante; determinar sus propiedades.
5. Decidir las especificaciones de funcionamiento.
6. Decidir acerca del tipo de controlador a utilizar.
7. Diseñar un controlador que permita cumplir en lo posible las especificaciones; de lo contrario, modificar las especificaciones ó generalizar el tipo de controlador.
8. Simular el sistema controlado resultante, bien en un computador ó en una planta piloto.
9. Si fuese el caso, repetir el procedimiento desde el paso 1, nuevamente.
10. Seleccionar el hardware y el software e implantar el controlador.
11. Si es requerido, sintonizar el controlador en línea.

En este momento, es importante señalar que el rol del ingeniero de control no se debe limitar al diseño del sistema de control sino que por el contrario, debe asistir en la selección y configuración del hardware considerando un punto de vista muy amplio del desempeño del sistema, esto le permitirá indicar directamente y sin ambigüedades donde no se pueden cumplir los objetivos de desempeño.

Desde el punto de vista del control por computadoras, el diseño de sistemas de control involucra algunas tareas adicionales en las etapas antes señales, en virtud de las presencia de elementos que permiten el control de procesos continuos mediante computadores, tal como se muestra en la Figura 7.3.

Una vez definido el modelo y las especificaciones de desempeño, en la etapa de diseño del controlador se pueden seguir dos vertientes:

- Diseño continuo y discretización.
- Discretización y diseño digital.

Por el lado del diseño continuo, se trata de determinar un controlador continuo para el proceso, a través de las técnicas convencionales, considerando los efectos de los elementos de conversión digital analógica, es decir, las funciones de transferencias de los equivalentes de retención. Una vez calculado el controlador, se procede a su discretización por cualquier método: transformación bilineal, en adelanto, en atraso, por retenedor de orden cero, etc. [5].

Por la vía del diseño discreto, en principio se obtiene un equivalente discreto del modelo resultante y posteriormente se aplican las técnicas de síntesis de controladores digitales: controlador PID digital, ubicación de polos,

realimentación de estados, métodos frecuenciales, etc. Por supuesto que, una vez definido el controlador digital se procede a generar el algoritmo de control.

En cuanto al modelo discreto equivalente, éste se puede representar por una ecuación en diferencia, una función de transferencia de pulso ó una descripción en espacio de estados, lo cual establece los requerimientos matemáticos para el análisis y síntesis de los sistemas de control digital.

Desde el punto de vista del control de procesos, en la unidad de procesamiento de datos de un SAD se puede realizar la implementación de los algoritmos de control a objeto de tomar acciones correctivas para satisfacer las especificaciones de desempeño de la planta.

La señal de control que genera el computador debe ser acondicionada de acuerdo al elemento final de control, para que las acciones correctivas sean efectivas. El dispositivo que convierte señales digitales en analógicas se denomina DAC, [3]. Estos dispositivos son los que permiten la interconectividad del computador digital con el mundo analógico.

Esta estructura de manejo de procesos continuos a través del computador digital, constituye lo que se denomina *Sistema de Control de Datos Muestreados (SCDM)* y que se puede visualizar en la Figura 7.3.

Otro objetivo específico de un sistema de control, es la función de servomecanismo; ello corresponde al objetivo de lograr que las variables controladas sigan una trayectoria deseadas o de referencia. En el control de procesos el objetivo principal, como ya se mencionó, es la de regulación, por cuanto se trata de mantener a los procesos en unas condiciones estáticas deseadas.

El uso de los computadores digitales en el lazo de control, proporciona las siguientes ventajas:

- Reducir costos.
- Flexibilizar la respuesta ante cambios en el diseño.
- Inmunidad al ruido.
- Controlar varios lazos simultáneamente.
- Implantar estrategias de control complejas.
- Integrar, de manera plena, los procesos industriales.
- Optimizar.
- Datos históricos para control estadístico de los procesos.

Otra ventaja adicional del CDD, proviene de la naturaleza misma del computador para realizar las funciones de secuenciamiento y toma de decisiones lógicas, como las requeridas en los arranques o parada de los procesos, o en los procesos por lotes (batch). Esto se refiere como el *control secuencial* que es la tarea típica de los Controladores Lógicos Programables (PLC's).

7.2.2. *Control supervisorio*

La incorporación del computador en el control de procesos ha incrementado el rango de actividades que se pueden desarrollar. El computador no tan solo puede controlar, directamente, la operación de las plantas, sino que también proporciona a los gerentes e ingenieros un marco comprensivo del estado de las operaciones de los procesos. Es en este rol *supervisorio*, y en la forma de presentación de la información, que se han hecho los mayores cambios que en la actualidad, comenzando con los Sistemas de Adquisición de Datos y Control Supervisorio (SCADA) [4], los DCS, basados en la filosofía de los modelos de Manufactura Integrada por Computadores (CIM) [8], la cual permite la integración total de los procesos.

En la Figura 7.5 se muestra gráficamente el concepto de control supervisorio. Como se puede observar, en esta configuración la función de control es realizada por los controladores dedicados (ó módulos de control), los cuales pueden ser analógicos o basados en microprocesadores; mientras que el computador es el responsable de suministrar los puntos de ajuste de una manera óptima (para minimizar la energía o maximizar la producción), a partir de especificaciones de funcionamiento y/o especificaciones económicamente justificadas.

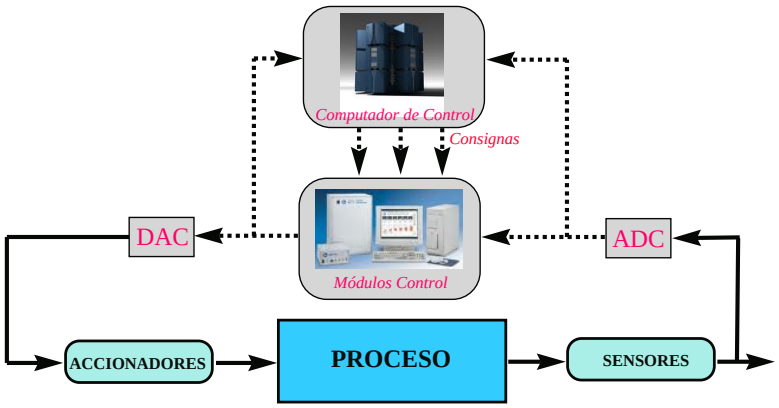

Figura 7.5: Control supervisorio.

Muchas de las aplicaciones del control supervisorio son muy simples y están basadas en el conocimiento de las características del estado estacionario de la planta. Si bien, muy pocos sistemas emplean algoritmos de control complejos, en la actualidad existe la tendencia para la implementación de sistemas complejos como la optimización multivariable, el control predictivo multivariable robusto, control multivariable robusto, sistemas in-

teligentes, entre otros [2]. Muchas de las técnicas empleadas en el control supervisorio han incluido optimización basada en programación lineal y simulaciones que involucran modelos no-lineales complejos de la dinámica de la planta, con algoritmos en tiempo real que se ejecutan en paralelo con la planta en operación.

7.2.3. Control jerárquico

La combinación del CDD y el control supervisorio constituyen una estructura muy sencilla de *control jerárquico*. Mediante este concepto se ha formalizado la organización de un conjunto de funciones referidas al control integral de una planta mediante el uso de uno o varios computadores [1]. Este es el desarrollo natural de la estructura de toma de decisiones en un complejo de producción, la cual se basa en niveles decisorios.

Figura 7.6: Control jerárquico.

Los niveles de toma de decisiones tienen una descomposición vertical (estructura piramidal), tal como se muestra en la Figura 7.6. En el nivel inferior se realiza la interacción directa con el proceso, es decir, la adquisición de datos y el CDD. En el nivel intermedio se encuentra el control supervisorio donde se establecen los perfiles de producción a base de la optimización de

la operación en un corto plazo. Finalmente, en el nivel superior se conforman los planes estratégicos de mediano y largo plazo, a base de sistemas de información gerencial en línea y las condiciones de mercado y producción, es decir, se ejecuta el control de la planta desde un punto de vista del manejo de los recursos económicos, humanos y ambientales.

Sobre la base de esta estructura piramidal existe la tendencia hacia el control integral de la planta como bien se ha reportado en el modelo CIM, los DCS, los SCADAs y las redes de PLC's.

Referencias

1. J. Aguilar, A. Rios, F. Rivas, O. Teran, L. Leon, and N. Perez. Definición de dominios y paradigmas en una arquitectura de automatización industrial. Technical Report 05-04, Fundacite-Mérida, Mérida, 2004.
2. Joseph Aguilar-Martin. *Inteligencia Artificial para la Supervisión de Procesos Industriales*. Consejo de Publicaciones, ULA, 2007.
3. K. Åström and B. Wittenmark. *Computer Controlled Systems*. Prentice-Hall, 1989.
4. S. Boyer. *SCADA: Supervisory Control and Data Acquisition*. ISA, 1999.
5. G. Franklin, J. Powell, and M. Workman. *Digital Control of Dynamic Systems*. Addison-Wesley Publishing Company, 1990.
6. Norman Nise. *Control Systems Engineering*. Addison Wesley Publishing Company, 1995.
7. Katsuhiko Ogata. *Ingeniería de control moderno*. Prentice Hall International, 1993.
8. Theodore J. Williams. *A Reference Model For Computer Integrated Manufacturing (CIM)*. Instument Society of America, 1991.

Capítulo 8
Arquitecturas de Automatización

Resumen: Siguiendo su evolución tecnológica, en este capítulo se presenta un análisis de diferentes arquitecturas para implantar sistemas para la automatización de procesos.

8.1. Introducción

Las estrategias de automatización de un proceso técnico hacen uso intensivo de la información; tanto interna a los procesos, que indican la disponibilidad de productos, capacidades y condiciones de las plantas, como externa, referida a las condiciones de mercado, que inducen el tipo de producción a ser realizada durante un lapso de tiempo. Esta información es básica para el proceso de planificación de la producción [10, 15].

En consecuencia, para la automatización se debe tomar en cuenta, además de los aspectos mencionados anteriormente, la organización del negocio, las relaciones entre las plantas y de éstas con los elementos de almacenamiento de productos intermedios, productos finales y el despacho. Todo esto hace que la automatización sea difícil de implantar si no se cuenta con una plataforma de datos segura y con un alto grado de disponibilidad, y de elementos de control, tanto decisorio como regulatorio, basados en un modelo universal del proceso, o en múltiples modelos relacionados entre sí, que permitan la implantación de un concepto de automatización integrada de las operaciones de producción [1, 19, 8].

La necesidad de integración de los sistemas de automatización (sistemas de información + sistemas decisorios + sistemas de apoyo a la toma de decisiones) ha sido ampliamente planteada, tanto para la industria manufacturera, donde tiene su máximo exponente en el modelo CIM [18], como para la industria de procesos, descrita en la pirámide de automatización de la ISO.

De una manera general, se puede decir que el proceso de automatización de una planta incorpora las fases de diseño del proceso con técnicas que van desde la simulación hasta el diseño asistido por computadora, para proseguir con las etapas de control automático y sistemas de apoyo a la toma de decisiones en tiempo real.

Un ejemplo de ello corresponde al desarrollo de un proceso de automatización, mediante una integración de las operaciones, la aplicación de técnicas emergentes y avanzadas en el control y una re-instrumentación de la planta [11, 14, 17]. Dentro de los beneficios que se desprenden de este desarrollo de automatización se cuentan: a) Aspectos económicos derivados de la mejora en los procesos como son: mejor calidad de los productos, mayor disponibilidad de la planta y una reducción en el consumo de energía y en otros costos de producción. b) Aspectos relacionados con la seguridad de funcionamiento, seguridad del personal y un mejor índice de protección ambiental.

De esta manera, este estudio procura hacer un descripción de las principales arquitecturas para la automatización industrial.

8.2. Requerimientos de automatización

Uno de los aspectos fundamentales para el logro de la automatización global de la planta, viene dado por la existencia de un ambiente que permita construir un sistema de planificación y secuenciamiento de las operaciones integrado con el resto de las actividades del complejo de producción. Para la integración de las operaciones es necesario disponer de:

1. Una plataforma de datos y sistemas de información con el objeto de disponer, a diferentes niveles, de la información necesaria para la realización de las funciones en las unidades de procesos, así como las actividades de optimización y planificación global del complejo de producción.
2. Modelos internos en cada unidad de producción asociados a la física del proceso, a efecto de realizar una optimización local de los mismos.
3. Modelos asociados a cada unidad de producción, con el fin de inferir: rendimientos, disponibilidad del sistema, capacidad de producción, etc., para poder realizar una planificación de actividades en base a cada unidad de producción.
4. Modelos asociados a las interacciones entre las diferentes unidades.

Estos modelos permiten construir sistemas de control, sistemas de apoyo a la toma de decisiones y sistemas para determinar el estado de un proceso, de una unidad de producción o del complejo mismo. Las funciones se pueden asociar a tres macro-sistemas definidos como:

1. Sistemas de control y optimización local de unidades (Operaciones)

2. Sistemas de integración y programación de las operaciones (Programación)
3. Sistemas de planificación y optimización global de la planta (Optimización).

Por lo tanto, la construcción de un sistema de producción altamente competitivo demanda una infraestructura, además de flexible para la generación de distintos productos bajo diferentes escenarios, ágil e inteligente, capaz de responder apropiadamente según las circunstancias del mercado y los métodos y mecanismos de producción. Esta exigencia implica disponer de sistemas altamente complejos e integrados, que funcionen de manera inteligente, adaptables a condiciones cambiantes de una manera rápida y eficiente. La integración se logra mediante la existencia de:

1. Una plataforma de computación y redes que permita la transferencia de información entre las diferentes operaciones.
2. Una alta disponibilidad de datos confiables del proceso.
3. Una organización de la arquitectura informática, que soporte la interoperación entre las funciones de gerencia y de producción en base a la detección de cambios en las condiciones de operación.

En ese contexto, es necesario conocer el negocio, la infraestructura de comunicación, los sistemas de hardware y los sistemas de información, para determinar la potencialidad del desarrollo de sistemas con el objeto de:

- Establecer el estado de los procesos mediante el uso de elementos de vanguardia tecnológica utilizando la información presente.
- Integrar sistemas de apoyo a la toma de decisiones dentro de la visión de automatización integrada.
- Determinar la factibilidad de usar actuadores/sensores inteligentes integrados a la plataforma de información.

Ello es posible si se establece una visión de automatización integrada del proceso productivo y sus mecanismos de soporte y supervisión [3, 12, 13].

En el marco del control jerárquico surgen los requerimientos de automatización de procesos, los cuales, en función de un modelo, se relacionan a los siguientes beneficios:

- Presentar una visión completa del proceso de automatización.
- Permitir la determinación del mejor método para realizar la automatización.
- Considerar la asignación errónea de recursos o fallas en el proceso.
- Crear diferentes arquitecturas a partir de modelos referenciales basados en las principales ventajas de las mismas.

Así, es importante establecer el modelo de automatización, que en general se relaciona con los sistemas y mecanismos de producción. En tal sentido, es posible concretar, de manera general, los requerimientos de los modelos de automatización:

- El modelo de automatización debe permitir la descripción de todos los aspectos del ciclo de vida del sistema, abarcando todos los conceptos involucrados en el proceso.
- Debe incorporar diferentes puntos de vista para describir por completo el proceso productivo, tales como información y control, equipos, mano de obra, organización gerencial, así como las relaciones con otros procesos.
- Debe ser independiente de la tecnología existente.
- Debe estar abierta a la estandarización.

8.3. Modelos de Automatización

Definición 8.1. Un modelo de automatización consiste en una manera genérica de organizar e integrar componentes de sistemas; sirve como punto de partida para el diseño de un gran número de sistemas en un área de aplicación, especifíca la estructura general del sistema y muestra cuales tareas han de ser ejecutadas, además de permitir al diseñador el uso del modelo para especificar una arquitectura que establezca los aspectos más importantes que deben considerarse durante el proceso de modelado e integración empresarial, como la terminología empleada, estructura del sistema y actividades de cada uno de sus componentes.

Debido al crecimiento y complejidad de los procesos productivos, la mayoría de los sistemas automatizados han sido modelados en estructuras jerárquicas estáticas, con tareas separadas y funciones especializadas. Son múltiples las arquitecturas definidas con el fin de modelar una industria automatizada, con el fin de determinar y facilitar la integración de las actividades desarrolladas. En el caso de los procesos productivos se encuentran principalmente los modelos de automatización jerárquicos y heterárquicos [17, 6].

8.4. Modelos Jerárquicos

Los sistemas jerárquicos típicamente tienen una estructura rígida que les impide reaccionar de una manera ágil ante variaciones. Dentro del contexto de automatización la palabra jerarquía o sistema jerárquico debe ser interpretada como un sistema de subsistemas flojamente interrelacionados, cada uno de los últimos siendo a su vez jerárquicos hasta llegar a algún nivel

mas bajo de subsistemas elementales. En arquitecturas jerárquicas, los diferentes niveles no pueden tomar la iniciativa, de esta manera el sistema es vulnerable ante perturbaciones, dado que su autonomía y reactividad ante disturbios son débiles. La arquitectura resultante es muy rígida, por lo tanto, es costosa de desarrollar y difícil de mantener.

La modificación de las estructuras automatizadas para agregar, quitar o cambiar recursos es difícil, ya que requiere actualización de todos los niveles para reconocer el estado de todo el sistema. Además, los fallos ocurridos en niveles inferiores se propagan hacia los superiores invalidando en algunos casos la planificación y afectando el funcionamiento de las demás tareas inherentes a la automatización.

8.4.1. Características de los modelos jerárquicos

Estos modelos presentan las siguientes características:

- Siguen la estructura humana gerencial de la planta.
- Promueven el principio de autonomía (la responsabilidad puede ser delegada hacia los niveles inferiores de la jerarquía)
- Promueven el principio de localidad (las unidades de la planta son usualmente distribuidas pero también usualmente comprimidas donde el control distribuido puede ser aplicado)
- Permiten la distribución de las tareas de la planta a sistemas multicomputacionales debido a la disposición en capas de las funciones de control dentro de la jerarquía.
- Existe flexibilidad en la introducción de nuevos tecnologías.
- Las funciones mas altas de la jerarquía tienden a enfocarse en planificación mientras que los niveles bajos se centran en la ejecución.
- Necesidad de limitar la complejidad de entidades individuales para facilitar la comprehensión humana y la manejabilidad computacional.
- Robustez, predictibilidad y eficiencia.

Los modelos jerárquicos disuelven el vínculo entre el tamaño y la complejidad, en virtud de la jerarquía, la complejidad de una organización, tal como se evalúa desde cualquier posición dentro de ella es casi independiente de su tamaño total. Además, se reduce la necesidad de transmisión de información entre los diferentes elementos que conforman la organización. Un nivel solo necesita información detallada sobre las actividades correspondientes a su nivel e información sumaria adicional sobre el comportamiento medio en otras unidades. Es suficiente saber cuales son las salidas que el subsistema debe proporcionar a otros entes, sin importar a qué procesos serán sometidas por este último [2, 19].

8.4.1.1. Modelo de Automatización Piramidal

Caracterizando las operaciones que se realizan en una industria de procesos, aparece una jerarquía funcional piramidal y una interrelación entre las mismas. Las operaciones más complejas, en cuanto al tipo de decisiones y los modelos asociados a la toma de las mismas, se encuentran en los niveles superiores de la pirámide de automatización, pero son menos frecuentes en tiempo, que las decisiones existentes en los niveles que están más cerca del proceso físico, tal como se muestra en la Figura 8.1.

En los niveles más altos de la pirámide de automatización, la información utilizada engloba las diferentes unidades productivas y sus interacciones con el mercado, aspectos financieros, disponibilidad de suministros, políticas gubernamentales, etc. En los niveles más cercanos al proceso físico, las funciones realizadas siguen las directivas de los niveles superiores, pero optimizan las operaciones internas a cada unidad.

Así, el modelo piramidal consta de diferentes niveles que abarcan las distintas actividades y funciones de una planta coordinada de manera jerárquica, cubriendo desde los aspectos de control de los procesos físicos en su nivel mas bajo, hasta los niveles donde se realizan las funciones corporativas de la planta. Cada nivel se caracteriza por un tipo de información y de procesamiento diferente, siendo necesaria la integración del proceso automatizado para incluir la comunicación interna en cada nivel, y entre niveles, con el fin de lograr sistemas que permitan ejecutar las diferentes tareas de control existentes en una empresa.

Este es el modelo de automatización más difundido en el ambiente de producción de manufactura y adaptada a la producción continua, siguiendo las consideraciones establecidas por la ISO. En este modelo se distinguen, fundamentalmente, 4 niveles con una clara separación del procesamiento de la información y diferente manejo de puntos de vista técnicos.

- Empresa y planificación: nivel mas alto y donde se fijan las estrategias y metas a cumplir, de acuerdo a la planificación, que se realiza según las estrategias fijadas, así como la gestión de la producción. Se considera como en el nivel de control estratégico.
- Supervisión y optimización: donde se realiza el monitoreo de las diferentes unidades de producción, y se verifica la sincronización y coordinación de los procesos inter-dependientes. Se define como el nivel de control supervisorio.
- Control local: se compone de todos los elementos destinados al control regulatorio de los procesos.
- Proceso: abarca tanto el proceso físico como los instrumentos asociados, los cuales están en contacto con la operación y control de dicho proceso.

El esquema de actividades por niveles permite estandarizar procedimientos y soluciones a problemas, además de proveer una visión de la arquitectura de automatización. En este modelo, la granularidad de la información

crece a medida que se va bajando en los niveles, y la toma de decisiones es jerárquica. Las características de la organización hace posible el modelado de la plataforma de automatización siguiendo este esquema jerárquico por niveles.

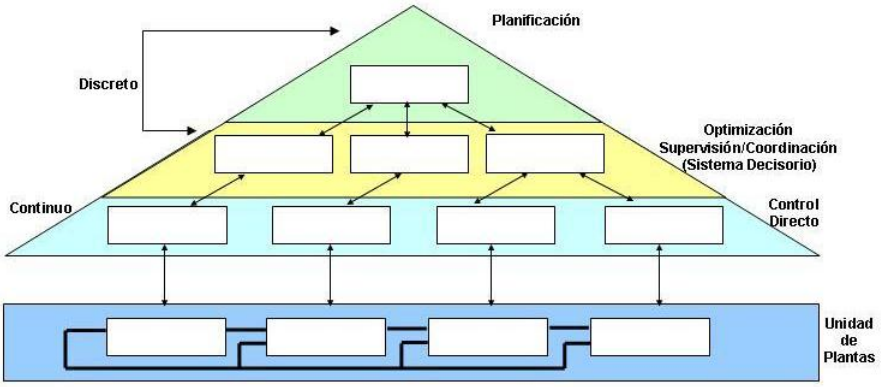

Figura 8.1: Modelo de automatización piramidal

En base a este modelo, para una funcionalidad integrada, los sistemas de información permiten la realización de la integración de las operaciones mediante una plataforma de datos y transformaciones de los mismos, de tal manera que presten información (movimiento de datos ascendente) y faciliten la transmisión de lineamientos (movimientos de datos descendente) entre los diferentes niveles, así como también información horizontal dentro de un mismo nivel. El esquema de la Figura 8.2 nos muestra como los sistemas de información apoyan las distintas actividades realizadas en procesos productivos.

8.4.1.2. Modelo de Referencia CIM

Este modelo, que trata de definir los requerimientos comunes de todas las implantaciones de procesos de manufactura, es una colección detallada de las tareas de gerencia de la información genérica y de control automático y sus necesarias especificaciones funcionales para plantas de manufactura. Para su implantación, en este modelo se dividen las tareas del control de la planta en capas funcionales. Es representado desde el punto de vista jerárquico por una estructura de once capas (ver la Figura 8.3), donde los elementos físicos del sistema son representados por las cinco capas inferiores mientras que los elementos de software son representados por las

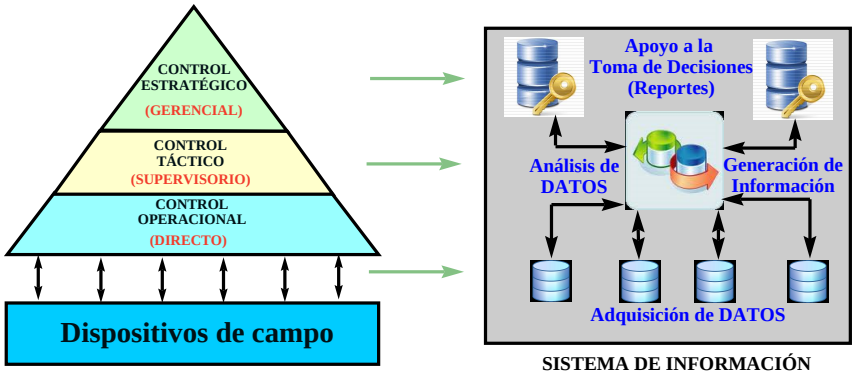

Figura 8.2: Automatización según los sistemas de información.

seis capas superiores. La descripción detallada de estas diferentes capas se puede revisar en [18].

- Nivel 1 Ambiente Físico (incluyendo humanos).
- Nivel 2 Chequeo de datos de entrada-salida.
- Nivel 3 Comunicación.
- Nivel 4 Base de datos de procesos y tareas.
- Nivel 5 Elementos del sistema de computación.
- Nivel 6 Administración de los recursos del sistema.
- Nivel 7 Código compilado o interpretado.
- Nivel 8 Código fuente de lenguaje de alto nivel.
- Nivel 9 Algoritmos y procedimientos genéricos.
- Nivel 10 Procesos específicos o modelos matemáticos de la planta.
- Nivel 11 Declaración de tareas.

La arquitectura resultante a partir de este modelo referencial comprenderá:

1. La definición de entidades y de tareas relacionadas.
2. La relación entre entidades.
3. El flujo de datos requeridos entre entidades.
4. La definición de la estructura de la gerencia de los datos y las necesidades de diccionarios de datos.
5. Las interfaces para influencias externas.

Figura 8.3: Modelo de referencia CIM.

8.4.2. *METAS Método para la Automatización Integral de Sistemas de Producción Contínua*

Desarrollada en la Universidad de los Andes, es una metodología de automatización que sirve de guía para la elaboración de planes estratégicos de automatización integral que permitan definir la estructura global integrada de información, gestión, comunicación y control que requiere una empresa de producción continua [7]. Propone la incorporación de actividades de modelaje orientado a objetos para la representación de los procesos de negocios y para la organización de la información [6]. Hace referencia a un modelo de automatización útil para la caracterización y ubicación de los sistemas encontrados en la empresa y las funciones llevadas a cabo por ellos. Propone la representación de la empresa por medio de caras que abarcan diferentes áreas de la misma agrupando las tareas de planificación, coordinación, supervisión y control, (ver la Figura 8.4).

Así, la arquitectura está fundamentada en la clásica pirámide de automatización y contempla, además del proceso físico inherente a la actividad central de una empresa de producción, cinco caras o elementos integrados que representan diferentes aspectos de una empresa.

- Arquitectura de procesos de decisión.
- Arquitectura de objetos de datos.

- Arquitectura de aplicaciones de software.
- Arquitectura de tecnologías de información y comunicaciones.
- Arquitectura de tecnologías de producción.

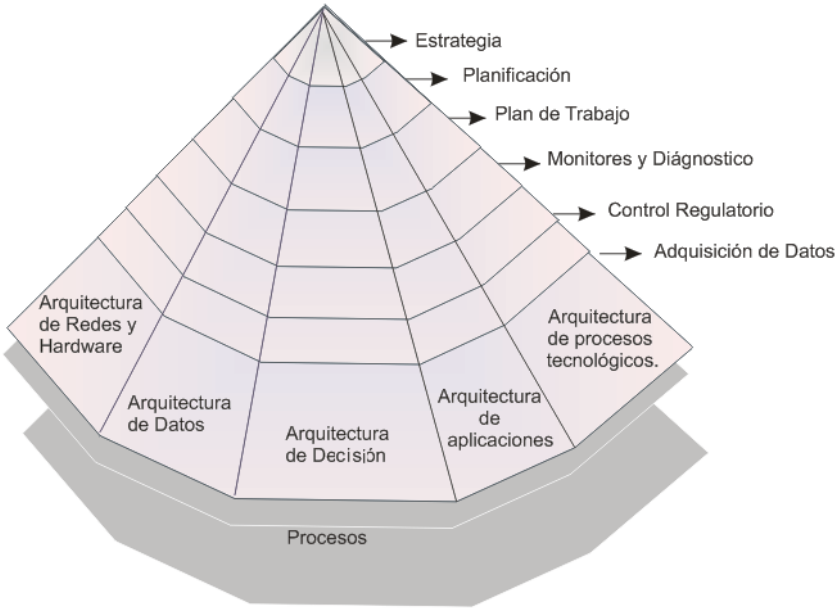

Figura 8.4: METAS

8.5. Modelos Heterárquicos

Tienen un buen desempeño ante cambios y pueden adaptarse continuamente a su entorno, se basan en la fragmentación del sistema en unidades pequeñas y completamente autónomas. Las arquitecturas heterárquicas se basan en una total autonomía local (control distribuido), resultando en un entorno en el cual los componentes cooperan para alcanzar objetivos globales gracias a la toma de decisiones locales. Estos componentes autónomos son agentes, y la cooperación se estructura a través de protocolos de negociación. El enfoque heterárquico prohíbe toda tipo de jerarquía con el objeto de dar todo el poder a los módulos básicos, se desempeña generalmente bien en entornos simples [16].

Al eliminar las relaciones de jerarquía en el sistema, los módulos cooperan como iguales dando lugar a una arquitectura plana, en lugar de asignar relaciones de subordinación y supervisión. Para proveer la robustez y flexibilidad necesaria se establecen esquemas de comunicación, donde al existir fallas en un módulo otro pueda ejecutar sus actividades. El punto principal en los modelos heterárquicos es la coordinación para prevenir la existencia de anarquías en la arquitectura y aprovechar las ventajas de distribución, modularidad, mantenimiento y reconfiguración.

8.5.1. Características de los Modelos Heterárquicos

Estos modelos exhiben las siguentes características:

- Los diferentes elementos de la arquitectura se comunican como iguales sin existencia de jerarquías, a través de reglas que definen las negociaciones y cooperación entre las tareas.
- Debe ser capaz de autoconfigurarse, ser escalable y tolerante a fallas.
- Especial énfasis en el proceso de coordinación, ya que no existen niveles superiores que observen a todo el sistema ni información global del mismo.

8.5.2. PROSA: Process Resource Order Staff Architecture

Propone la descentralización de las decisiones por medio de la atribución de cierto nivel de dependencia a cada unidad que compone la arquitectura, la cual básicamente, se compone de tres entidades semi-independientes y cooperantes denominadas holones, con capacidad de tomar decisiones y ejecutar acciones, (ver la Figura 8.5). Su deficiencia radica en la gran cantidades de unidades necesarias para modelar una empresa compleja siguiendo este modelo, lo cual dificulta el flujo de información y producción. Este problema es solventado proponiendo una modificación del modelo a una disposición jerárquica de holones auxiliares como elementos óptimos para la asistencia de los holones básicos en la ejecución de sus tareas, permitiendo la incorporación de soluciones centralizadas. La incorporación de este arreglo jerárquico de holones no introduce rigidez al sistema por ser sólo elementos de asistencia para la arquitectura, [5].

La arquitectura resultante reduce la complejidad para integrar nuevos componentes y habilita la fácil reconfiguración del sistema, permitiendo a los holones realizar diferentes actividades concernientes a otros niveles, [20].

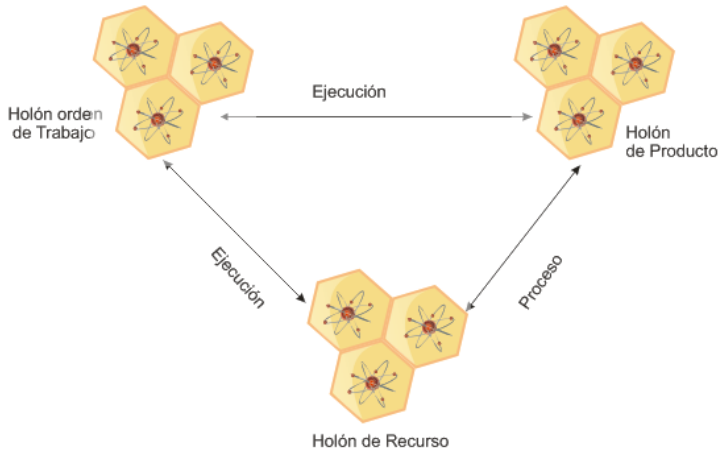

Figura 8.5: PROSA

8.5.3. *PROHA:* Product Resource Order Heterarchical Architecture

PROHA es un modelo construido a partir de PROSA, el diseño se basa en el planteamiento de un sistema heterárquico de agentes deliberativos. Esta metodología se adapta a los sistemas de grandes dimensiones y alta complejidad, ya que permite crear sistemas de software que representan cada uno de los elementos que configuran el sistema productivo a fin de facilitar su gestión y control, [20]. Esta metodología se divide en tres procesos.

- Identificación de agentes.
- Diseño de agentes.
- Definición de protocolos de interacción.

Esta arquitectura dota a los sistemas productivos de un control de alta eficiencia y flexibilidad, puesto que aporta la capacidad de una rápida respuesta ante cambios externos y, lo que es también importante, ante cambios internos, con la búsqueda de distintas alternativas. Además, PROHA aporta un aumento claro de la capacidad de las plantas productivas, que redunda económicamente en las empresas.

8.5.4. Tabla comparativa de modelos de automatización

En base a las características de los modelos jerárquicos y heterárquicos, en la Tabla 8.1 se presenta una comparación de esos esquemas de automatización.

Modelos Jerárquicos	Modelos Heterárquicos
Basado en jerarquías	Se prohíbe todo tipo de jerarquía (Arquitectura plana)
Estructuras relativamente rígidas	Buen desempeño ante cambios
Autonomía débil	Total autonomía local
Posible propagación de fallas	Robustez y flexibilidad para evitar la propagación de fallas
Coordinación ejecutada en los niveles superiores de la arquitectura	Posibles problemas de coordinación

Tabla 8.1: Comparación de Modelos de Automatización

8.6. Modelos Híbridos

Constituido por aquellos modelos que combinan las ventajas de los sistemas jerárquicos y heterárquicos, al mismo tiempo que evitan sus desventajas. Dentro de esta clasificación se encuentran las arquitecturas holónicas caracterizadas por:

- Otorgar autonomía a los módulos individuales para evitar las estructuras rígidas de los sistemas jerárquicos. Esto dota al sistema de un mecanismo para generar respuestas rápidas ante perturbaciones y la habilidad para reconfigurarse a sí mismo ante nuevos requerimientos.
- Poseen ciertas heterarquías para prevenir la existencia de jerarquías, como sucede en los sistemas piramidales.
- Los elementos de la arquitectura pueden pertenecer a múltiples jerarquías flexibles, o formar jerarquías temporales, donde no existe dependencia de niveles para el funcionamiento del modelo.
- La existencia de jerarquías flexibles asegura el desempeño controlable y predecible de la arquitectura.

Dos modelos híbridos muy interesantes, se presentan en la Sección 10.5 y en la Sección 10.10.

En resumen, la automatización industrial ha pasado desde los sistemas centralizados, a sistemas distribuidos llegando al gran reto de los sistemas integrados, ver la Figura 8.6.

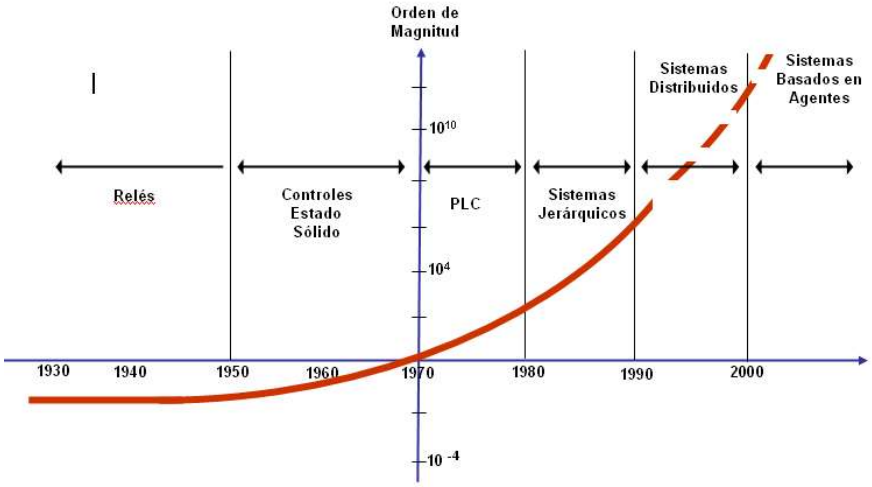

Figura 8.6: Evolución histórica de la automatización.

Así, el futuro de la automatización industrial estará regido por un *sistema integrado*, basado en agentes inteligentes, construido a partir de las tecnologías de información-comunicación y de las tecnologías de automatización de procesos, empleando todas las fortalezas de las técnicas de inteligencia artificial distribuida y de optimización [9, 4]. En este sistema integrado se dispondrán de redes de campo, redes de unidades de producción, redes corporativas, etc., las cuales darán soporte a todas las operaciones de complejo productivo, donde la inteligencia estará disponible, funcionalmente, en todos los procesos, según el modelo de negocio. Esto conlleva a un *aplanamiento* de las arquitecturas jerárquicas hasta alcanzar una automatización heterárquica holónica, ver la Figura 8.7, donde se disponen de sistemas que operan integral y coordinadamente. Estos sistemas son recursos, bienes o servicios que son ofrecidos como productos por socios.

Esta estrategia de automatización a futuro podría garantizar una alta disponibilidad del proceso con bajos costos de producción, siguiendo estrategias proactivas de negocio, de acuerdo a métricas bien definidas para el mejoramiento continuo y la evaluación de los índices de desempeño. Esta estrategia de funcionamiento proactivo es posible a través de la inteligencia artificial distribuida aplicada en las distintas fases operativas, según el modelo de negocio.

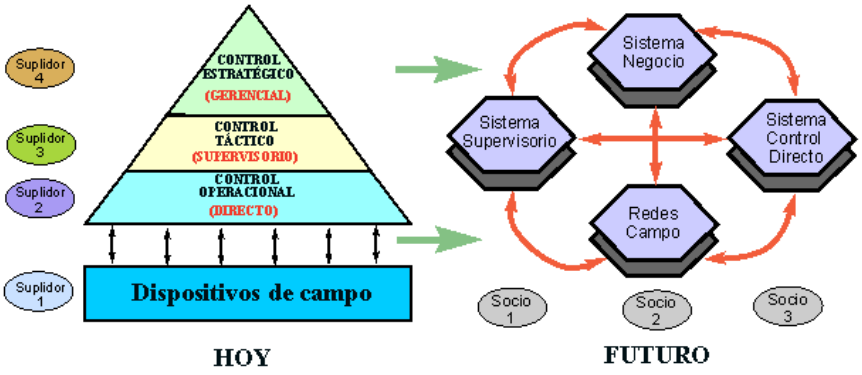

Figura 8.7: Automatización a futuro.

Referencias

1. J. Aguilar, A. Rios, F. Rivas, O. Teran, L. Leon, and N. Perez. Definición de dominios y paradigmas en una arquitectura de automatización industrial. Technical Report 05-04, Fundacite-Mérida, Mérida, 2004.
2. P. Antsaklis and K. Passino. *An introduction to autonomous and intelligent control*. Kluwer, Boston, U.S.A., 1993.
3. C. Bravo, J. Aguilar, F. Rivas, and M. Cerrada. Design of an architecture for industrial automation based on multi-agents systems. In *Proceeding of the 16th IFAC World Congress*, Prague, 2005.
4. C. Bravo, J. Aguilar-Castro, A. Ríos-Bolívar, J. Aguilar-Martin, and F. Rivas. Arquitectura de referencia para integración en empresas de producción industrial basada en la inteligencia artificial distribuida. *Revista Gerencia Tecnológica Informática, GTI*, 6(15):13–25, 2007.
5. H. Van Brusel, J. Wyns, L. Valckenaers, and P. Peeters. Referente architecture for holonic manufacturing systems: Prosa. *Computer In Industry*, 37:255–274, 1998.
6. Jesús Calderón. Integración de unidades de producción: Una estrategia para la automatización de procesos industriales. Master's thesis, ULA, Postgrado de Instrumentación y Automatización, 2001.
7. Edgar Chacón, Isabel Besembel, and Jean Claude Hennet. Coordination and optimization in oil & gas production complexes. *Computers in Industry*, 53(1):17–37, 2004.
8. F. Hidrobo, A. Ríos-Bolívar, J. Aguilar, and L. León. An architecture for industrial automation based on intelligent agents. *WSEAS Transaction on Computers*, 4(12):1808–1815, 2005.
9. Sirkka-Liisa Jämsä Jounela. Future trends in process automation. *Annual Reviews in Control*, 31(2):211–220, 2007.
10. J.M. Mendesa, P.J. Leit aoc, F.J. Restivob, A.W. Colomboa, and A. Bepperling. Engineering of service-oriented automation systems: a survey. *Innovative Production Machines and Systems*, 1:33–38, 2008.
11. Jim Pinto. Instrumentation & control on the frontiers of a new millennium. *Journal Instruments & Control Systems*, 1(2):78–90, 2000.
12. A. Ríos-Bolívar, F. Hidrobo, M. Cerrada, and J. Aguilar. Control and supervision system development with intelligent agents. *WSEAS Transactions on Systems*, 6(1):141–148, 2007.

13. Tariq Samad, Paul McLaughlin, and Joseph Lu. System Architecture for Process Automation: Review and trends. *Journal of Process Control*, 17:191–201, 2007.
14. Olli Ventä Teemu Tommila and Kari Koskinen. Next generation industrial automation: Needs and opportunities. *Automation Technology Review*, pages 34–41, 2001.
15. O. Terán, F. Narciso, A. Ríos-Bolívar, F. Hidrobo, J. Alvarez, L. León, J. Aguilar, and D. Hernández. Un marco metodológico para el desarrollo de aplicaciones para automatización industrial. *Revista de la Facultad de Ingeniería*, *UCV*, pages 57–69, 2009.
16. Vyatkin Valeriy. Oooneida. open object-oriented knowledge economy for intelligent industrial automation. In *Proc. 1st IEEE INDIN Conference*, Banff, Canada, 2003.
17. Lingfeng Wang and Kay-Chen Tan. *Modern Industrial Automation Software Design*. IEEE Press, 2006.
18. Theodore J. Williams. *A Reference Model For Computer Integrated Manufacturing (CIM)*. Instument Society of America, 1991.
19. Henry Wu, David P. Buse, Junqiu Feng, Pu Sun, and J. Fitch. e-automation: an architecture for distributed industrial automations systems. *Int. Journal of Automation and Computing*, 1(1):17–25, 2004.
20. J. Wyns. *Referente architecture for Holonic Manufacturing Systems - the key to support evolution and reconfiguration*. PhD thesis, Universidad Católica de Leuven, Bélgica, 1999.

Capítulo 9
Integración en Automatización

Resumen: En este capítulo se presenta un análisis de la problemática de integración en automatización de procesos.

9.1. Introducción

Como ha sido mencionado, debido a las exigencias de productividad, seguridad operacional y rentabilidad, la automatización de procesos demanda TICs sofisticadas y complejas, que deben operar dentro de diferentes escenarios y con distintas características para el manejo de datos e información. Así, el número, la variedad, y la sofisticación de las aplicaciones informáticas para la automatización de procesos continúa creciendo, y la arquitectura de sistemas de automatización determina cuan rápido y productivamente pueden ser desarrolladas, ejecutadas y mantenidas tales aplicaciones. La arquitectura impacta en las aplicaciones a través de funciones y características tales como el soporte para los tipos de datos, los mecanismos de comunicación entre procesos, las políticas de planificación en tiempo real, los mecanismos de migración en línea, la integración de componentes y la interoperabilidad de bloques modulares.

En ese contexto, la integración de diferentes tecnologías existentes en una empresa se ha constituido en un factor determinante en el proceso productivo, de manera especial cuando estos procesos se desarrollan bajo alguna estrategia de automatización. En este escenario, se requiere el manejo óptimo de la información disponible, desde el nivel operacional, donde se ejecutan acciones de control sobre la planta, hasta los niveles de planificación corporativa, donde se toman decisiones que afectan el desempeño global de la empresa. Es por ello que, para la integración, los elementos fundamentales son: los productores de datos, la infraestructura de comunicación, y los

mecanismos de procesamiento de datos; componiendo así la información necesaria en las estrategias para la automatización integrada. Por lo tanto, el dominio de integración es el que define como los componentes de la arquitectura de automatización, generalmente distribuidos, son integrados de forma tal que puedan comunicarse, interactuar, compartir datos, sin importar el lugar donde residan; mediante buses de campo en tiempo real con servicios de funcionamiento integral [8].

A partir del modelo referencial de automatización piramidal, la integración puede ser definida como:

Definición 9.1. La integración es un proceso que incluye la producción de datos, la infraestructura de comunicación, los mecanismos de procesamiento de datos, la comunicación interna en cada nivel y la comunicación entre niveles, con el fin de lograr sistemas que permitan ejecutar las diferentes tareas de control y gestión existentes en una empresa.

Otra definición alternativa es propuesta en [13], donde se presenta la integración como la coordinación de las operaciones de todos los elementos de la empresa que trabajan en forma conjunta para lograr el cumplimiento, en forma óptima, de la misión de la empresa. Fundamentalmente, la integración se basa en el desarrollo de modelos de intercambio de información, de cooperación y de representación común de los datos, para poder coordinar y ejecutar las acciones u objetivos planteados.

La integración abarca la interrelación entre las distintas áreas del conocimiento existentes en una empresa, entre éstas:

- Investigación de operaciones.
- Computación.
- Control de procesos.
- Redes de comunicaciones.

La necesidad de integración tiene su origen en distintos factores que se presentan, tradicionalmente, al momento de coordinar el trabajo de los diversos elementos involucrados en las distintas fases del proceso productivo, entre estos factores están:

- Diversidad de marcas, sistemas operativos, protocolos de comunicación y bases de datos. Algunos de estos protocolos de comunicación son primitivos, ineficientes y específicos para la transmisión de una información predeterminada. Específicamente, en el nivel operacional puede existir toda una variedad de protocolos industriales, poco o nada compatibles entre si.

- Dispersión de datos con redundancia parcialmente controlada. En algunos casos la misma información es representada de varias formas distintas (nomenclaturas diferentes), lo cual obliga a mantener tablas de conversión, por lo que se hace necesario implantar sistemas complejos de mantenimiento y actualización.
- Necesidad de grandes esfuerzos para el soporte y mantenimiento de la estructura, tanto en equipos (hardware) como en programas (software), cuya obsolescencia es rápida.
- La interacción con los diversos sistemas, a través de interfaces de usuario disímiles y poco amistosas.
- Poca integración entre las áreas de control de procesos, informática e instrumentación; esto debido a la rápida evolución de cada área.

Es así como, a los fines de llegar a un proceso productivo plenamente integrado, los diversos sistemas para el soporte y supervisión de la producción deben estar definidos y caracterizados según la arquitectura de la empresa, en donde estén bien establecidos los objetos y procesos de negocio que la componen y los mecanismos de supervisión sobre los mismos, de tal manera que ello facilite la integración, mediante:

- El uso de estándares, posiblemente abiertos. Este aspecto es de suma importancia, ya que son normativas establecidas por asociaciones, comisiones y organizaciones internacionales que regulan los aspectos relacionados con la programación y interoperabilidad, para facilitar la interacción de aplicaciones y dispositivos de manera uniforme.
- La integración en software y soporte de comunicaciones, lo cual incluye los mecanismos para integrar datos y aplicaciones dentro de la plataforma de automatización, con el objeto de estructurar un bus de intercambio de información. Además, se deben especificar los requerimientos en cuanto a medios de transmisión y velocidad de transmisión, para la integración de datos en los diferentes niveles del proceso productivo.
- La integración en hardware, que considera las características de interfaces para múltiples protocolos entre los diferentes dispositivos que se interconecten en la plataforma de automatización. En lo posible, debe permitir el uso de dispositivos distribuidos con interfaz para redes multipunto y con especificación abierta.
- La agregación y disgregación, que debe responder al establecimiento de mecanismos que permitan solventar la contradicción entre distribución e integración, por medio de la descomposición del modelo productivo total como una agregación de modelos que representan el comportamiento del proceso y las unidades de producción, lo que simplifica el proceso de toma de decisiones.
- El manejo de la heterogeneidad de tecnologías, mediante una definición clara de la funcionabilidad y sus propiedades, lo que probablemente conduce a disponer de tecnologías libres y estándares abiertos.

- El manejo de la inteligencia en los procesos, lo cual se orienta a disponer de dispositivos inteligentes en los distintos niveles, con el objeto de delegar funciones complejas y de optimización, entre otras. Esto conlleva a disponer de dispositivos con características de autonomía, sociabilidad y subordinación interoperables, hacia una plataforma horizontal a nivel físico. En definitiva, disponer de dispositivos con cierta inteligencia e interoperables.

En ese sentido, a continuación se describen los modelos conceptuales que explican los esquemas de integración, y las tecnologías de información y de comunicación que se pueden disponer para efectos de llegar al concepto de *empresa inteligente de producción industrial integrada*.

9.2. Esquemas de Integración

Para sustentar y dar soporte a todas las etapas del proceso productivo existentes en una empresa de producción industrial, en general, se dispone de un conjunto variado de sistemas de información y control, los cuales deben trabajar en forma conjunta y coordinada para lograr el objetivo de producción propuesto. En este punto, es clave el esquema de integración a usar, debido a que muchos de estos sistemas han sido desarrollados en momentos diferentes y usando tecnologías distintas. También, es común que estos sistemas operen en plataformas computacionales diferentes o con diferentes tecnologías, y se ejecutan sobre distintos sistemas operativos.

Para atacar este problema han surgido diversos esquemas de integración. Un primer esquema se concentró en la integración de equipos de computación y de control por medio de redes locales, con el objeto de brindar conectividad entre equipos y garantizar la transmisión de información entre los mismos.

La integración de los sistemas de información es un problema que va más allá de la simple interconexión física de los equipos. Para la solución de este problema han surgido propuestas que van desde la integración de estos sistemas a través de una interfaz gráfica común, hasta llegar a la integración de los procesos de negocios inter-empresas basados en nuevas tecnologías de información y comunicaciones.

Desde el punto de vista estructural, la integración se puede plantear de acuerdo al modelo de automatización industrial que se esté empleando, o de acuerdo a la estructura gerencial en la que se fundamenta la empresa automatizada:

- Modelado vertical del proceso, asociado a una jerarquía de decisiones.
- Modelo horizontal del proceso, a partir del flujo de información y de productos, así como en enfoques planos (heterarquías).
- La visión holárquica.

Con base en estos modelos de empresa, se encuentran distintos modelos de integración asociados a cada uno de estos enfoques:

- Integración Vertical.
- Integración Horizontal.
- Sistemas de Integración Holónica.

9.3. Integración Vertical

La integración vertical está vinculada con el flujo de decisiones entre niveles gerenciales y retroalimentación de información. En este caso, el manejo de las decisiones tiene una estructura totalmente jerárquica, en donde a las distintas funciones de la empresa se le asocia un orden de jerarquía.

Para el caso de integración vertical, el modelo más utilizado es la pirámide de automatización, la cual provee funciones de control directo (regulatorio, secuencial), supervisión (implica el manejo de la parametrización de los controladores), coordinación, optimización y planificación de las operaciones dentro de una planta. Otros modelos comúnmente utilizados son: CIM [12], PERA [14] y SP-95 [9].

La integración vertical involucra:

- El manejo de las decisiones con una estructura totalmente jerárquica.
- Algún modelo basado en la pirámide de automatización u otro modelo jerárquico.
- El uso de las TICs asociadas a la descomposición por unidades autónomas.
- División en varios niveles, según requerimientos.

El modelo de integración vertical puede ser dividido en una serie de capas o niveles, estos niveles permiten avanzar gradualmente en la integración de sistemas desde un nivel netamente físico, hasta llegar a un nivel de integración de procesos de información, control y decisión.

La Figura 9.1 muestra un ejemplo para una arquitectura integrada en niveles o capas.

Con base en los requerimientos y las necesidades de integración que se pueden encontrar en las distintas empresas de producción, se distinguen seis niveles básicos de integración a nivel industrial.

- Integración basada en redes.
- Integración de datos.
- Integración de procesos.
- Integración de software.en
- Integración usando interfaces gráficas de usuario (*GUI*).
- Integración empresarial.

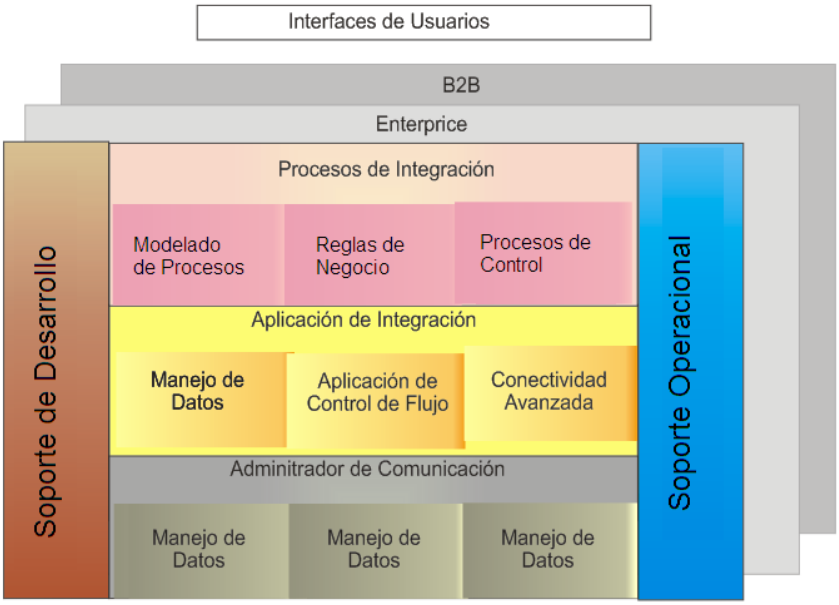

Figura 9.1: Capas de integración.

9.3.1. Integración basada en redes

La integración basada en redes representa la base de los procesos integrados, y se fundamenta en la interconexión e integración de hardware, tales como equipos de computación y sistemas de control, usando como medio de comunicación redes, tanto de área local como de área ancha.

Con ésta se provee el soporte a los distintos procesos que pueden ser encontrados dentro de una empresa de producción; esto es posible por medio de la interconexión de equipos tales como sistemas a tiempo real, redes de control y sistemas de supervisión, lo cual permite disponer de los datos y la información de los procesos en cualquier aplicación de la empresa.

Para que este tipo de integración pueda darse en forma efectiva, la empresa de producción debe contar con los dispositivos necesarios y con las capacidades de interconexión a los distintos tipos de redes, así como también a los dispositivos y equipos involucrados en el proceso operacional.

La interconexión es indispensable en todos los enfoques de integración, si no existe este nivel, no puede haber comunicación entre los sub-sistemas, por lo tanto, no puede haber integración.

9.3.2. Integración basada en datos

En este nivel, la integración se realiza por medio del intercambio o transferencia de datos entre dos o más archivos, aplicaciones o bases de datos. Debido a que los datos pueden encontrase en formatos distintos o no compatibles, es necesario que las distintas aplicaciones que los manejan, implementen transformaciones sobre dichos datos, con el objeto de crear un formato que pueda se comprendido y manejado por otras aplicaciones.

Otra manera de implementar esta integración es mediante el uso de bases de datos comunes, donde distintas aplicaciones intercambian datos accediendo a la base de datos común. La integración por base de datos se puede realizar de dos modalidades diferentes: de base de datos a base de datos, o mediante el uso de bases de datos federadas.

- Para la integración de base de datos a base de datos se emplean mecanismos tales como la replicación de datos, conectores ODBC[1] o portadores de mensajes (*message brokers*).
- Mediante el uso de bases de datos federadas la información se integra a través de un modelo de datos virtual único, el cual establece las correspondencias (*mappings*) con las bases de datos integradas. La evolución en los mecanismos de integración de bases de datos heterogéneas ha facilitado la integración entre sistemas de información ubicados en diferentes niveles de la jerarquía de una empresa.

9.3.3. Integración de procesos

La integración de procesos se basa en el uso intensivo de medios de gestión de servicios (*middleware*), mapas de objetos, datos y transacciones entre los diferentes ambientes o procesos involucrados en el sistema de producción.

Este tipo de integración se caracteriza por la transferencia de información, en forma confiable y segura, entre dos o más aplicaciones, las cuales pueden acudir a distintas técnicas como la representación común de los datos y el pase de mensajes remotos entre objetos. Esta manera de integrar se da directamente a través de procesos, los cuales interoperan mediante interfaces de programación (*APIs*) o mediante la transferencia de mensajes.

La integración mediante interfaces de programación requiere que cada aplicación disponga de un conjunto de APIs, a través de los cuales otras

[1] *Open DataBase Connectivity* (ODBC) es un estándar de acceso a las bases de datos desarrollado por SQL Access Group en 1992. El objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos, almacene los datos.

aplicaciones pueden acceder a los procesos y datos que ella está dispuesta a compartir o hacer visible.

La integración mediante pases de mensajes remotos emplea varios mecanismos tales como objetos distribuidos, servidores de aplicación, monitores de procesamiento de transacciones, etc.

Las arquitecturas de objetos distribuidos CORBA [16], *Enterprise JavaBeans* (EJB) y COM se utilizan normalmente para implementar la integración a nivel de procesos [4, 5].

La integración de procesos es un requisito para la integración empresarial, pues se asume que las diferentes funciones de la empresa cooperan entre sí mediante la interoperación de los procesos que soportan las funciones de la empresa.

9.3.4. Integración usando interfaces gráficas de usuario (GUI)

La integración por medio de interfaces gráficas de usuario nace de la necesidad de la visualización y conocimiento por parte del operador de los estados y variables de los procesos, dicha representación permite al operador tener conocimiento del estado de la planta.

Esta modalidad de integración se da a un nivel visual y puede ser realizada por medio de mímicos e interfaces digitales que agrupan la representación, desde simples procesos hasta un conjunto generalizado de procesos de producción.

La integración por medio de interfaces gráficas permite la recuperación de la información de los procesos independientemente del sistema que la mantenga, la base de datos utilizada, o la plataforma o sistema operativo que se utilice.

Las interfaces gráficas de usuario, generalmente basadas en tecnologías WEB, pueden ser utilizadas tanto para la visualización de la información como para la introducción de datos en forma conjunta, manteniendo la independencia y la separación de cada aplicación.

Esta forma de integración es común en sistemas de información WEB que acceden a diferentes bases de datos con la intención de visualizar su contenido en una misma interfaz.

9.3.5. Integración empresarial

Esta modalidad de integración tiene como objetivo principal compartir información de producción, por medio de procesos o modelos de negocios. Dicha forma de integración agrupa factores tales como procesos de decisión

y manejo de la información de la empresa, a través de un enfoque integral de planificación estratégica y toma de decisiones.

En este nivel, la integración se sustenta en el desarrollo de modelos de intercambio de información y representación común de los datos, y se lleva a cabo por medio de aplicaciones que permiten que los procesos de negocios de una empresa se comuniquen o intercambien información entre ellos o con los procesos de negocios de otras empresas.

Actualmente, existen distintas plataformas que permiten la integración de los procesos de negocios de las empresas, una de las más conocidas son las plataformas orientadas a tecnologías de negocios tales como B2B (*Business-to-Business*) [7].

9.4. Integración Horizontal

La integración horizontal está vinculada con el flujo tecnológico, por ejemplo flujo de materiales, de documentos técnicos, productos e información. En este modelo, distintas unidades de negocio realizan actividades complementarias, donde la información de una depende de la información de la otra. La integración se logra cuando todas las sub-actividades inherentes a la empresa o proceso productivo se integran horizontalmente, logrando de esta manera el objetivo planteado.

Para el caso de integración horizontal, los modelo más utilizados son los flujos de trabajo (*workflows*) y los modelos basados en cadenas de valor.

9.4.1. Flujos de trabajo

Los flujos de trabajo se basan en el estudio de los aspectos operacionales de una actividad de trabajo: la forma en que se estructuran las tareas, como se realizan, el orden correlativo, la sincronización, el flujo de la información que soporta las tareas y la forma de hacer el seguimiento al cumplimiento de las tareas. Los flujos de trabajo permiten compartir la información entre los distintos componentes de los procesos, manteniéndolos relacionados mediante la interacción de las distintas unidades, las cuales conservan su autonomía.

Estos modelos facilitan la automatización de los flujos de trabajo entre procesos, y permiten integrar los procesos de la empresa, rediseñados con la ayuda de nuevas estrategias.

El comportamiento general de un sistema basado en flujos de trabajo se puede obtener mediante la composición y la integración de los distintos elementos que lo conforman.

9.4.2. Modelos basados en Cadena de Valor

El concepto de cadena de valor se enfoca en la identificación de los procesos y operaciones que aportan valor al negocio, desde la creación de la demanda hasta que ésta es entregada como producto final. La cadena de valor da un esquema de descomposición horizontal basado en el flujo de conocimientos, productos e informaciones, entre unidades funcionales del mismo nivel.

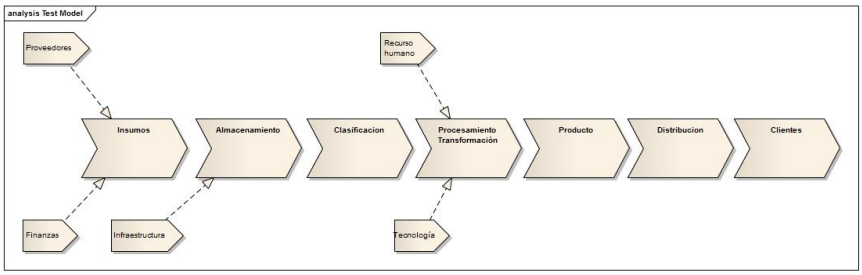

Figura 9.2: Cadena de Valor.

Como puede observarse en la Figura 9.2, en los modelos basados en cadena de valor, las distintas funciones de la empresa se descomponen en subactividades, que integradas horizontalmente cumplen con la función especificada.

9.5. Sistema de Integración Holónica

Las técnicas de inteligencia artificial han sido utilizadas en la fabricación inteligente desde hace más de dos décadas. Sin embargo, los desarrollos recientes en el área de sistemas holónicos han proporcionado nuevos e interesantes resultados de especial interés en esta área.

Recientemente, se ha estado aplicando tecnología de holones a la integración de empresas de fabricación y administración de cadenas de suministro, planificación de producción, asignación de recursos y control, manipulación de materiales, y desarrollo de nuevos tipos de sistemas de producción. Esta tecnología ha sido mayormente aplicada en sistemas de producción de manufactura.

Un Sistema de Fabricación Holónico (HMS) es una holarquía que integra el rango completo de actividades de fabricación, desde reserva de pedidos hasta diseño, producción, control y mercadeo, para obtener una empresa de

fabricación ágil. La potencia de las organizaciones holónicas, u holarquías, es que permiten la construcción de sistemas muy complejos que son, no obstante, eficientes en el uso de recursos, resistentes a perturbaciones (tanto internas como externas), y adaptables a cambios en su entorno.

Dentro de una holarquía, los holones pueden crear y modificar dinámicamente jerarquías. Más aún, los holones pueden participar en múltiples holarquías al mismo tiempo. La Figura 9.3 muestra un ejemplo para una arquitectura de integración basada en holones.

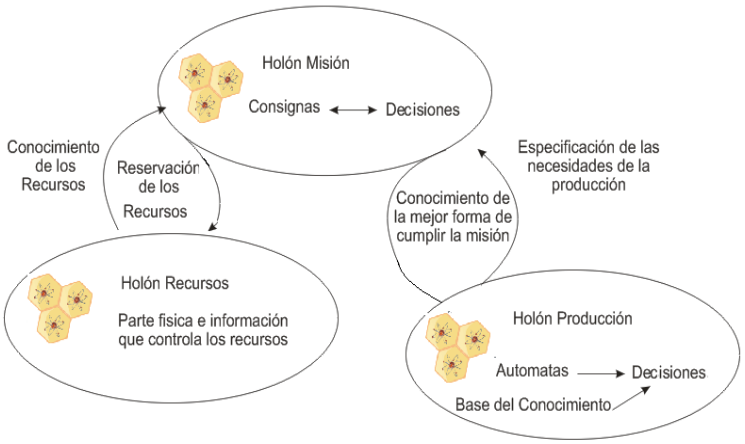

Figura 9.3: Sistema holónico.

La idea de los sistemas de integración holónica es proporcionar procesos dinámicos y descentralizados, en los cuales los holones son la base del procesamiento, y en donde los distintos problemas pueden ser resueltos a través de la integración y distribución de los diferentes componentes de un sistema utilizando mecanismos de agregación y desagregación.

Entre los modelos para integración holónica se destacan la propuesta PROSA [15], y las visiones presentadas por OOONEIDA (*Open Object-Oriented kNowledge Economy for Intelligent Industrial Automation*) [10]. En ellas se incorporan los elementos de cooperación y las arquitecturas para integrar, lo que se denomina la *empresa inteligente*.

9.6. Métodos de Integración

Para lograr la integración de una empresa automatizada es necesario la implantación de distintos métodos que permitan la comunicación, coordi-

nación, cooperación y control de los diversos elementos involucrados en el proceso productivo. Algunos de estos métodos han sido nombrados en las secciones anteriores, a continuación se presenta una lista de los métodos comúnmente usados:

- Implementación de redes de datos y de procesos, tanto de área local como de área ancha. La integración se logra con una combinación de equipos y redes de comunicación que permiten la transferencia de información entre los distintos niveles de la empresa. Este método debe aplicars en todos los enfoques. Si no existe interconexión entre los sistemas, no puede haber integración.
- Uso de tecnología de información. La integración tanto vertical como horizontal se logra a través de tecnologías de información, es decir a través del manejo de la información (base de datos, mensajería, negociación, etc). Es necesario el uso de las tecnologías de información y telecomunicaciones para lograr la comunicación y el intercambio de información de forma confiable y segura.
- Uso de flujos de trabajo. Éstos permiten compartir información entre los componentes del proceso, de manera de mantener integrados los proceso mediante la interacción de las distintas unidades.
- Integración de bases de datos. Para su implantación se emplean mecanismos tales como la replicación de datos, conectores ODBC, o corredores de mensajes (message brokers).
- Mediante interfaces de programación (APIs), o mediante el pase de mensajes remotos entre objetos.
- Objetos distribuidos. Las arquitecturas de objetos distribuidos, tales como CORBA, Enterprise JavaBeans (EJB) y COM, se utilizan normalmente para implementar la integración a nivel de procesos.
- Uso de interfaces gráficas de usuario (GUIs). Este método permite integrar al operador con los procesos por medio de interfaces gráficas y/o mímicos. Generalmente, estas interfaces están basadas en tecnologías Web.
- El uso intensivo de medios de gestión de servicios. Estos se combinan con el uso de mapa de datos, objetos y transacciones, para la integración de procesos, lo que permite mantener actualizada la información de la empresa.
- A través de nuevas tecnologías de negocios tales como el B2B (Business-to-Business).

9.7. Arquitecturas de integración

Desde el punto de vista de arquitecturas de integración, varios enfoques han sido propuestos con el fin de satisfacer las necesidades de integración

de sistemas y/o software presentes en empresas de producción industrial, entre éstos tenemos:

- EAI - Integración de Aplicaciones Empresariales
- Arquitecturas de Objetos Distribuidos
- Aplicaciones de Integración e Ingeniería Empresarial
- El enfoque de Automatización e Integración de Empresa

9.7.1. EAI - Integración de Aplicaciones Empresariales

La Integración de Aplicaciones Empresariales (EAI), se define como la integración de varias aplicaciones existentes con el objeto de compartir libremente información y procesos [5]. También puede ser definida como un sistema que integra las aplicaciones ya existentes para enlazar los procesos de negocio dentro de la empresa y los participantes de la cadena de valor (distribuidores, clientes y socios).

EAI integra nuevas aplicaciones con las ya existentes, incluyendo las aplicaciones heredadas o los paquetes de software, de forma que en conjunto proporcionen las funcionalidades necesarias para soportar los procesos de negocio de la empresa. Esta modalidad de integración permite a la empresa mantener el ritmo de los cambios y reaccionar ante ellos. Se presta especial atención a:

- Sistemas o aplicaciones legadas.
- Bases de datos y sistemas de información.
- Aplicaciones basadas en ERP (*Enterprise Resource Planning*).
- Aplicaciones basadas en WEB.

EAI es un enfoque de integración de software que permite compartir datos y procesos entre diferentes aplicaciones y fuentes de datos interconectadas, también tiene como función comunicar las diferentes aplicaciones mediante conectores, tanto dentro de la organización como inter-organizativa, y coordinar los diferentes procesos de negocio (definen y controlan el orden y secuencia de los elementos del proceso).

La EAI involucra, además de la conexión de las aplicaciones a través de mecanismos de integración (*middleware*), la transformación de la semántica de los datos que son intercambiados entre las aplicaciones [16].

9.7.2. Objetos Distribuidos

Un objeto distribuido es aquel que está gestionado por un servidor y los clientes invocan sus métodos utilizando un mecanismo de invocación remota. El cliente invoca el método mediante un mensaje al servidor que gestiona

el objeto, se ejecuta el método del objeto en el servidor, y el resultado se devuelve al cliente en otro mensaje.

Un sistema de objetos distribuidos es un tipo particular de sistema distribuido cuyos objetos se dispersan en una o más aplicaciones, sobre uno o más computadores ubicados, generalmente, en sitios diferentes e interconectados a través de redes. Los objetos distribuidos tienen la capacidad de ser ejecutados en plataformas de hardware y software diferentes (computadores y sistemas operativos distintos). Requieren para su operación o ejecución de una infraestructura de integración y distribución que permite que los mensajes circulen entre objetos de una aplicación a otra y de un computador a otro.

Las tecnologías más importantes orientadas a objetos distribuidos son:

- **RMI.- Remote Invocation Method:** *Framework* utilizado para crear sistemas distribuidos de Java. El sistema RMI de Java permite a un objeto que se está ejecutando en una Máquina Virtual (MV), llamar a métodos de otro objeto que está en otra MV diferente. Esta tecnología está asociada al lenguaje de programación Java, es decir, permite la comunicación entre objetos creados en este lenguaje.
- **DCOM.- Distributed Component Object Model:** Modelo de Objeto Componente Distribuido, es un conjunto de conceptos e interfaces de programa, en el cual los objetos de programa del cliente pueden solicitar servicios de objetos de programa servidores en otros computadores dentro de la red.
- **CORBA.- Common Object Request Broker Architecture:** Tecnología introducida por el Grupo de Administración de Objetos (OMG), creada para establecer una plataforma para la gestión de objetos remotos independiente del lenguaje de programación [16].

Las características mas importantes de las arquitecturas basadas en objetos distribuidos son:

- Estas arquitecturas atacan directamente los problemas de integración a nivel de procesos y datos, pues permiten que los objetos de un proceso de negocio puedan comunicarse con los objetos de otro proceso a través de la invocación de mensajes remotos, bien sea para consultar datos o requerir la ejecución de una acción.
- Los objetos distribuidos son componentes de software independientes que pueden ser accedidos desde una aplicación cliente remota a través de la invocación de sus métodos.
- Requieren para su operación, de una infraestructura de integración y distribución que permita que los mensajes circulen entre objetos de una aplicación a otra y de un computador a otro.

9.7.3. Aplicaciones de integración e ingeniería empresarial

Según [4], este esquema de integración consiste en la definición, análisis, rediseño e integración de procesos de negocios, procesos de datos y conocimiento, aplicaciones de software y sistemas de información dentro de una empresa, con el objetivo de mejorar el rendimiento global de la empresa.

Este enfoque se centra en la integración empresarial desde una perspectiva global, en la cual están involucrados:

- Los procesos de negocios de la empresa mediante su coordinación, comunicación y cooperación.
- Las redes de computadoras y los protocolos de comunicación.
- El uso de aplicaciones a través de *middleware* (ODBC, JDBC, RPC, CORBA, etc.).

Este enfoque está orientado a las empresas de manufactura o producción discreta. Sus modelos y métodos más conocidos son GRAI [2], CIMOSA [3] y PERA [11]. El modelo **GRAI** analiza y describe la estructura de toma de decisiones de un sistema productivo de manufactura y realiza una propuesta de integración. **CIMOSA** describe la tecnología disponible en cada nivel del modelo CIM (Computer Integrated Manufacturing) y las aplicaciones posibles que sirven para lograr la integración. Finalmente, **PERA** realiza una mezcla de los dos anteriores.

Por otro lado, en [1] se presenta una arquitectura de referencia para la integración de empresas de producción industrial basada en inteligencia artificial distribuida, inspirada en modelos holónicos y sistemas multiagentes. Allí se aborda el modelado de las empresas de producción mediante la definición de una arquitectura que permita representar los objetos y procesos de negocio sobre su plataforma de tecnología de información, implementando mecanismos de supervisión inteligentes a través de técnicas de inteligencia artificial distribuida.

La arquitectura consta de tres capas: *integración*, *modelo de datos* y *capa de gestión*. A partir de esas capas se pueden: a) Representar los objetos de negocio mediante una estructura común que simplifica su modelado. b) Representar diversos niveles de la empresa mediante un enfoque composicional recursivo. c) Representar los flujos de transferencia de productos e información entre las facilidades de la empresa.

9.7.4. Automatización e integración empresarial

Enfoque desarrollado por un grupo de investigadores de la Universidad de los Andes, se le denominó Automatización e Integración Empresarial (A&IE) [6].

Este enfoque se fundamenta en tres componentes conceptuales y metodológicos, estrechamente relacionados:

- El Modelado Empresarial. Describe como elaborar un modelo organizacional de la empresa, que sea consistente con el estado actual de la misma y comprensible para todos sus miembros. El método de modelado empresarial indica como definir los fines (visión, misión y objetivos) de la empresa, los procesos de negocios requeridos para alcanzar los objetivos, los actores que ejecutan tales procesos, y como ellos se agrupan en una estructura organizacional.
- El Modelo Referencial de Automatización e Integración (MRAI). Este modelo define la forma general que tiene la infraestructura de automatización e informática de una empresa integrada. El modelo cubre los procesos de transformación, la infraestructura de equipos de transformación (arquitectura de tecnologías de producción), las decisiones u organización de producción (arquitectura de decisión), y los diferentes aspectos asociados al mantenimiento, transformación y procesamiento de la información (arquitecturas de objetos, y aplicaciones y tecnologías de información y comunicación)
- El método METAS para la automatización e integración empresarial. El objetivo principal de este método es guiar el proceso de elaboración del plan estratégico de automatización e integración de la empresa, mediante la especificación o diseño de cada una de las arquitecturas contempladas en el modelo MRAI.

Referencias

1. C. Bravo, J. Aguilar-Castro, A. Ríos-Bolívar, J. Aguilar-Martin, and F. Rivas. Arquitectura de referencia para integración en empresas de producción industrial basada en la inteligencia artificial distribuida. *Revista Gerencia Tecnológica Informática*, *GTI*, 6(15):13–25, 2007.
2. G. Doumeingts, B. Vallespir, D. Darricar, and M. Roboam. Design methodology for advanced manufacturing systems. *Computers in Industry*, 9(4):271–296, 1987.
3. K. Kosanke, F. Vernadat, and M. Zelm. CIMOSA: enterprise engineering and integration. *Computers in Industry*, (40):83–97, 1999.
4. S.H. Lim, N. Juster, and A. Pennington. Enterprise modeling and integration: A Taxonomy of seven key aspects. *Computers in Industry*, (34):339–359, 1997.
5. D.S. Linthicum. *Enterprise Application Integration*. Addison-Wesley, Boston, USA, 2000.
6. J. Montilva, E. Chacon, C. Arevalo, and G. Urdaneta. Automatización de sistemas e integración de software en sistemas de producción. In *IV Congreso de Automatización y Control*, Mérida, Venezuela, Noviembre 2003.
7. S.S.Y. Shim, V.S. Pendyala, M. Sundaram, and J.Z. Gao. Business-to-business *e*-commerce frameworks. *Computer*, 33(10):40–47, 2000.
8. O. Terán, F. Narciso, A. Ríos-Bolívar, F. Hidrobo, J. Alvarez, L. León, J. Aguilar, and D. Hernández. Un marco metodológico para el desarrollo de aplicaciones para automatización industrial. *Revista de la Facultad de Ingeniería*, *UCV*, pages 57–69, 2009.

9. Keith Unger. *ISA SP-95 Enterprise Control System Integration*. ISA, USA, 2002.
10. V.V. Vyatkin, J.H. Christensen, and J.L.M. Lastra. OOONEIDA: an open, object-oriented knowledge economy for intelligent industrial automation. *Industrial Informatics, IEEE Transactions on*, 1(1):4 – 17, 2005.
11. T. J. Williams. The Purdue Enterprise Reference Architecture. *Computers in Industry*, 24:141–158, 1994.
12. Theodore J. Williams. *A Reference Model For Computer Integrated Manufacturing (CIM)*. Instument Society of America, 1991.
13. Theodore J. Williams and Hong Li. Pera and geram - enterprise reference architectures in enterprise integration. In *DIISM*, pages 3–30, 1998.
14. Theodore J. Williams and Hong Li. *PERA and GERAM: Enterprise Reference Architectures in Enterprise Integration*, chapter Information Infrastructure Systems for Manufacturing II, pages 1–27. Kluwer Academic Publishers, 1998.
15. Jo Wyns. *Reference Architecture For Holonic Manufacturing Systems - the key to support evolution and reconfiguration*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 1998.
16. R. Zahavi. *Enterprise Application Integration with CORBA*. John Wiley & Sons, New York, USA, 2000.

Parte IV
Sistemas Multiagentes en Entornos de Automatización

Preámbulo

Hoy día, los procesos industriales enfrentan grandes retos a fin de lograr altos niveles de producción, con la calidad de producto exigida por el mercado, requiriendo para ello mantener altos niveles de confiabilidad y disponibilidad de los procesos involucrados. Tradicionalmente, la atención estaba dirigida a los sistemas de control local, donde el interés era mantener los puntos de operación de los procesos en niveles adecuados, esperando que esto se reflejara en la obtención de un producto con ciertas especificaciones, tanto en cantidad como en calidad, dentro de las metas planteados por la empresa. El desgaste de los equipos, sea por largos tiempo de uso ó sea por estar sometidos a condiciones de operación desviadas de las condiciones para las que fueron diseñados, empieza a ser un problema que requiere de la atención de un área de la empresa como lo es el mantenimiento. Aparecen, entonces, políticas de mantenimiento a realizar sobre los equipos que demandan una planificación que involucra aspectos humanos y económicos tales como el garantizar la disponibilidad de mano de obra para la realización de tareas de mantenimiento, la disponibilidad de un inventario de partes que permita reparar en lapsos de tiempo oportuno los equipos sometidos a mantenimiento; pero también recientemente demandan necesidades de automatización para la implementación de aplicaciones que permitan implementar tareas de detección y diagnóstico de fallas en línea, requiriendo de la disponibilidad de datos, información y conocimiento, que pueden estar distribuidos.

Un primer requerimiento de integración de actividades surge con la necesidad de coordinar las áreas de control y mantenimiento, donde la aplicación de tareas de mantenimiento pueda requerir de ajustes temporales de los puntos de operación de los procesos controlados, lo que pudiera traducirse en una disminución de los objetivos de producción. Aparece, entonces, otro actor de la empresa, el área de producción, que requiere de datos, información y conocimiento, también distribuido, tanto de las condiciones internas de la empresa, básicamente asociadas a las capacidades instaladas de producción; como a las externas, asociadas a los requerimientos del mercado. En el escenario interno, surge otra necesidad de integración entre el área de producción y mantenimiento, en tanto que la gestión de mantenimiento debe garantizar la disponibilidad de los equipos y mantener disponible la información en cuanto a este aspecto; en el escenario externo, surge la necesidad de integración entre las áreas de control y producción, por cuanto los objetivos de producción se traducen en la fijación de puntos de operación en los objetivos de control de procesos, que a su vez deben enmarcarse en la disponibilidad y confiabilidad de los equipos garantizada por el área de mantenimiento.

El incremento al acceso y disponibilidad de la tecnologías de información como vía para el control e implementación de los planes tácticos de

las empresa, ha hecho que se desarrollen cantidad de aplicaciones de automatización que, en un principio, apuntaban al desarrollo de sistemas de información que permitiesen la organización de los datos de forma local. La aparición de nuevas tecnologías ha soportado la integración de los datos de forma distribuida, primeramente sobre arquitecturas cliente-servidor, dejando atrás a los clásicos sistemas de información locales, hasta llegar al desarrollo de aplicaciones basadas en *servicios web* para alcanzar la integración de conocimiento, información y datos, distribuidos a lo largo de los diferentes niveles ó áreas de la empresa.

Actualmente, donde los avances de la tecnología, en intervalos de tiempo cada vez mas pequeños, ha dejado de ser un problema para lograr altos niveles de automatización industrial, la atención se ha dirigido hacia el uso de las tecnologías de información y comunicación para apoyar a los planes y visión estratégica de la empresa, mas que soportar los planes tácticos que apoyan a dicha visión estratégica. De esta manera, además de requerir la integración de datos, información y conocimiento, se requiere de sistemas capaces de apoyar la toma de decisiones, aprender, adaptarse a los contextos de operación, tanto internos como externos, permitir la generación de nuevo conocimiento e información que emerge de la dinámica propia de la empresa, soportado por mecanismos de comunicación más versátiles y dinámicos.

La tecnología de SMA, como evolución del área de la inteligencia artificial con el fin de lograr sistemas computacionales autónomos, aparece entonces como la alternativa para introducir los nuevos paradigmas de información y comunicación en automatización, en virtud de sus propiedades de autonomía, comunicación, reactividad, inteligencia y movilidad, entre otras. Aún cuando en sus inicios la incorporación de esta tecnología fue recibida de manera bastante tímida en el área de aplicaciones industriales, hoy día existe una importante cantidad de propuestas y implementaciones de herramientas computacionales basadas en SMA para soportar todas las actividades y áreas industriales, que van desde la formulación de arquitecturas, hasta la especificación de los sistemas, listos para su implantación.

En esta parte, dedicada al tema de aplicaciones basadas en el paradigma de multiagentes para la automatización de procesos, se presentan diferentes modelos basados en SMA que los autores han contribuido a desarrollar en el marco de diferentes proyectos de investigación.

En primer lugar, en el Capítulo 10 se analizan las arquitecturas de automatización soportadas por el paradigma de agentes inteligentes, haciendo énfasis en la arquitectura acrecentada y desarrollada por los autores, como un aporte a la aplicación de los sistemas distribuidos inteligentes en el dominio de la automatización y control de procesos.

En el Capítulo 11 se presenta una propuesta que implementa las funcionalidades del dominio de control. Esta propuesta esta pensada para implantarse en una arquitectura específica, y propone una comunidad de agentes que se ocupa no sólo del diseño y ejecución de políticas de control, sino

también de la evaluación de su desempeño. En el Capítulo 12 se presentan diferentes propuestas que implementan las funcionalidades del dominio de supervisión. Una primera propuesta especifica una comunidad de agentes orientados tanto a la supervisión de las tareas de control como al manejo de la confiabilidad de los procesos. Las otras propuestas de este capítulo están enfocadas al diagnóstico de fallas, tareas que han recibido especial atención en la comunidad industrial. Finalmente, el Capítulo 13 presenta dos enfoques de SMA para implementar las funcionalidades del área de planificación. Todos los modelos presentados son especificados usando la metodología MASINA propuesta por los autores y especificada en el Capítulo 3.

Capítulo 10
Arquitecturas de Automatización basadas en Agentes

Resumen: En este capítulo se presenta un análisis de las diferentes arquitecturas de automatización de procesos construidas a partir de agentes inteligentes.

10.1. Introducción

En la industria, la necesidad de adoptar diferentes tecnologías, a fin de responder a las demandas del mercado ha hecho que la complejidad de los procesos de producción se incremente, requiriendo, además, de la integración de todos los niveles de la pirámide de automatización.

Este incremento de la complejidad esta basada en el hecho de tener cambios constantes y drásticos, en algunos momentos, en lo que se refiere a la posibilidad de planificar ante la aparición de eventos inesperados, cambios impredecibles en los proveedores y consumidores, posicionamiento en el mercado ante altos niveles de competitividad y, en general, cambios en los paradigmas de producción. El concepto de reconfigurabilidad surge como una nueva característica que debe tener la industria para responder adecuadamente a estos cambios.

En los enfoques clásicos de automatización se han propuesto y desarrollado los sistemas MES (Manufacturing Execution Systems), como la vía para proveer de información a todos los niveles de automatización, permitiendo el enlace entre los niveles de gerencia y de campo (planta). Sin embargo, en vez de proveer de un medio de gestión de información, los MES se han transformado en una extensión del nivel de planificación (Enterprise Resources Planning ERP), siendo sistemas mas bien rígidos y enmarcados en arquitecturas centralizadas, siguiendo la filosofía de sistemas reactivos.

Para desarrollar el concepto de reconfigurabilidad en la industria, es necesario garantizar, entonces, la comunicación entre los diferentes niveles de automatización, tanto en la línea vertical como en la horizontal, lo que permite el intercambio oportuno de la información entre dichos niveles. Esta necesidad de flexibilidad en la comunicación hace que los agentes sean los entes capaces de satisfacer este requerimiento, principalmente debido a la posibilidad de:

- Reducción de la carga de la red industrial, puesto que cada agente evalúa los datos localmente, sin necesidad de estar conectado con los sistemas ERP.
- Conocimiento de la condición actual de la planta por todos los agentes, incluyéndola en sus decisiones.
- Cambio en la características de un agente ó grupo de agentes determinado, en caso de reconfiguración del sistema.

Existe gran cantidad de propuestas de arquitecturas de automatización y control basadas en agentes [9]. En esta sección se presentan algunas de las disponibles en la literatura, a fin de dar al lector una idea de las mismas.

10.2. Arquitectura para control y planificación basado en agentes

En [14] se propone una arquitectura basada en sistemas multiagentes que permite la toma de decisiones aprovechando las capacidades de negociación de los agentes, para responder en forma dinámica a los cambios en las diferentes actividades de manufactura. El sistema propuesto tiene la capacidad de monitorear todas las actividades del proceso de producción, ejecutar simulaciones en tiempo real, analizar y planificar las actividades de producción y mantener un sistema de alarmas siempre activo. Las principales funciones de este SMA son:

- Generación, basada en tiempo, del programa de operaciones y producción.
- Monitoreo y control activo de la ejecución de las actividades del piso de planta, incluyendo el control del progreso de las operaciones para alcanzar la programación planificada, incorporando capacidades de detección de condiciones anormales, para luego identificar alternativas de operación factibles.
- Evaluación en tiempo real de las actividades de producción, tanto en efectividad como en desempeño.

Para alcanzar estas funcionalidades, los autores en [14] proponen un marco de referencia con tres módulos, descritos a continuación (ver Figura 10.1):

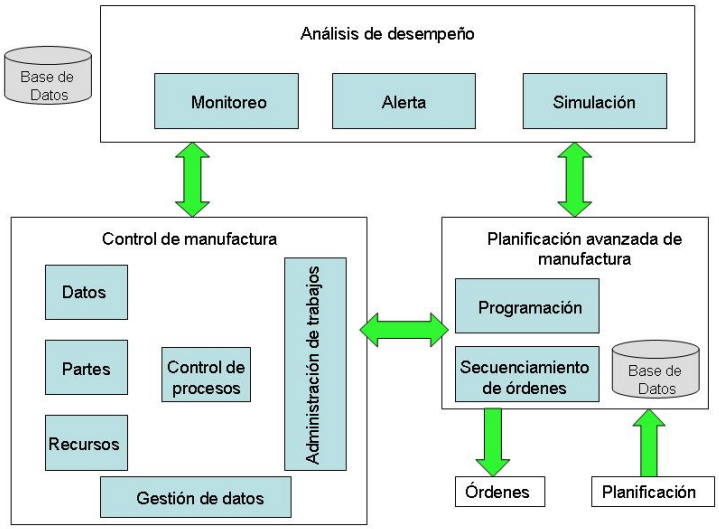

Figura 10.1: Marco de Referencia Modular

1. **Módulo de planificación avanzada de manufactura**: debe generar la programación de operaciones basada en la información de la demanda que proviene de la programación de la producción maestra y de la información de la producción activa, y de los eventos provenientes de los otros módulos.
2. **Módulo de control de manufactura**: debe registrar y controlar de modo efectivo la ejecución del sistema de manufactura a través de la realimentación efectiva del estado de la producción.
3. **Módulo de análisis de desempeño**: ejecuta un mecanismo de monitoreo basado en eventos que evalúa el desempeño de la producción y de la logística del sistema de manufactura. Debe detectar la aparición de eventos anormales, clasificar las causas e identificar las alternativas de operación posibles, tomando en cuenta la efectividad de las nuevas alternativas a través de simulaciones.

10.2.1. Arquitectura de agentes

Cada uno de los módulos de la sección anterior están conformados por un conjunto de agentes que deben implementar las funcionalidades de dichos módulos. Para la definición de los agentes, los autores se basan en una clasificación de tres tipos de agentes:

1. Agentes de ejecución, los cuales llevan a cabo procedimientos y tomas de decisiones.
2. Agentes de información, los que proveen de información y datos a otros agentes.
3. Agentes móviles, capaces de ejecutarse, moverse y comunicarse libremente en una red.

Así pues, se definen los siguientes agentes de ejecución:

- *Agente Secuenciamiento de Órdenes*: genera las órdenes demandadas.
- *Agente Programación*: genera las programación de producción en base a las órdenes demandadas.
- *Agente Administración de Trabajo*: actualiza las órdenes de manufactura en función de la programación de producción y el progreso y actualización de la información de producción (cantidad y tiempo de finalización).
- *Agente Control de Proceso*: provee la ruta de manufactura y las instrucciones de producción a cada agente móvil. Con el fin de controlar el progreso de la producción, este agente monitorea las operaciones del piso de planta y obtiene la información de la producción de cada agente móvil. También es responsable de administrar los procesos de oferta ó subasta de recursos de manufactura.
- *Agente Monitoreo de Eventos*: monitorea las actividades de manufactura asociadas a cada órden de manufactura.
- *Agente Alerta de Eventos*: debe enviar mensaje de atención ó peligro para alertar a las operaciones de piso de planta o a los agentes de programación de la ocurrencia de un evento anormal, para su corrección.

Los agentes de información definidos son:

- *Agente Middleware*: representa un software de gestión de servicios para leer y escribir datos en las etiquetas (tag) asociadas al proceso de manufactura.
- *Agente Datos*: cada uno de estos agentes debe coleccionar y proveer datos desde y para los agentes móviles y de recursos a través del agente middleware, o desde las bases de datos.

Los agentes móviles definidos son:

- *Agente Componente*: representa un componente asociado a una etiqueta (tag) del proceso de manufactura, y puede desempeñar la actividad de

manufactura del componente de acuerdo a la programación de operaciones e instrucciones de producción.

- *Agente Recursos*: representa un recurso de manufactura asociado a una etiqueta (tag) de proceso de manufactura, y debe proveer a tiempo la información de producción asociada a los recursos de manufactura.

La Figura 10.2 muestra la composición de los módulos con las diferentes clases de agentes. Por ejemplo, siguiendo el flujo de comunicación, el proceso de comunicación entre los agentes ocurre de la siguiente manera [14]:

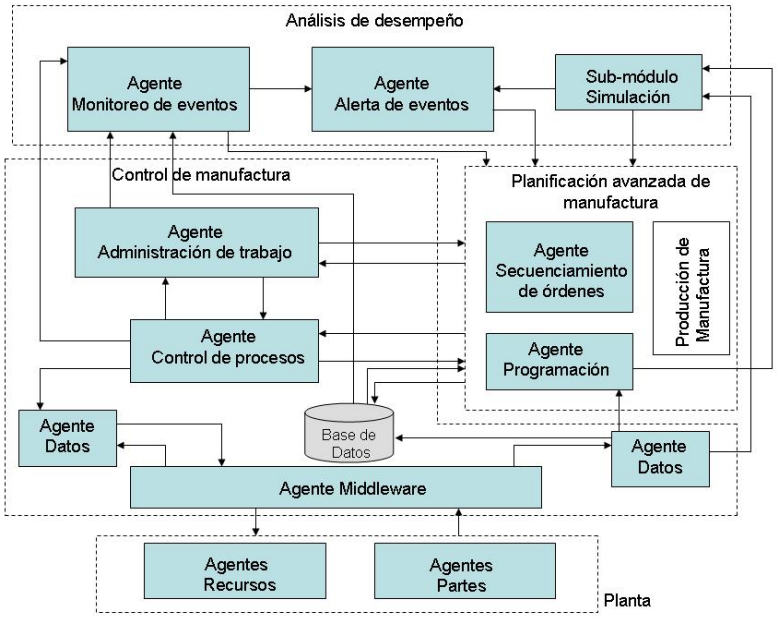

Figura 10.2: Arquitectura de Agentes

Un agente Programación, del módulo de planificación avanzada de manufactura, genera un programa de producción basada en la prioridad de la demanda y la información de la producción que proviene de los agentes de secuenciamiento de órdenes y de administración de trabajo. Además, el agente Programación genera la programación de operaciones. La decisión de asignar el recurso de manufactura apropiado para cada tarea de operación se obtiene del proceso de subasta entre el agente Control de Proceso y el agente Recursos. En el nivel de ejecución del piso de planta, un agente Componente asociado a una etiqueta del proceso de manufactura procesará la tarea de operación basada en la instrucción de operación que provee

el agente Control de Proceso. El agente Componente envía la información de producción al agente Control de Proceso a través del agente Middleware y el agente Datos para chequear si es necesario continuar su próxima operación o terminar la correspondiente orden de manufactura, mientras que el agente Componente completa la tarea de operación. Finalmente, el agente Control de Procesos envía la información de producción al agente Administración de Trabajo para verificar si la orden de manufactura ha sido completada ó no.

Cuando un agente Componente anuncia una situación anormal, el agente Programación podría recibir este mensaje y, a través de un proceso de oferta ó subasta, el agente Control de Procesos puede seleccionar otro recurso de manufactura ó el agente Programación podría generar una nueva programación de operaciones y enviarlas al agente Componente para que continúe con su operación. El agente Monitoreo puede monitorear cada orden de manufactura, clasificar la causa, y notificar al agente Alerta de Eventos para que envíe un mensaje de peligro al resto de los agentes relacionados.

Esta arquitectura de agentes enmarca a un sistema manejado por eventos que permite responder de manera dinámica a los cambios en los eventos de negocio, y manejar las excepciones. La arquitectura permite realimentar el estado de la producción y la programación de operaciones desde los niveles de planificación, logrando evaluar la efectividad de la producción y la programación de operaciones, el monitoreo y evaluación del desempeño del piso de planta, usando la información de manufactura en tiempo real.

10.3. Arquitectura para control de sistemas de manufactura flexible

En [13] se presenta un sencillo diseño de un sistema de control de manufactura flexible basado en agentes. Básicamente, se propone un conjunto de agentes asociados a los componentes fundamentales de manufactura (ver la Figura 10.3):

- *Agente Supervisor*: conoce los mecanismos de decisión del sistema. Decide cuales partes deben ser procesadas y cuales agentes deberían usarse en su procesamiento.
- *Agente Parte*: es una abstracción de la partes que son manipuladas en el proceso de manufactura.
- *Agente Vehículo*: maneja los vehículos responsables de transportar las partes entre las diferentes estaciones de trabajo. Este agente es responsable de difundir un requerimiento de entrega a todos los vehículos, seleccionar un vehículo, o transmitir la información al resto de los agentes.
- *Agente Máquina*: maneja las máquinas del sistema del manufactura, decide cuál máquina activar según la tarea, entre otras cosas. Debe chequear los sensores para la verificación de errores, reportar el estado de

las máquinas al agente Supervisor y ejecutar acciones ordenadas por el supervisor.

- *Agente Robot*: maneja los robots y realiza la asignación de tareas. Al igual que el agente Máquina, decide cuál robot activar según la tarea. Debe chequear los sensores para la verificación de errores, reportar el estado de los robots al agente Supervisor, y ejecutar acciones ordenadas por el supervisor.
- *Agente Dispositivo*: maneja las comunicaciones, tiene un rol de mediador.

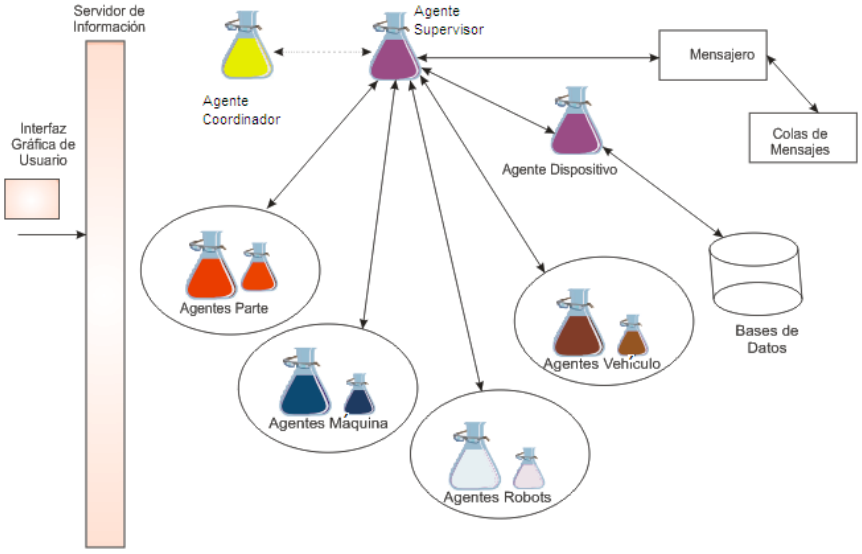

Figura 10.3: Arquitectura de control de manufactura basada en agentes

En modo general, el flujo de comunicación en el SMA ocurre de la siguiente manera:

Cuando se recibe una orden, el agente Supervisión recibe un mensaje de inicio y chequea las tareas que pueden ser ejecutadas por otros agentes. Si la tarea queda en la cola de espera, el agente Supervisor puede ejecutar una tarea previa y envía un anuncio a todos los agentes relacionados (agentes Parte, Máquina, Robot, Vehículo). Cada uno de estos agentes relacionados son responsables de preparar y enviar subastas a la cola de mensajes del agente Supervisor, quien posteriormente decide aceptar o rechazar las ofertas. Si una oferta es aceptada, la tarea es enviada a la cola de tareas del agente ganador y el agente Supervisor debe recibir mensajes de realimentación sobre el estado de la tarea. Este mecanismo de comunicación hace que

el mecanismo de subasta sea un aspecto fundamental en el desempeño del SMA.

En esta arquitectura, los agentes del sistema deben ejecutar las instrucciones que provienen del agente Supervisor, y están enfocados fundamentalmente a la parte operacional del proceso de manufactura.

10.4. Arquitectura para la integración de operaciones de manufactura

En [2] se presentan un enfoque basado en SMA para la integración de operaciones de manufactura. El enfoque apunta a optimizar los recursos dentro de las estructuras de manufactura y restricciones existentes, a la vez que considera la reconfiguración del sistema para responder a los cambios en la demanda. Se define una plataforma de decisión integrada donde concurren las decisiones de control y planificación conjuntamente con las decisiones de reconfiguración. El sistema propuesto permite modelar sistemas complejos heterogéneos, su estructura y restricciones físicas. Se propone una arquitectura donde los sub-sistemas y recursos en el sistema de manufactura interactúan con los componentes en las órdenes de producto. La plataforma permite, también, generar simulaciones de las diferentes opciones de reconfiguración, así como implantar acciones de control en dispositivos de hardware como PLC (Controladores Lógicos Programables) asociadas con las opciones de reconfiguración aceptadas.

La lógica de funcionamiento del sistema propuesto se centra en detectar cuándo la estructura actual del sistema de manufactura no puede responder a las demandas; luego, el modelo debe ser capaz de identificar las opciones de reconfiguración de productos a través de la relajación gradual de las restricciones actuales. La arquitectura de la plataforma de decisión se presenta en la Figura 10.4, en la cual se identifican tres capas:

1. **Capa de modelado y planificación**: en la cual se modelan el sistema de manufactura y las órdenes de producto; se optimiza el control, la planificación y la reconfiguración del sistema de manufactura. Por ejemplo, uno de los problemas clásicos de optimización en esta capa es la asignación de un trabajo que considera cierta cantidad de operaciones que tienen que ser completadas con ciertos recursos, en una secuencia dada, con un mínimo costo.
2. **Capa de flujo de proceso**: describe los enfoques que permiten identificar las opciones de reconfiguración, a partir de los resultados de planificación.
3. **Capa de simulación**: aloja un ambiente de simulación integrada de sistemas a eventos discretos, donde los modelos de simulación son generados automáticamente.

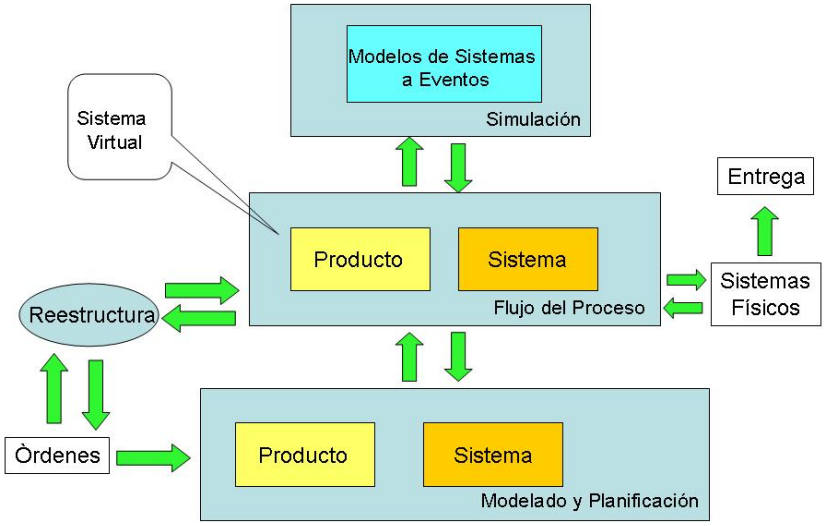

Figura 10.4: Arquitectura de la plataforma

Para cada una de estas capas ha sido propuesta una arquitectura de agentes, responsables de llevar a cabo las funcionalidades de cada capa.

10.4.1. *Arquitectura de agentes de la capa de modelado y planificación*

Cada nodo del árbol de jerarquía del sistema de manufactura es modelado como un agente, lo cual genera una organización de agentes padres y agentes hijos, como se muestra en la Figura 10.5.

De esta manera, para el modelado del producto se tienen los siguientes agentes:

- *Agentes Producto*: representa el producto final del proceso de manufactura, y tiene registrado un conjunto de agentes Componente como hijos.
- *Agentes Componente*: representan los componentes del producto y tiene registrados como hijos a cierta cantidad de agentes Componentes Básicos,
- *Agentes Componentes Básicos*: representa al componente de más bajo nivel en la estructura del producto.

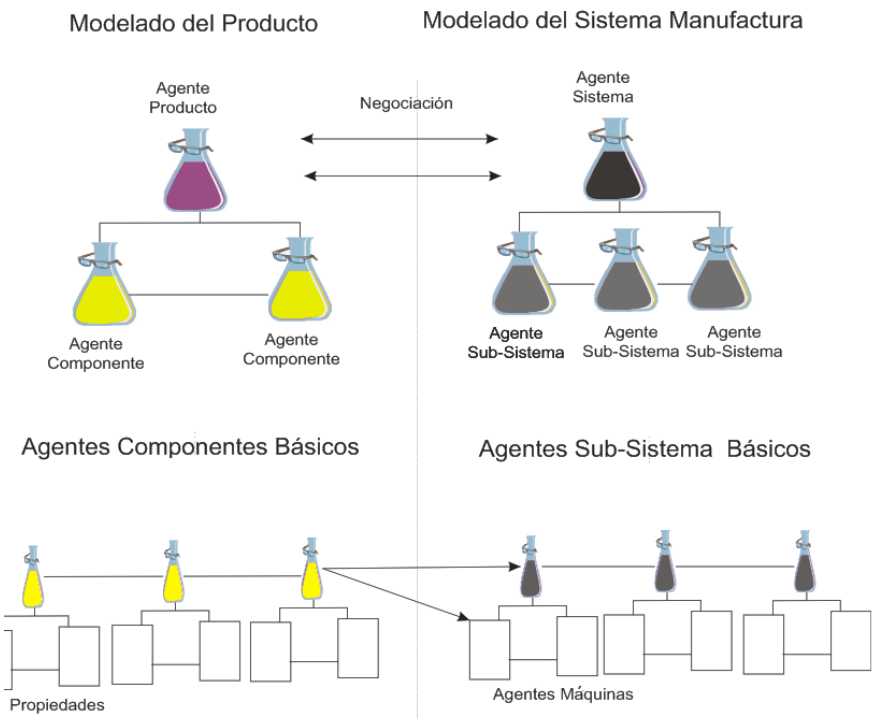

Figura 10.5: Arquitectura de la red de agentes

Para el caso del modelado del sistema de manufactura, se tienen los siguientes agentes, también organizados jerárquicamente:

- *Agente Sistema*: representa el sistema completo (fábrica), y tiene ciertos agentes Subsistema (piso de planta) registrados como hijos.
- *Agentes Subsistemas*: representa los diferentes pisos de planta de toda una fábrica, y tiene ciertos agentes Subsistemas Básicos (celda) registrados como hijos.
- *Agentes Subsistemas Básicos*: representan las diferentes celdas ó estaciones de trabajo del piso de planta, y tiene ciertos agentes Máquina registrados como hijos.
- *Agentes Máquina*: representan las máquinas que realizan los trabajos de manufactura.

Para cada agente del sistema de manufactura, los objetivos son logrados con mecanismos de subasta para formar parte del procesamiento de las órdenes de producto; mientras que para los agentes asociados al producto existe una relación de precedencia en términos de las operaciones que deben

ser completadas en diferentes niveles asociados a una secuencia determinada.

La arquitectura general de cada agente se muestra en la Figura 10.6, en donde se define una *unidad de control* y un *ambiente común*. La unidad de control contiene los mecanismos de razonamiento, inferencia y control de los agentes, las bases de conocimiento, así como los protocolos y mecanismo de subasta. El ambiente común contiene las bases de datos con la información de los agentes y los hijos registrados; contiene también el facilitador de comunicaciones, que es básicamente una pizarra, para facilitar la comunicación entre los agents padres y los agentes hijos. Se definen los siguientes agentes:

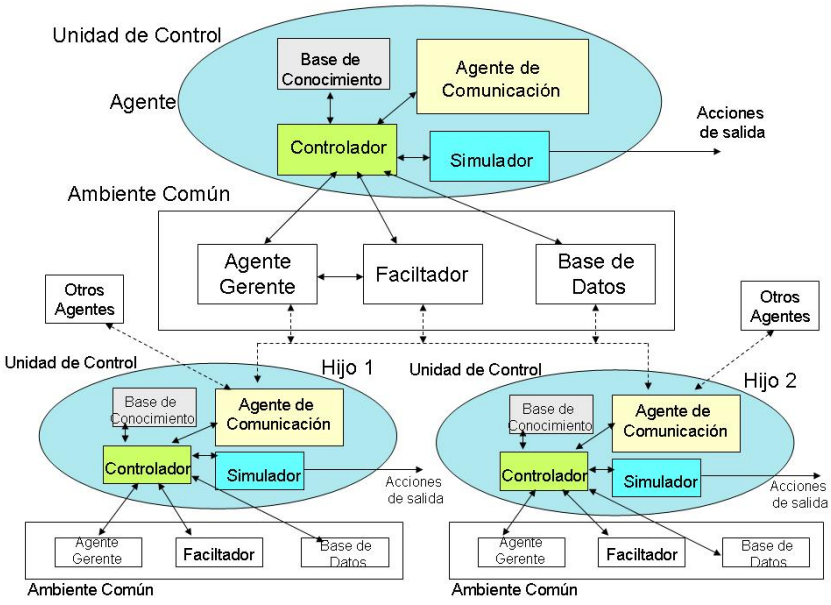

Figura 10.6: Arquitectura de automatización de manufactura basada en agentes

1. *Agente de Comunicación*: tiene la habilidad de comunicarse con otros agentes excepto los agentes hijos inmediatos, a través de mecanismos de subasta.
2. *Agente Gerente (Manager)*: provee los mecanismos para crear, instanciar, registrar, desregistrar y detener un agente, en virtud de que los agentes tienen la misma estructura y son creados como una copia de un prototipo.

10.4.2. Agente de la capa de flujo del proceso

En esta capa se define un solo agente, el *Agente Estructura*. Este agente mantiene un registro de las órdenes de producto procesadas por el sistema en un periodo de tiempo, con las rutas de procesamiento óptimo logradas mediante mecanismos de subasta. Este agente analiza todas las órdenes para un período de tiempo, y encuentra la proporción de componentes de producto que han usado configuraciones alternativas.

El agente Estructura debe proveer, a la capa de simulación, el modelo de flujo de trabajo y actividades, para poder simular las diferentes alternativas de configuración posibles.

El principal aporte de este trabajo es proveer un modelo de agentes que permite a las organizaciones de manufactura mejorar sus niveles de competitividad, integrando y coordinando sus operaciones, La arquitectura de SMA facilita la implementación y ejecución de un proceso jerárquico de subasta controlado óptimamente, logrando un método para identificar, simular y evaluar alternativas de reconfiguración del sistema.

10.5. La arquitectura PABADIS

La arquitectura PABADIS (*Plant Automation based on Distributed Systems*), ha sido propuesta en el marco del proyecto *PABADIS-PROMISE* [8]. Esta arquitectura permite desarrollar sistemas MES distribuidos basados en la tecnología de agentes con el fin de enfrentar las limitaciones de los MES clásicos: complejidad, flexibilidad y robustez [4].

Esta arquitectura está basada en la automatización piramidal, pero utilizando sistemas distribuidos y agentes inteligentes, se reduce la jerarquía a dos capas, disuelve la capa de supervisión y divide su funcionalidad en un parte centralizada que puede ser ubicada dentro del sistema de planificación y una parte descentralizada que puede ser implementada por agentes móviles, ver la Figura 10.7.

Así, PABADIS es una arquitectura que refuerza la descentralización de los sistemas de automatización, resaltando las siguientes características [4, 7]:

1. La programación a nivel de planta se reduce a la ejecución del proceso de producción, a través de órdenes de generación de nuevos agentes.
2. No hay necesidad de una comunicación exhaustiva con los sistemas MES y ERP.
3. Las funcionalidades del MES, implementadas con agentes, pueden cambiarse con solo ajustar el código de los agentes.
4. La integración del piso de planta con los sistemas ERP se realiza a través de un supervisor.

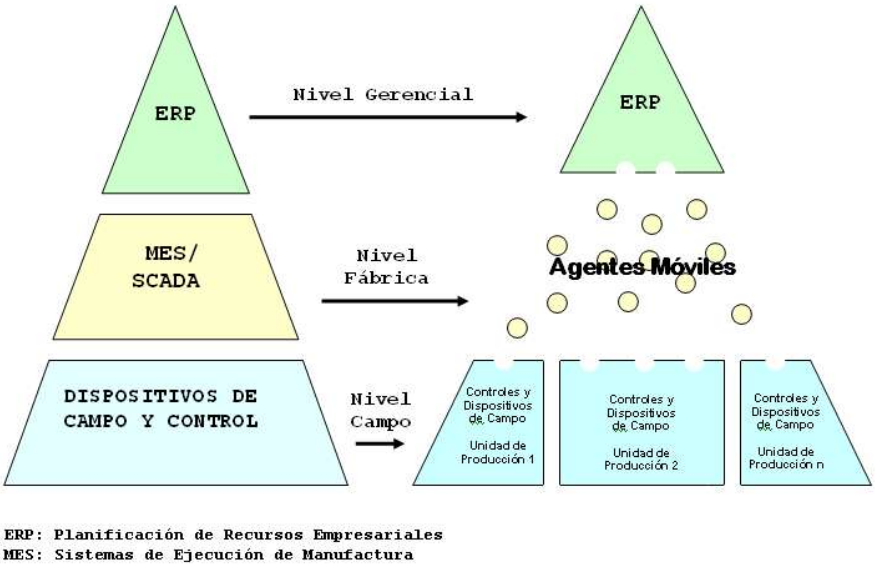

Figura 10.7: PABADIS

Estas características permiten que los procesos de toma de decisiones bajen a las capas intermedias de la pirámide de automatización, disminuyendo los tiempos de reacción debido a los eventos ocurridos en el nivel de campo. De esta manera, los sistemas ERP son los responsables de la planificación estratégica y los sistemas MES son responsables de la planificación táctica, realizadas por unidades distribuidas y cooperativas, lo que se traduce en incorporar beneficios de los sistemas proactivos [8].

En general, el enfoque PABADIS propone disolver el MES en una parte centralizada adjunta al sistema ERP, y otra parte descentralizada que es implementada por agentes. Cada agente esta asociado a una pieza de trabajo que tiene la información necesaria del producto, y se mueve a través de la planta de la misma forma como lo hace la pieza física. Esta información puede ser relativa a:

- Secuencia y programación de producción.
- Datos de producción de la máquinas asociadas a las piezas.
- Estado del procesamiento.
- Información administrativa sobre la orden.
- Uso de los recursos.
- Información del control de calidad.

La arquitectura PABADIS esta descrita en dos niveles: *un nivel alto*, conceptual, basado en roles y sub-roles que encapsulan las funcionalidades requeridas para el control de la producción, y *una descripción a bajo nivel*,

orientada a la implementación, donde los roles son asignados a agentes que implementan finalmente al sistema de ejecución de manufactura MES. A continuación se describen tanto el modelo de roles, como el modelo de agentes.

10.5.1. Modelo de Roles

La arquitectura PABADIS esta centrada en la definición de roles que luego son instanciados a través de agentes. Se definen ocho roles que describen las funcionalidades del sistema, los cuales se ilustran en la Figura 10.8 y se describen a continuación:

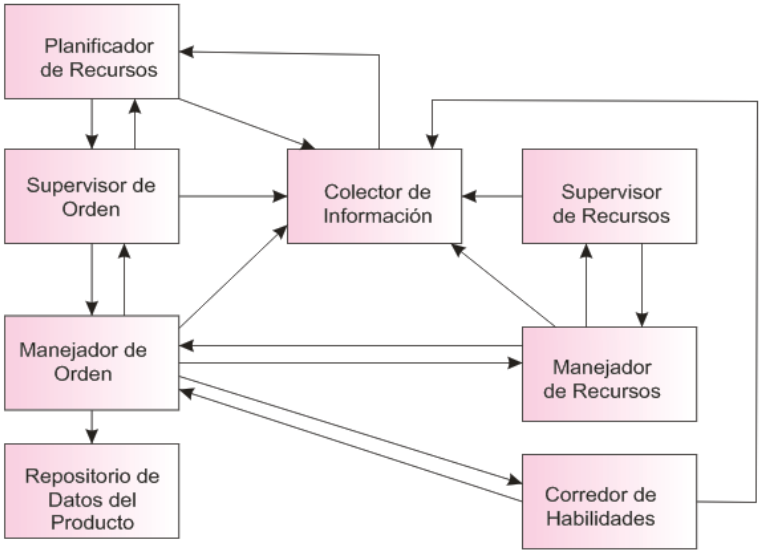

Figura 10.8: Modelo de roles de la arquitectura PABADIS

1. **Manejador de Orden.** Es responsable del control de ejecución de órdenes de trabajo. Abarca todo tipo de órdenes, de diferente naturaleza, tales como órdenes de manufactura, órdenes de mantenimiento y órdenes de información. De esta manera debe controlar:

   - La colección y adquisición de información de los procesos de manufactura, de las órdenes y de los productos.

- La selección de los recursos para el proceso de ejecución de la manufactura.
- La localización de los recursos y programación.
- El diseño de las aplicaciones de control de campo.
- El proceso de manufactura a alto nivel.
- La creación y supervisión de sub-órdenes.

Un Manejador de Ordenes esta habilitado para crear, inicializar y supervisar otros manejadores de órdenes para partes de una orden, por tanto, un manejador de órdenes debe coordinar estos otros manejadores de órdenes. Debido a la complejidad de este rol, se definen otros sub-roles tales como:

- *Programador de órdenes.* Es responsable del control de la ejecución de una orden a alto nivel.
- *Controlador del proceso.* Es responsables del control del proceso de manufactura. Controla, inicia y monitorea el procesamiento físico de un recurso, tal y como lo hacen los sistemas MES.
- *Negociador del proceso.* Es responsable del control de la ejecución del proceso de manufactura.
- *Diseñador de las aplicaciones de control.* Es responsable del diseño de las aplicaciones de control requeridas para la ejecución de una orden.

2. **Manejador de Recursos.** Es responsable del manejo de los recursos dedicados. Incluye los recursos humanos, recursos de transporte, recursos de material, entre otros. Debe ser capaz de manejar:

   - Las habilidades de los recursos.
   - Recursos de control físico que aseguren el proceso de manufactura.
   - Programación de recursos.

   Un Manejador de Recursos se asigna a un solo recurso, y se inicia con un evento de inicio y termina con un evento de fin. Para el Manejador de Recursos, también se definen otros sub-roles:

   - *Manejador de Habilidades.* Es responsable del manejo de la información asociada a las habilidades del recurso y monitoreo de eventos internos del recurso.
   - *Manejador de Negociaciones.* Es responsable de la programación y localización del recurso, incluyendo la negociación entre Manejadores de Ordenes y Manejador de Recursos.
   - *Controlador de Recursos.* Maneja el control de los recursos en el nivel de control de campo, esto es, recurso de control físico e interfaz hacia los sistemas de control supervisorio.

3. **Supervisor de Orden.** Es responsable de la creación, inicialización y control supervisorio de los manejadores de órdenes. Un Supervisor de Orden

recibe una inicialización y una orden de cambio de información desde los ERP y los transforma en información del Manejador de Orden. Un Supervisor de Orden esta asociado a un planificador de recursos empresariales.
4. **Planificador de Recursos Empresariales.** Es responsable de manejo de los datos de una orden a alto nivel. Maneja la integración de parte de órdenes del sistema ERP. Un Planificador de Recursos Empresariales puede estar asociado a varios Supervisores de Orden.
5. **Corredor de Habilidades** Es un directorio de servicios de todas las habilidades ofrecidas por los recursos. Debe ser capaz de coleccionar todas las habilidades de los recursos registradas por un Manejador de Recursos, evaluar los cambios en las habilidades, informar sobre los cambios relevantes de habilidades.
6. **Supervisor de Recursos.** Maneja la integración de los sistemas legados en los niveles de control y de los MES, así como la integración con sistemas externos como los SCADA's, las Interfaz Hombre-Máquina, o la intervención humana para la programación y localización de recursos relacionados con mantenimiento o recursos humanos.
7. **Colector de Información.** Es un receptor genérico de información responsable de la colección de datos relacionados con órdenes y recursos. Maneja la integración de información entre los ERP, SCADA's, históricos, entre otros.
8. **Repositorio de Datos del Producto.** Es responsable del manejo de toda la información del producto, esto es, la descripción general del producto, proceso de manufactura y sus características, aplicaciones de control, entre otros.

Una vez definidos los roles, la arquitectura se basa en la descripción de actividades y protocolos que especifican el comportamiento de dichos roles. Así, las actividades describen el comportamiento interno de un rol, según los eventos entrantes. Los protocolos definen la interacción entre los diferentes roles y se agrupan en cinco tipos:

- Negociación y localización de recursos.
- Ejecución de pasos de procesamiento.
- Manejo de datos de orden.
- Monitoreo de disponibilidad de recursos.
- Requerimiento de información general.

El modelo de roles de la Figura 10.8 describe la arquitectura general de PABADIS, la cual se concentra en las funcionalidades y responsabilidades requeridas en los procesos de manufactura. La implementación de los roles se realiza a través de su instanciación en entidades específicas (agentes), según los requerimientos del sistema con respecto a su producción, recursos y productos.

10.5.2. Modelo de agentes

No todos los roles definidos en la sección anterior se implementan como agentes. Sin embargo, cualquiera sea la característica del proceso de producción, se definen en PABADIS dos entidades implementadas por agentes:

1. **Agente Orden.** Es responsable de la manufactura de un producto. El Agente Orden instancia el rol de un Manejador de Orden y, en consecuencia, los sub-roles para él definidos. En este sentido, es responsable de la ejecución y control de una orden de producción en el nivel del MES de la pirámide de automatización. Este agente deliberativo es creado e inicializado por el Supervisor de Orden y usa las habilidades ofrecidas por el Agente Recurso, a través del Corredor de Habilidades. El Agente Orden ofrece el reporte de producción a su supervisor (Supervisor del Agente Orden). El Agente Orden también puede implementar el rol de Repositorio de Datos del Producto. En algunos casos, los subroles pueden implementarse en diferentes tipo de Agentes Orden, teniendo especial atención en la separación de la fase de planificación (Programador de Orden y Negociador del Proceso) y la fase de ejecución (Controlador del Proceso y Diseñador de Aplicaciones de Control).
2. **Agente Recurso.** Representa un recurso del sistema de agentes. El Agente Recurso instancia el rol del un Manejador de Recursos y los sub-roles a éste asociados. Así, el Agente Recurso maneja los recursos físicos, lógicos, humanos, entre otros, ofreciendo sus habilidades a la comunidad de Agentes Orden. El Agente Recurso es responsables de la ejecución y control de órdenes a nivel de control de campo, usa los dispositivos de control y actividades de los recursos controlados, y anuncia sus capacidades al Corredor de Habilidades. El Agente Recurso en una entidad permanente y reactiva que refleja la situación de la planta, independientemente de una orden de producción. Adicionalmente, un Agente Recurso puede implementar el rol de un Corredor de Habilidades distribuido, ofreciendo sus habilidades a la comunidad de agentes.
3. **Agentes Especializados.** Los otros roles del modelo no son específicamente implementados por agentes pero algunos están en estrecha relación con el SMA, lo que significa que pueden implementar parte de su funcionalidad como agentes. De esta manera, se definen los siguientes agentes:

   - El **Agente Supervisor de Agente Orden**, instancia el rol del Supervisor de Orden. En general, se considera una sola instancia de este agente puesto que la implementación de la interfaz con el nivel ERP requiere un solo punto de acceso con el nivel MES.
   - El **Agente Supervisor de Agente Recurso**, el cual provee de puntos de acceso adicionales para la integración con sistemas externos para introducir cambios en la programación de los recursos, cambios en el estado del recurso, cambios en la aplicaciones de control, entre otros.

- El **Agente Corredor de Habilidades**, el cual puede ser un agente centralizado ó distribuido.

Otros roles del modelo son considerados como componentes de soporte o de interfaz, los cuales no son implementados como agentes [8]. En la Figura 10.9 se muestra el modelo de agentes y sus relaciones.

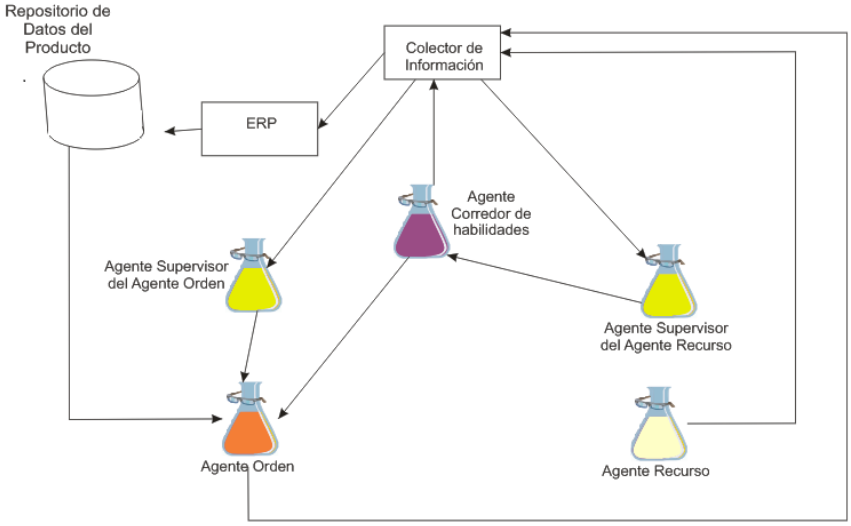

Figura 10.9: Modelo de agentes de la arquitectura PABADIS

10.6. La arquitectura MetaMorph

MetaMorph es una arquitectura adaptativa basada en agentes, propuesta con el fin de abordar el problema de adaptación de los sistemas principalmente de manufactura, en el marco de las necesidades de flexibilidad y reconfigurabilidad de los sistemas de producción [12]. Su nombre deriva precisamente de la característica principal de tener forma, estructura y actividades cambiantes que se adaptan dinámicamente a los cambios del ambiente y tareas que surgen de forma emergente. La arquitectura esta concebida en cuatro niveles:

- *Empresa virtual:* Trata los aspectos asociados a las alianzas estratégicas para la unificación de subsistemas de manufactura heterogénea en una coalición virtual de subsistemas cooperativos.

- *Sistemas inteligentes distribuidos:* Se refiere a los sistemas que son requeridos para permitir la autonomía de cada miembro de la red empresarial de manufactura
- *Ingeniería concurrente:* Se refiere a mecanismos para el manejo de la información para el diseño de procesos y productos, cumpliendo con tiempos oportunos.
- *Arquitectura de agentes:* Definida para poder construir y operar los subsistemas de manufactura.

MetaMorph se basa en una arquitectura federada centrada en mediación, la cual es un tipo particular de organización donde los agentes inteligentes pueden enlazarse con agentes llamados Agentes Mediadores, para encontrar otros agentes en el sistema. Estos agentes mediadores son coordinadores del sistema, promoviendo la cooperación y aprendizaje entre los agentes.

10.6.1. Modelo de Agentes

Se definen cuatro tipo de agentes principales en MetaMorph:

1. **Agente Recurso:** se definen para representar los dispositivos de manufactura y operación. Son autónomos y cooperativos.
2. **Agente Mediador:** coordinan las interacciones entre los agentes recurso y mediadores, inclusive. Son autónomos, cooperativos y de aprendizaje. Ejecutan los roles de coordinación, incluyendo resolución de conflictos de decisión, establecen subsistemas colaborativos (coaliciones), así como también implementar mecanismos de *brokering* entre agentes. De esta manera, los mediadores necesitan tener suficiente conocimiento organizacional para poder enlazar los requerimientos de los agentes con los recursos necesarios.
3. **Agente Coalición:** son grupos (*cluster*) de agentes que trabajan cooperativamente. En MetaMorph, el mecanismo de negociación se basa en la descomposición de tareas y en la formación de grupos de agentes. Las tareas de alto nivel se descomponen a través de actos de mediación en su correspondiente nivel de información. Cada subtarea se distribuye para determinar el mejor plan. Así, los mediadores que han aprendido de las interacciones entre los agentes, identifica la coalición que puede usarse para la búsqueda distribuida para la resolución del las tareas. De esta manera, los Agentes Coalición son en MetaMorph el primer mecanismo de resolución de problemas, dotando a la arquitectura con propiedades de escalabilidad y agregación.
4. **Agente Clonación:** son agentes clonados de los Agentes Recurso, a fin de procesar información en modo concurrente. Los agentes clones se incluyen en grupos (cluster) de coordinación virtual donde los agentes negocian para encontrar la mejor solución. El mecanismo de clonación puede

clonar agentes recurso desde computadores remotos en computadores locales donde residen los agentes mediadores, con el fin de reducir el tiempo de comunicación y de programación de tareas.

10.6.2. Modelo del Agente Mediador

El Agente Mediador es clave en la arquitectura MetaMorph, puesto que deben encapsular diferentes comportamientos de manufactura para lograr la coordinación de los agentes del sistema. Así, este agente se diseña sobre la base de la especificación de diferentes actividades asociadas a diferentes meta-niveles. De esta manera, se crean diferentes tipos de agentes mediadores para cumplir con la actividades de la fábrica.

Las actividades de meta-nivel se distribuyen en siete niveles de abstracción (dominios), que siguen patrones típicos de las diferentes areas de los procesos de manufactura. En cada nivel de abstracción se suceden las diferentes etapas de planificación, programación y control. Cada Agente Mediador puede realizar algunas o todas las actividades de cada meta-nivel. Tales niveles de abstracción son:

1. *Empresa*, contiene el conocimiento del sistema y representa las metas de la fábrica través de objetivos concretos.
2. *Diseño y especificación del producto*, abarca datos asociados a las tareas de manufactura que permite a los mediadores reconocer las tareas que deben coordinarse.
3. *Organizaciones virtuales*, su alcance llega al conocimiento detallado del comportamiento de los recursos, establece y reconoce de forma dinámica las relaciones entre recursos y agentes.
4. *Planificación y programación*, integra las restricciones tecnológicas con restricciones temporales en un modelo de procesamiento de información concurrente.
5. *Ejecución*, facilita las transiciones entre los dispositivos físicos, coordina transacciones entre dispositivos de manufactura y otros meta-niveles (dominios).
6. *Comunicación*, provee de un lenguaje de comunicación común.
7. *Aprendizaje*, comprende razonamientos e intercambio de mensaje que deben ser aprendidos y automatizados.

Teniendo en cuenta la definición de los anteriores dominios, existen dos niveles diferentes de Agentes Mediadores. Un primer nivel, mediadores de alto nivel, que se especializa en coordinación de grupos virtuales, y un segundo nivel, mediadores de bajo nivel, especializado en la coordinación de agentes dentro de grupos virtuales.

Cada empresa de manufactura necesita al menos un mediador de alto nivel, el mediador de empresa, que funge como integrador del sistema y es

capaz de reconocer a todos los mediadores de bajo nivel, las plataformas y los recursos de la empresa.

En la Figura 10.10 se representan las relaciones entre los diferentes agentes de MetaMorph.

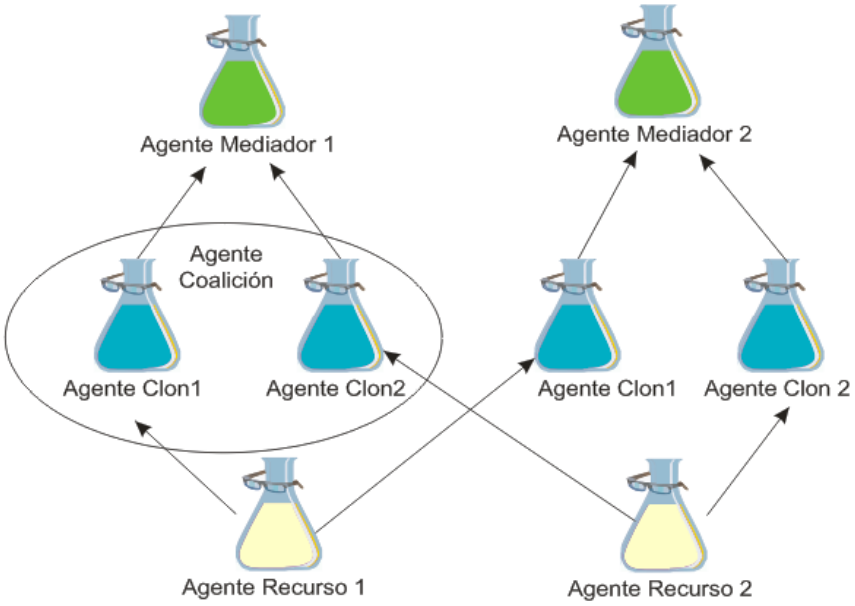

Figura 10.10: Modelo de agentes de la arquitectura MetaMorph

10.7. La Arquitectura AARIA

La arquitectura AARIA (*Autonomous Agents for Rock Island Arsenal*), es una arquitectura basada en agentes orientada al problema de programación (*scheduling*) en manufactura. Los requerimientos a partir de los cuales surge esta propuesta son aquellos referidos a la programación y control en piso de planta, de tal manera que los agentes deben responder a estos requerimientos. La propuesta se concentra mas en el por qué y no en el qué hace el proyecto ó el cómo lo hace [10].

Las tareas de programación y control en piso de planta requieren recursos que les permita producir productos de alta calidad en el momento que los clientes los requieran, por tanto debe haber un proceso de decisión para elegir cuáles máquinas ó dispositivos deben usarse en función del tipo de producto que se requiera.

En esta propuesta, los requerimientos se agrupan en dos tipos: interfaces externas y operaciones internas.

Los requerimientos asociados a las interfaces externas se refieren a:

- Soporte a la interacción cliente/proveedor.
- Soporte al personal de operación y equipos.
- Soporte a los ingenieros de manufactura y gerencia.

Los requerimientos asociados a las Operaciones Internas se refieren a:

- Cambios frecuentes en los procedimientos de manufactura.
- Funcionalidades en el área de ERP.
- Diferentes esquemas de control.
- Metamorfosis, para que los sistemas de apoyo al proceso sean capaces de representar una entidad tal y como son en el transcurso del tiempo (una entidad se inicia como una orden y termina como un producto).
- Uniformidad en la interfaz interna-externa.

10.7.1. Definición de agentes

En base a los requerimientos establecidos, esta propuesta define las siguientes clases de agentes:

1. **Proceso Unitario.** Es la abstracción de la parte del proceso de producción cuya instanciación específica en tiempo y espacio es una operación que debe realizarse. Conoce qué partes y recursos se necesitan para ejecutar un proceso que genera una parte específica del producto. Es un agente persistente.
2. **Recurso.** Se refiere a las herramientas o recursos físicos (máquinas, energía, programas, documentación, entre otros) necesarias para ejecutar una operación y se caracterizan por tener programas de mantenimiento, disponibilidad y costo de uso. Los operadores de máquinas también son considerados un recurso. Mantiene la historia y programación de los compromisos del recurso. Es un agente persistente.
3. **Director.** Se refiere al responsable humano de la operación, equivalente al director de planta.
4. **Parte.** Es la abstracción tanto de los materiales (entradas) como de los productos (salidas) de un proceso unitario. Mantiene la información sobre su tipo, su inventario, su historia de producción con el fin de difundir sus capacidades de producción futura. Es un agente persistente.
5. **Cliente.** Se refiere al beneficiario de la ejecución del trabajo (quien genera la orden de trabajo).
6. **Proveedor.** Es una abstracción de otra variedad de beneficiario (a quien el material es comprado).

Las interacciones entre los agentes persistentes son modeladas a través de otros agentes transitorios:

1. Agente Compromisos.
2. Agente Materiales.
3. Agente Productos.
4. Agente Operaciones.

Los agentes Compromisos, Materiales y Producto modelan las interacciones entre el agente Proceso Unitario y otros agentes. El agente Operaciones modela los aspectos transitorios del agente Proceso Unitario.

Los agentes transitorios tiene asociado un ciclo de vida en seis fases: En demanda, En compromiso, Comprometido, Disponible, Activo y Archivado. Por ejemplo, en la fase Archivado los agentes transitorios son almacenados en su respectivo agente persistente como parte del registro del sistema. El agente Recurso absorbe al agente Compromisos archivados, el agente Parte absorbe al agente Productos archivados y Materiales archivados. El agente Proceso Unitario mantiene su propio agente Operaciones archivadas.

Con esta clasificación de agentes, la propuesta AARIA garantiza las implicaciones derivadas de los requerimientos definidos en la sección anterior. En la Figura 10.11 se muestran las relaciones entre algunos de los agentes de esta arquitectura.

10.8. Una arquitectura para automatización industrial

En [5, 11] se presenta una propuesta de arquitectura para automatización industrial basada en agentes, que pretende dotar de inteligencia a los niveles inferiores de la pirámide de automatización. A partir de los requerimientos funcionales del proceso de automatización industrial, se propone una arquitectura jerárquica compuesta por un nivel superior, que contiene dos clases de SMA que implementan las funcionalidades de control y automatización, y un nivel inferior (middleware) el cual soporta las actividades de la capa superior y administra las comunicaciones desde y hacia los dispositivos de campo.

En esta propuesta, los diferentes niveles de la pirámide de automatización son representados como componentes (ó sub-sistemas) que son vistos como agentes y comunidades de agentes, cuyas relaciones organizacionales son implementadas a través de los mecanismos de coordinación, cooperación y negociación, a fin que la inteligencia pueda ser distribuida a través de la estructura jerárquica piramidal, como se muestra en la Figura 10.12

La arquitectura de agentes que se propone en [5, 11] toma en cuenta el alto grado de heterogeneidad de los procesos de automatización industrial que se ilustran en la Figura 10.12, tanto en los dispositivos físicos a bajo nivel

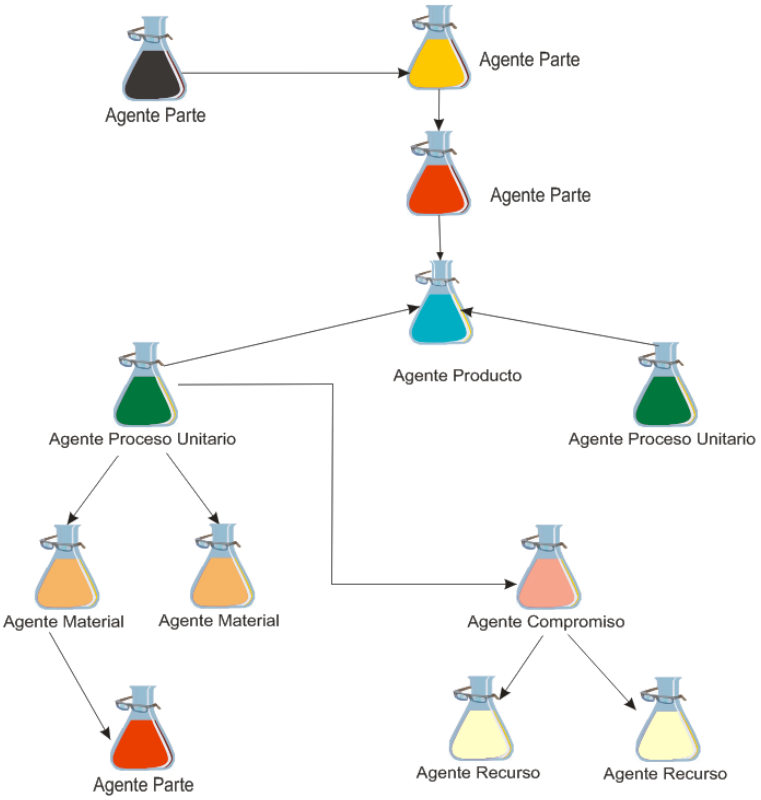

Figura 10.11: Agentes de la arquitectura AARIA

(PLC, RTU, entre otros), en los dispositivos físicos a alto nivel (computadores industriales, servidores, clusters), como en las aplicaciones que abarcan desde protocolos de comunicación y visualizadores hasta aplicaciones complejas propias del negocio.

En la Figura 10.13 se muestra la arquitectura de agentes, la cual tiene cuatro capas:

1. **Capa Superior.** Es el nivel superior. En esta capa existen dos comunidades de agentes:

   - *Agentes Proceso:* representan los dispositivos y abstracciones del proceso real. Los agentes se definen en base a la división física del proceso controlado y a la división funcional de las tareas. La representación del proceso a través de agentes permite ejecutar tareas inteligentes tales como evaluación de parámetros y variables, comparaciones de desempeño, entre otros. Esta descomposición de agentes permite soportar

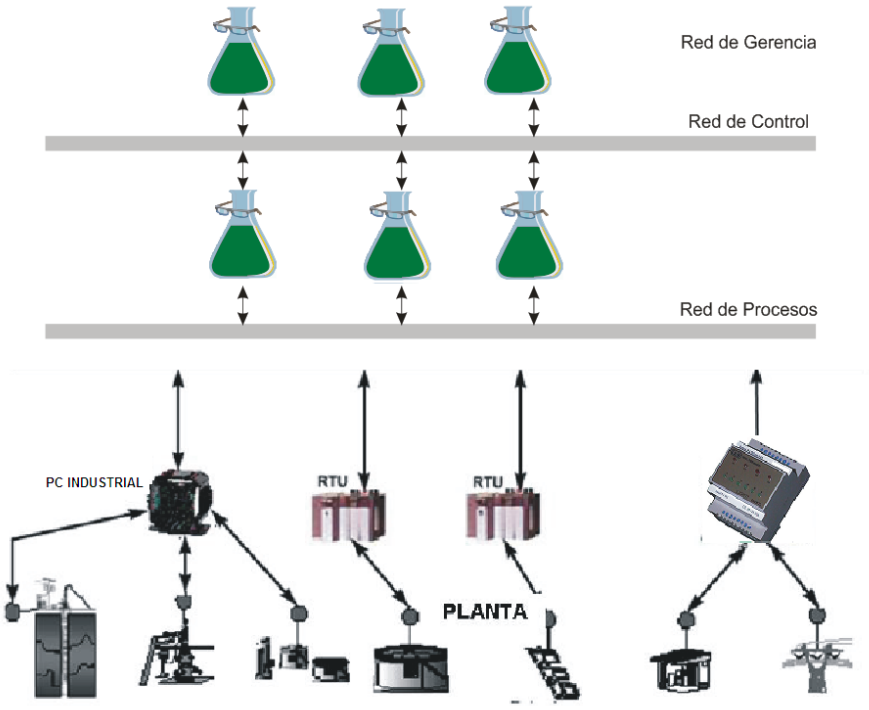

Figura 10.12: Topología de agentes para automatización

abstracción de información e intercambiar conocimiento refinado sobre el proceso, por lo que la principal función de los Agentes Proceso es mantener actualizado el conocimientos sobre el estado del área que representa, basado en la medición de las variables de su dominio.

- *Agentes Aplicación:* implementan funciones específicas de automatización industrial tales como visualización, supervisión, control, optimización, entre otras. Los agentes de aplicación puede ser agentes especializados que permiten implementar envoltorios para aplicaciones ya implementadas por otros componentes de software. De esta manera, los agentes Aplicación ofrecen servicios a los agentes Proceso.

La capa superior tiene las siguientes funciones:

- Monitoreo de variables de operación y manejo de sus iteracciones.
- Especificación e implementación de comandos de control.
- Programación de la producción.
- Manejo de los datos de producción.
- Planificación de tareas de mantenimiento.

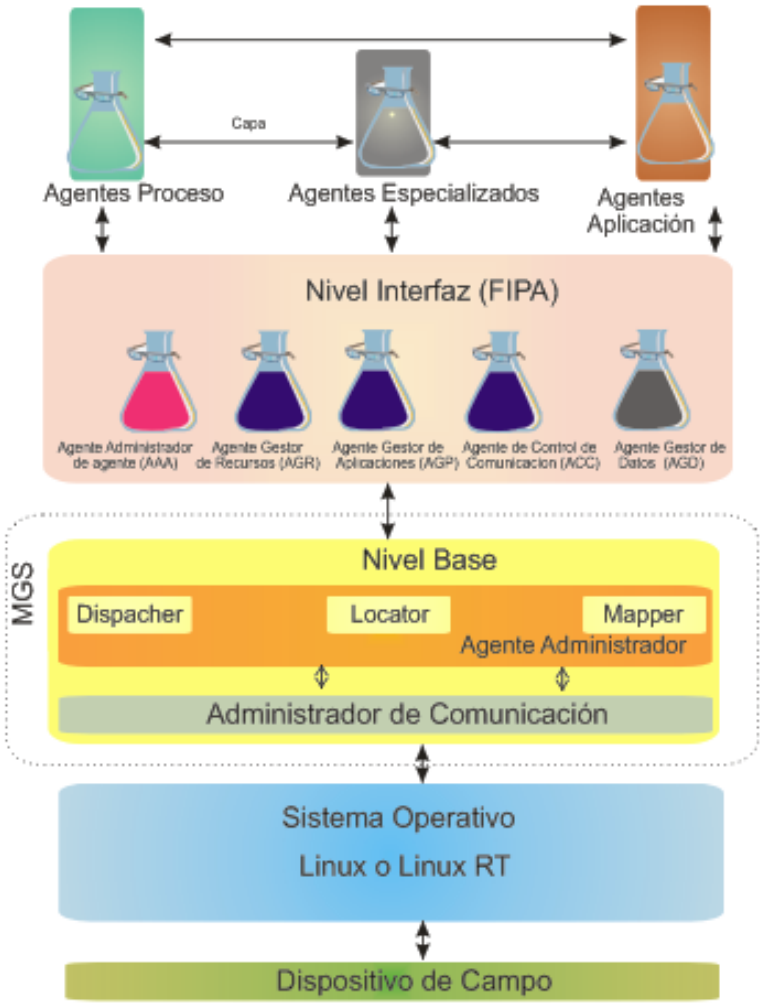

Figura 10.13: Arquitectura de agentes para automatización

- Manejar interfaces con actores externos tales como ingenieros, clientes, proveedores, entre otros.

2. **Capa de Interfaz.** Esta capa ofrece los servicios para el manejo de las operaciones y comunicaciones entre las comunidades de agentes de la capa superior. En la Sección 5.4 se detallan las características de esta capa.
3. **Capa Base.** Esta capa contiene dos subsistemas, uno que maneja las comunicaciones entre los diferentes sitios y otro que maneja los servicios

de los agentes. En la Sección 5.4 se especifican las características de esta capa.

Las capas de Interfaz, Base y Sistema Operativo constituyen el nivel inferior (Middleware), que implanta las abstracciones mínimas para especificación, instalación y manipulación de agentes y objetos, constituyendo el medio de gestión de servicios, (ver la Sección 5.4).

10.9. Sistema de Control Distribuido Inteligente basado en Agentes (SCDIA)

El modelo de *Sistemas de Control Distribuido Inteligente* (SCDIA) es un modelo de referencia basado en SMA para el desarrollo de sistemas de control distribuido inteligentes, presentado por los autores en [6]. Este SMA esta inspirado en los modelos clásicos de referencia multi-nivel, como el modelo ISO/OSI presentado en la Sección 8.4.1.1, y que se muestra en la Figura 10.14.

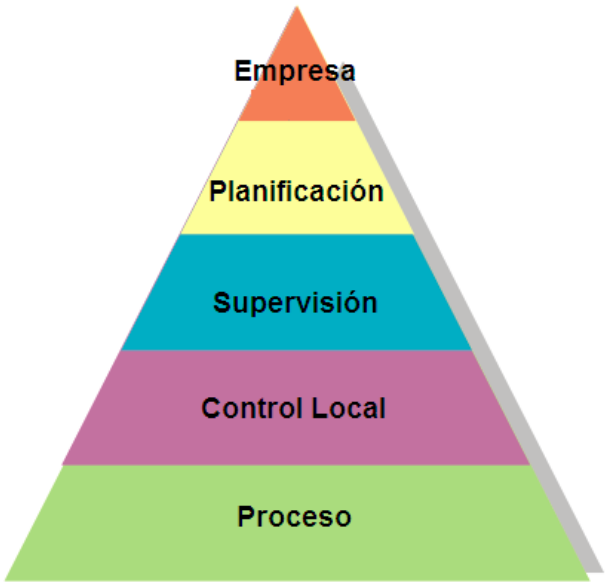

Figura 10.14: Modelo jerárquico ISO/OSI

El modelo SCDIA consiste en una colección de agentes que representan los elementos presentes en un lazo de control de procesos, con la intención de establecer un mecanismo genérico para el manejo de las actividades re-

lacionadas con la automatización industrial. En la Figura 10.15 se presenta el modelo de referencia SCDIA, el cual se enmarca en una arquitectura jerárquica que contempla una serie de agentes responsables de ejecutar las tareas propias de cada jerarquía. Dichos agentes interactúan con el fin de enviar órdenes o respuestas a niveles inferiores y/o superiores, respectivamente, permitiendo integrar los diferentes paradigmas de inteligencia. El uso de SMA permite ver al sistema de control como un conjunto de agentes que pueden ser implementados individualmente. De esta manera, el modelo SCDIA tiene características de un sistema descentralizado, emergente, autónomo y concurrente.

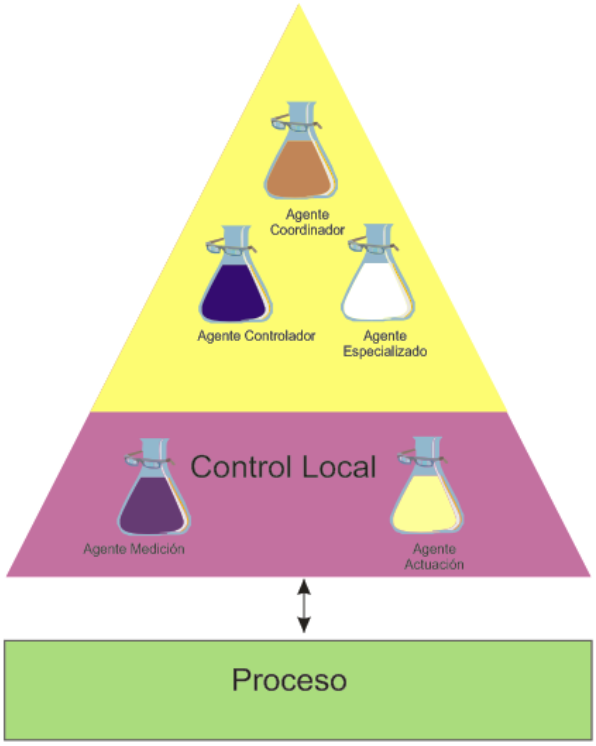

Figura 10.15: Modelo de referencia SCDIA

10.9.1. Los agentes de control del SCDIA

El modelo SCDIA esta conformado por cuatro agentes fundamentales:

1. **Agente Observación.** Recolecta la información necesaria para conocer el estado del proceso. Este agente puede recibir información tanto de elementos físicos, como los sensores del proceso, como de cualquier otra fuente lógica que provea información importante para tareas de los agentes del SCDIA.
2. **Agente Controlador.** Toma decisiones basado en la observación del estado del sistema, y las transforma en órdenes que deben ser cumplidas por otros agentes del SCDIA.
3. **Agente Actuación.** Ejecuta las decisiones tomadas por los agentes Controladores, Coordinadores y/o Especializados.
4. **Agente Coordinador.** Procesa la información recibida por el agente controlador, generando planes para la realización de las tareas para la ejecución de las órdenes del agente controlador. Puede flexibilizar y/o modificar las decisiones del agente controlador para establece nuevos objetivos y servicios globales. Dirige a los agentes presentes en su comunidad.
5. **Agente Especializado.** Ejecuta tareas particulares necesarias en la comunidad de control.

La comunidad de agentes de control descrita se distribuye, en la arquitectura de la Figura 10.15, en dos niveles: un nivel de interacción con el proceso, en donde se encuentran el agente de observación y de actuación y se corresponde con el nivel Proceso del modelo ISO/OSI; y un nivel de decisión en donde se encuentran los demás agentes de la comunidad, y se corresponde con el nivel Control del ya mencionado modelo ISO/OSI.

En general, el SCDIA es visto como una red de agentes autónomos, con diferentes responsabilidades según el nivel al que pertenecen. Los agentes están distribuidos a través de la jerarquía de control y dispersos geográficamente. Cada agente funciona en paralelo, pero comparten e intercambian información para cooperar en la realización de las tareas. La interacción dinámica de los múltiples agentes sucede no sólo entre los niveles, sino en el interior de los niveles.

Los agentes complejos pueden ser vistos como otros SMA (compuestos por otros agentes), creándose así una jerarquía de agentes, por lo que se deben definir clases de agentes, con ciertos atributos que los caracterizan (habilidades, limitaciones), para facilitar la reutilización y recombinación de los mismos para formar nuevos agentes.

Los agentes del modelo SCDIA requieren de la comunidad de agentes del MGS ya descrito en el Capítulo 5, para la integración con otros SMA u otros elementos del medio donde esta inmerso el SCDIA. En la Figura 10.16 se ilustra dicha interacción.

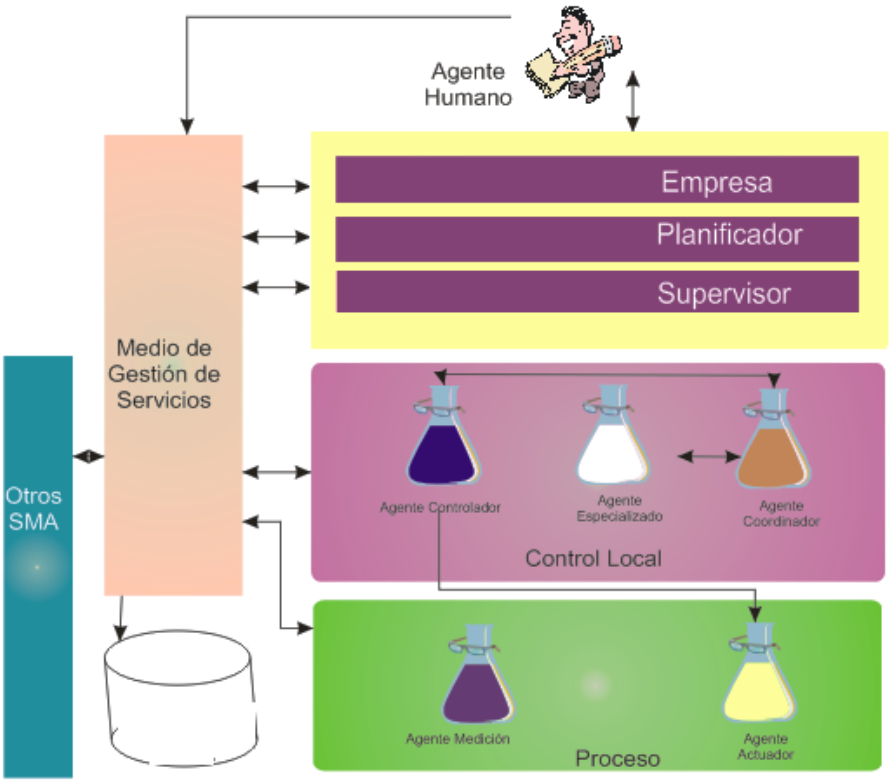

Figura 10.16: Integración del SCDIA a través del MGS

10.9.2. Modelo de un sistema de control de nivel usando SCDIA

A fin de ilustrar el uso del modelo de referencia SCDIA, se considera un sistema de control de nivel. El sistema consiste de un tanque y tres válvulas de abertura regulable y se muestra en la Figura 10.17. Las válvulas 1 y 2 regulan el vaciado de dos tipos de fluidos diferentes al tanque. Durante la mezcla de ambos fluidos (reactantes), un proceso químico toma lugar en el interior del tanque. Una vez que la gravedad específica de dicho fluido se encuentre en el rango de valores entre $G_{max} - G_{min}$, significa que se ha culminado la producción deseada. Además, hay una restricción en el nivel del fluido dentro del tanque, la cual debe estar en el intervalo $L_{max} - L_{min}$. Así, el objetivo de control es mantener las variables de altura y gravedad específica en los rangos de valores antes especificados. Se asume que existe la instrumentación necesaria para medir tanto la gravedad específica del

nuevo fluido que se esta generando (producto), así como en nivel de líquido en el interior del tanque.

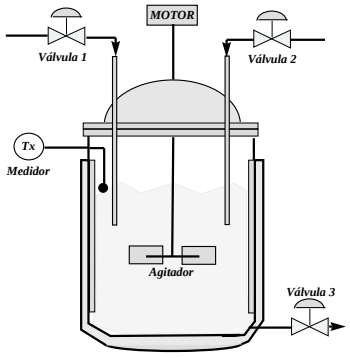

Figura 10.17: Sistema de control de una reacción

A continuación se definen los agentes de control según el modelo SCDIA:

1. **Nivel Control Local.** En este nivel se calculan las órdenes de control correspondientes a la apertura de cada una de las válvulas. Los agentes en este nivel son:
   - **Agente Coordinador.** Define las correlaciones entre las variables manipuladas y controladas, coordina las tareas de apertura de cada válvula en base a las órdenes de los agentes controladores.
   - **Agentes Controlador.** Calculan la ley de control correspondiente a la apertura de cada válvula para garantizar que los niveles del tanque y la gravedad específica del fluido estén dentro de los rangos requeridos. Existen tres agentes controladores.
2. **Nivel Proceso.** Consta del tanque, las válvulas y toda la instrumentación que hace posible la medición de las variables del proceso. En este nivel se tienen los siguientes agentes:
   - **Agentes Observación.** Existen dos tipos de agentes Observación, uno para medir el nivel y el otro para medir la gravedad específica, ya que estas son las dos variables a controlar.
   - **Agentes Actuación.** Los agentes Actuación van a ejecutar la ley de control, a fin de lograr la abertura adecuada de cada una de las tres válvulas.
   - **Agentes Interfaz.** Son agentes especializados responsables del acondicionamiento de las señales provenientes de los observadores. Dado que hay dos tipos de agentes Observación, también se definen dos tipos agentes interfaz, uno para la señal proveniente del sensor de nivel y otro para la señal proveniente del sensor de gravedad específica.

La Figura 10.18 muestra el modelo de agentes del sistema de control bajo estudio usando el modelo SCDIA. La especificación de cada agente se hace usando la metodología MASINA, descrita en la Sección 3.3.

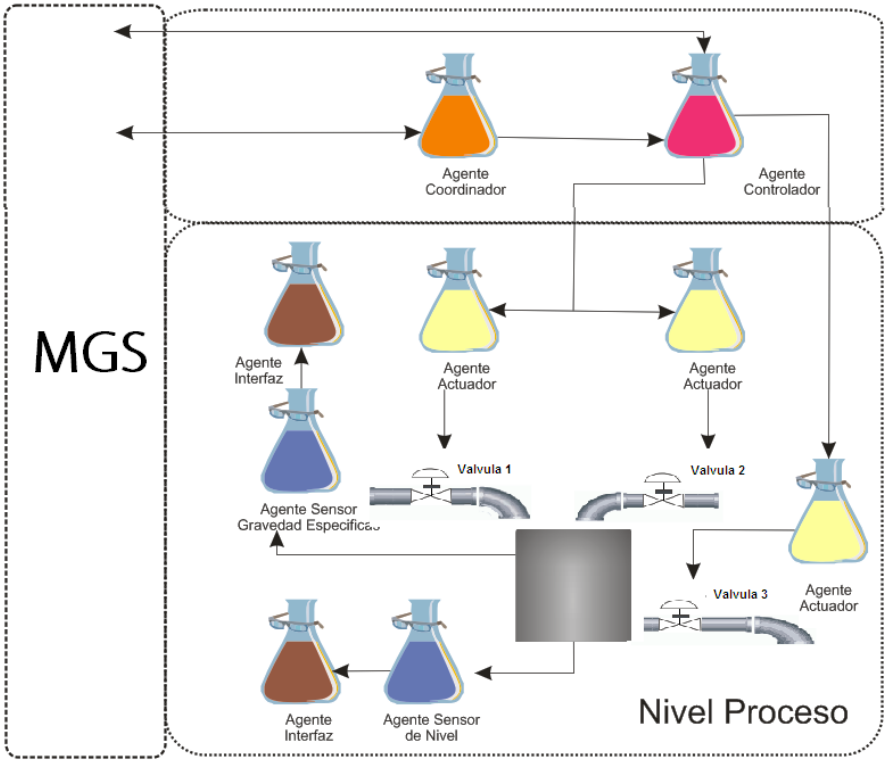

Figura 10.18: Sistema de control basado en el modelo SCDIA

10.10. Sistema Automatizado Distribuido Inteligente basado en Agentes (SADIA)

SADIA es un modelo híbrido compuesto por tres niveles de abstracción, cada uno representado por un SMA que, en el nivel más alto, modela los elementos componentes del proceso productivo, y en los niveles inferiores modelan la arquitectura de las aplicaciones que dan apoyo a dicho proceso, tales como control de proceso, supervisión o manejo de fallas.

Con el fin de dotar de inteligencia a los niveles mas bajos de la pirámide de la Figura 10.14 y maximizar los beneficios de producción enfatizando los requisitos de seguridad, confiabilidad, eficiencia y calidad de la producción, se propone el aplanamiento de las estructuras jerárquicas clásicas, como se observa en la Figura 10.19. Esta concepción permite pensar en modelos basados en agentes que permiten distribuir la mayor cantidad de actividades de la plataforma directamente sobre los procesos, y cuya consecuencia inmediata es la mayor autonomía de las unidades de producción.

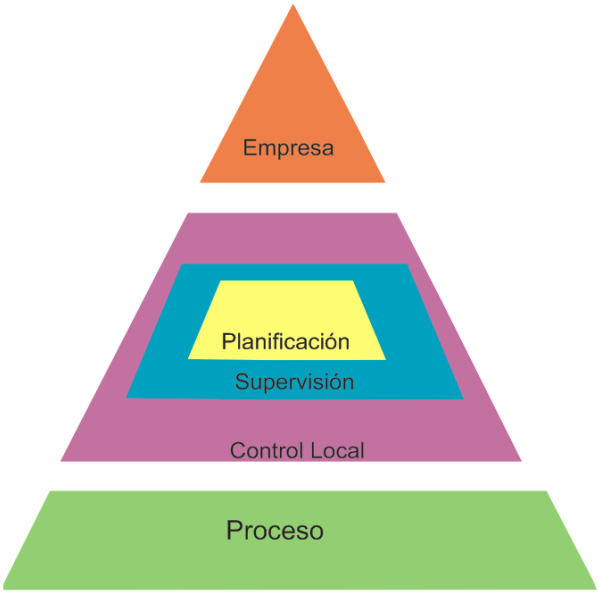

Figura 10.19: Modelo de plataforma de automatización industrial distribuida

El modelo de referencia Sistema Automatizado Distribuido Inteligente basado en Agentes (SADIA) es una propuesta de los autores presentada en [1, 3], que se enmarca en el modelo aplanado de la Figura 10.19. La Figura 10.20 muestra el modelo SADIA con sus diferentes niveles de abstracción.

10.10.1. Los niveles de abstracción de SADIA

A continuación se escriben los niveles de abstracción del modelo SADIA y los diferentes SMA definidos en cada nivel.

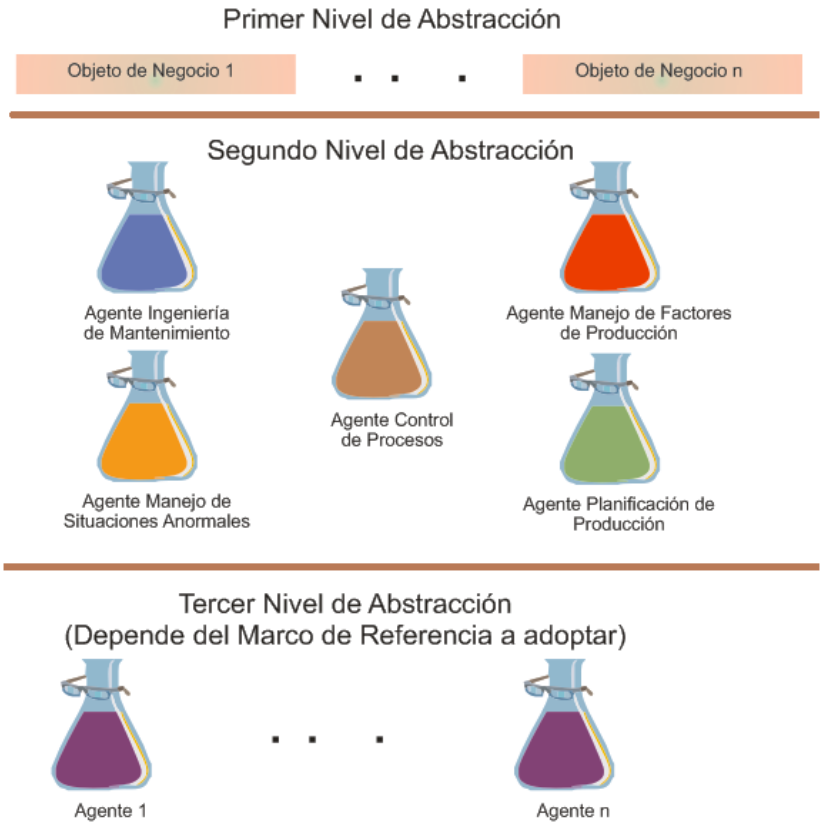

Figura 10.20: Modelo de referencia SADIA

1. **Primer nivel de abstracción.** En el primer nivel de abstracción el proceso productivo es visto como un SMA, donde las diversas unidades de producción son modeladas como agentes. Los agentes de este nivel negocian entre si para llegar a acuerdos que permitan cumplir con las metas de producción establecidas, y dichos acuerdos representan la lógica del negocio que rige el proceso productivo.
   La representación de los objetos del negocio como agentes está fundamentada en la idea de otorgar inteligencia y autonomía a cada elemento del proceso productivo. La arquitectura de este nivel es metamórfica, esto es, la cantidad y naturaleza de los agentes que componen este nivel dependerán del proceso que se está modelando, haciendo posible que la arquitectura aquí propuesta pueda ser adaptada a cualquier organización orientada a la producción. La naturaleza metamórfica de la arquitectura del primer nivel tiene que ver con cuáles son los objetos de negocio a ser representados como agentes y no con la estructura interna de los mismos.

2. **Segundo nivel de abstracción.** En este nivel se distribuye, en una colección de agentes, las actividades que se deben desarrollar para cumplir con los objetivos de cada agente del primer nivel (objetos de negocio). De esta manera, cada agente del primer nivel puede asociarse con un agente, o un SMA, del segundo nivel que realizan actividades para cumplir con las metas de cada objeto del negocio. Estas actividades pueden ser generales y comunes a todo agente del primer nivel de abstracción, como control de procesos, mantenimiento, manejo de situaciones anormales, manejo de los factores de producción y planificación de la producción.
   En consecuencia, todos los agentes del segundo nivel tendrán una arquitectura base constituida por agentes que desempeñan cada una de dichas actividades. Sin embargo, existen actividades que son exclusivas de un objeto de negocio en particular, las cuales son modeladas por medio de agentes especializados, que complementan la arquitectura base de cada agente del segundo nivel. Así, se propone la siguiente colección de agentes comunes para cada SMA del segundo nivel:

   a) **Agente Control de Procesos.** Este agente representa las aplicaciones de control de procesos, tanto a nivel de campo como a nivel de los centros de control. A nivel de campo se modelan las aplicaciones de control local que se ejecutan en dispositivos tales como PLCs o Unidades Terminales Remotas (RTU) y a nivel de centro de control se modelan las aplicaciones SCADA. Este agente también modela las aplicaciones de optimización de los sistemas de control de procesos.
   b) **Agente Planificación de Producción.** Este agente desarrolla las actividades necesarias para el cumplimiento de las cuotas de producción asignadas a cada objeto de negocio, en función de la condición y capacidad del mismo y de su interrelación con los demás objetos de negocio. Este agente debe estar dotado de capacidades para tomar decisiones, emitir órdenes de producción y pedidos, así como establecer la secuenciación de la producción. Por otro lado, este agente debe considerar métodos para la optimización de la producción.
   c) **Agente Manejo de Factores de Producción.** Este agente debe gestionar los recursos necesarios para el logro de las metas de los agentes de negocio en el proceso productivo (infraestructura, insumos, recursos humanos, energía, entre otros). Además, se encarga del manejo de los productos finales, su almacenamiento y despacho.
   d) **Agente Ingeniería de Mantenimiento.** Este agente se encarga de la elaboración de planes de mantenimiento preventivo y correctivo y de la ejecución las tareas de detección, diagnóstico, predicción y aislamiento de fallas, previstas en los planes. A partir del análisis de información de estado y de información histórica, este agente debe generar un modelo de datos que contiene toda la información acerca de los modos de fallas que se pueden presentar en el proceso, sus características, y los planes de mantenimiento que se deben ejecutar para

prevenir dichas fallas o para solventarlas, en caso de que ocurran de manera abrupta.

e) **Agente de Manejo de Situaciones Anormales.** Este agente es responsable de detectar situaciones anormales, emitir alarmas y ejecutar acciones correctivas para solventar dichas situaciones. Este agente hace uso de un modelo de datos alimentado por el agente Ingeniería de Mantenimiento ya descrito, y por una base de conocimiento que recoge la experiencia de operadores, ingenieros y expertos en manejo de situaciones anormales, que han interactuado con el proceso.

3. **Tercer nivel de abstracción.** Finalmente, ya que las actividades que desempeñan los agentes del segundo nivel son complejas, se propone un tercer nivel de abstracción en donde los agentes del segundo nivel son vistos como SMA, distribuyendo las tareas involucradas en el desarrollo de cada actividad entre diversos agentes, bajo un modelo de referencia determinado. Por ejemplo, el modelo de referencia para el agente Control de Proceso, en el tercer nivel de abstracción, puede ajustarse al modelo de referencia SCDIA presentado en la Sección 10.9, ya que dicho marco está diseñado justamente para las actividades que realiza el agente Control de Procesos.

Todos los agentes de cada uno de los niveles de abstracción hacen uso de los servicios provistos por el MGS, garantizándose así la comunicación entre todos los agentes y la gestión eficiente de los recursos y servicios requeridos por los mismos.

La descomposición en niveles de abstracción permite abordar el modelado de sistemas complejos de forma genérica, por medio de la definición de un SMA en cada nivel, lo que permite programar agentes autónomos y flexibles que desempeñan tareas específicas que pueden evolucionar de acuerdo a sus objetivos y los del SMA en el cual están inmersos.

10.10.2. El modelo SCDIA en el tercer nivel de abstracción de SADIA

Cada agente del segundo nivel de abstracción se puede modelar en el tercer nivel bajo un marco de referencia basado en SMA, con arquitecturas que van a depender de los objetivos de cada uno de ellos. El modelo SCDIA presentado en la Sección 10.9 puede usarse como marco de referencia para modelar los agentes del tercer nivel, ya que el SCDIA ofrece una visión de diseño que permite abordar cada una de las actividades involucradas en la automatización de procesos industriales. El SCDIA está inspirado en los componentes de un sistema de control de procesos, en consecuencia, su uso para modelar a los agentes del tercer nivel requiere concebir a cada agente del segundo nivel como un lazo de control.

En este caso, se considera como un *proceso* no sólo los componentes físicos que soportan el proceso de producción, sino que también abarca a los procesos lógicos soportados por sistemas informáticos (software) asociados directa o indirectamente con dichos procesos físicos, o cualquier otro elemento que pueda afectar el desempeño de la organización a modelar, incluidos los recursos humanos involucrados.

Desde el punto de vista funcional, según los objetivos de cada agente del segundo nivel de abstracción que se asocia a un SMA del tercer nivel, los agentes del modelo SCDIA se describen de la siguiente manera:

1. **Agente Observación.** Este agente tendrá la misión de:
   - Recolectar los datos provenientes de los sistemas de control local, control supervisorio, bases de datos históricas y corporativas, y los demás repositorios de datos que puedan aportar información acerca del estado de los procesos que se desarrollan en las diferentes instalaciones monitoreadas.
   - Acondicionar y/o validar los datos, calcular promedios y estimadores, hacer observación de estados y cualquier otra operación necesaria para obtener la información requerida por los demás agentes.
   - Transmitir la información de estado a los demás agentes del SMA.
2. **Agente Controlador.** Este agente recibe la información emitida por el Agente Observación con el fin de:
   - Determinar desviaciones de las condiciones actuales del proceso respecto a las condiciones deseadas y definidas en el modelo del proceso que posee el agente.
   - Ejecutar estructuras de control almacenadas dentro de motores de inferencia o cualquier otro mecanismo de razonamiento.
3. **Agente Actuación.** Este agente, en base a las decisiones tomadas por el Agente Control del Procesos, debe ejecutar acciones tales como:
   - Activar alarmas.
   - Medir variables.
   - Reconfigurar controladores locales.
   - Cambiar puntos de operación.
4. **Agente Coordinador.** Este agente es responsable fundamentalmente de:
   - Supervisar el funcionamiento de los motores de inferencia y modificarlos, en caso de ser necesario.
   - Crea y/o modificar flujos de trabajo (*workflows*).
5. **Agentes Especializados.** Estos agentes realizan tareas específicas, requeridas por algún agente del SMA. Entre las tareas específicas mas comunes se mencionan:

- Minería de datos.
- Cálculos estadísticos.
- Cálculos matemáticos de propósito general.

10.10.3. Modelo de una unidad de producción usando SADIA

El uso del modelo de referencia SADIA se ilustra con el estudio de una Unidad de Producción. Específicamente, se modela una Unidad de Explotación de Yacimiento (UEY) por Levantamiento Artificial por Gas (LAG) de un proceso de producción petrolera, el cual es uno de los métodos de levantamiento artificial más utilizado en la industria petrolera. Este método consiste en inyectar gas a alta presión en la tubería del pozo petrolero, ya sea de manera continua para aligerar la columna hidrostática en la tubería de producción (flujo continuo), o a intervalos regulares para desplazar los fluidos hacia la superficie en forma de tapones de líquidos (flujo intermitente). El gas inyectado hace que el fluido llegue a la superficie debido a la reducción de la presión que ejerce el fluido en la tubería de producción mediante la disminución de su densidad, ó debido a la expansión del gas inyectado, ó por el desplazamiento del fluido por alta presión del gas. Con este método es posible arrancar los pozos que producen por flujo natural, incrementar la producción de los pozos que declinan naturalmente, pero que aún producen sin necesidad de utilizar otros métodos artificiales, y descargar los fluidos de los pozos de gas.

Este proceso de producción es sumamente complejo, por lo que resulta interesante para la industria implementar una arquitectura de automatización que permita distribuir la inteligencia sobre todo el proceso. En esta sección se presenta el diseño de la arquitectura usando el modelo SADIA. A continuación de describe cada nivel de abstracción de dicho modelo.

1. **Primer nivel de abstracción.** En este nivel se definen los agentes asociados a los objetos del negocio, en este caso de la UEY. Para ello es necesario tener una comprensión clara de las funciones de la unidad de producción, las cuales se ilustran en la Figura 10.21.
   Las principales funcionalidades observadas en la Figura 10.21 son:

   - **Extracción de fluido.** Es el levantamiento de los hidrocarburos desde el yacimiento hasta la superficie.
   - **Recibo.** Se refiere a la recolección de crudo de los pozos asociados a las estaciones de crudo.
   - **Medida de pozos.** Comprende la ejecución de pruebas a la producción de cada pozo para determinar indicadores sobre la productividad del pozo.

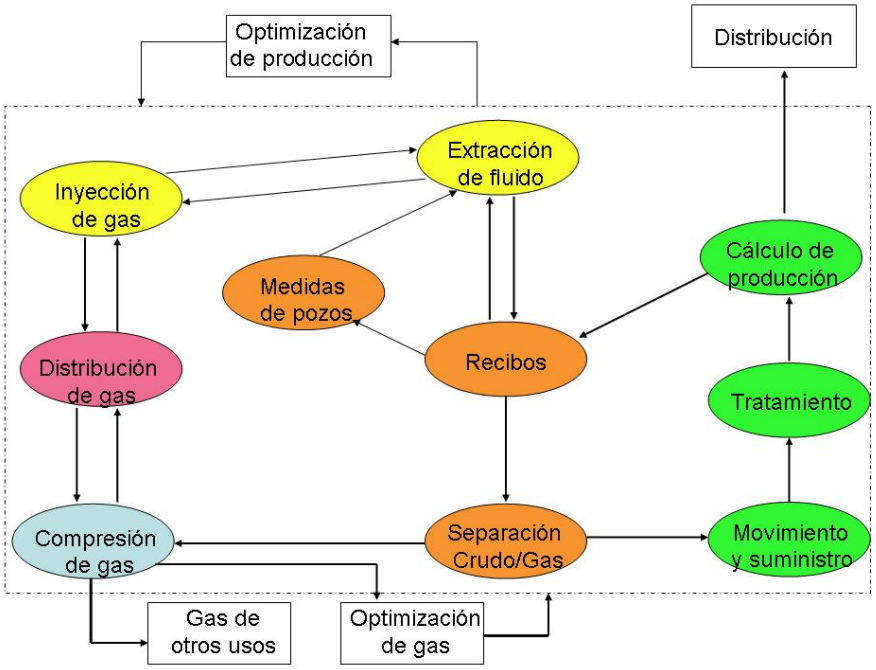

Figura 10.21: Diagrama funcional de una UEY por LAG

- **Separación crudo/gas.** Se refiere a la separación por medio de procesos mecánicos de la porción de gas que viene mezclada con el crudo.
- **Compresión de gas.** Es el proceso de elevar la presión del gas proveniente de las estaciones de flujo para su posterior uso.
- **Distribución de gas.** Comprende la asignación de cuotas de gas de inyección a cada pozo LAG.
- **Inyección de gas.** Se refiere a la inyección de gas a los pozos para aumentar la presión en el fondo del hoyo y elevar el fluido hasta la superficie.
- **Movimiento y suministro.** Comprende el almacenamiento de crudo en los patios de tanques y suministro hacia refinerías y puertos.
- **Tratamiento.** Se refiere al tratamiento físico-químico para separación de agua y eliminación de sustancias indeseables del crudo.
- **Cálculo de producción neta.** Es el cálculo de la producción neta de crudo de la UEY.

Una vez identificadas las principales funcionalidades de la UEY, se definen cinco agentes en primer nivel de abstracción los cuales representan las instalaciones típicas de un lazo de explotación por levantamiento ar-

tificial por gas y abarcan las funcionalidades ya descritas. Estos agentes son:

a) **Agente Pozo.** Este agente es responsable de todas las actividades necesarias para el funcionamiento de un pozo petrolero. Tiene la capacidad de desarrollar tareas de control, supervisión, programación de actividades de mantenimiento y/o reparación en pozos petroleros, análisis económico, y las demás actividades relacionadas con este objeto de negocio. Por otro lado, tendrá la capacidad de autoevaluarse y emprender acciones para la optimización de los métodos de producción.

b) **Agente Estación de Flujo.** Este agente modela el funcionamiento de las estaciones de flujo. Mediante él se podrán controlar y monitorear separadores, bombas, y los demás dispositivos con los que cuenta una estación de flujo. Además, mediante este agente se podrá realizar la planificación de la producción basada en el funcionamiento de la estación de flujo y de los pozos asociados a ella. Por otro lado, este agente contendrá métodos de optimización para la separación de gas/crudo y de análisis de productividad de la UEY.

c) **Agente Planta Compresora.** Este agente monitorea y controla las actividades relacionadas con las instalaciones destinadas a la compresión de gas en el lazo de explotación y, mediante coordinación con otros agentes, podrá planificar el consumo de gas por parte de las múltiples LAG y las demás instalaciones de la UEY, e incluso, por parte de instalaciones externas .

d) **Agente MLAG.** Este agente maneja las actividades relacionadas con la distribución del gas de inyección, llevadas a cabo en las múltiples LAG. Mediante este agente se puede hacer la planificación de la distribución de gas en la UEY, y aplicar métodos de optimización de producción para los pozos que funcionan basados en inyección de gas.

e) **Agente Patio de Tanques.** Este agente monitorea y controla las actividades de los patios de tanques de una UEY. Este agente permite establecer métodos de optimización para el movimiento y suministro del crudo, así como para su pre-tratamiento.

En la Figura 10.22 se muestra la definición de los agentes del primer nivel en relación a la especificación funcional de la UEY.

2. **Segundo nivel de abstracción**. En el segundo nivel de abstracción se diseñan los agentes (SMA) que realizan las actividades del área de la automatización industrial. Por esta razón, para modelar el comportamiento de los SMA del segundo nivel de abstracción, se requiere construir el diagrama funcional de control de la empresa para entender las funcionalidad del segundo nivel [1]. Este diagrama se muestra en la Figura 10.23, y sus principales funcionalidades son:

[1] Una diagrama detallado puede obtenerse usando el estándar ANSI/ISA 95.00.01, el cual describe las funciones y el flujo de información entre los diversos componentes de una empresa orientada a la producción.

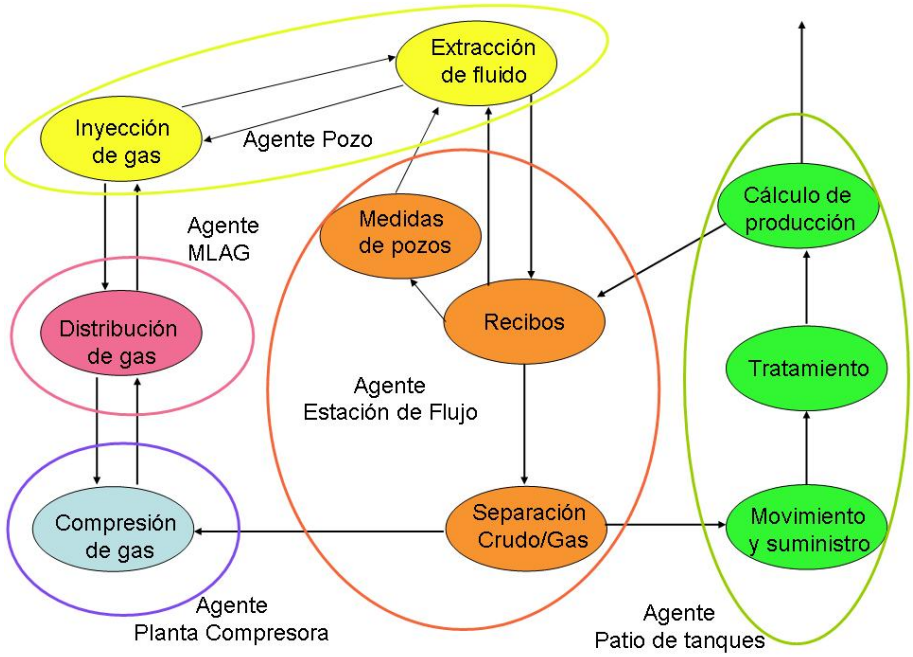

Figura 10.22: Agentes del primer nivel de SADIA

- **Procesamiento de orden.** Se refiere al manejo de las órdenes de los clientes.
- **Planificación de producción.** Tiene que ver con la elaboración y ejecución de planes de producción, determinación de los requerimientos de materia prima y estimación de la disponibilidad de productos finales.
- **Control de la producción.** Comprende el control de la transformación de la materia prima en productos finales, de acuerdo al plan de producción.
- **Control de insumos y energía.** Se refiere a la gestión del inventario, transferencia y calidad de los insumos y de la energía disponible.
- **Procura.** Es la ejecución de las órdenes de requerimientos de materiales, partes, insumos y demás elementos necesarios para la producción.
- **Control de calidad.** Esta funcionalidad debe asegurar la calidad de los productos finales, siguiendo estándares y normas.
- **Control de inventario de productos.** Es el manejo del inventario y la disponibilidad de los productos finales.
- **Control de costos de producción.** Se refiere al cálculo y ejecución de reportes de los costos de la producción.
- **Despacho de productos.** Es la organización del despacho, transporte y entrega de los productos finales a los clientes.

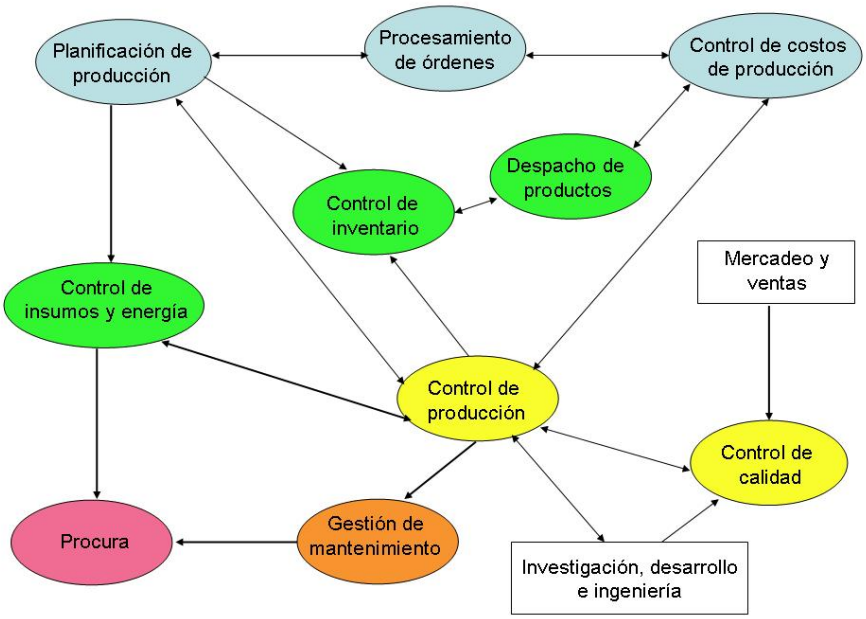

Figura 10.23: Diagrama funcional de control de una UEY por LAG

- **Gestión de mantenimiento.** Se refiere al mantenimiento de las instalaciones, ejecución de planes preventivos de mantenimiento, monitoreo de fallas y situaciones anormales, cálculo de confiabilidad operacional.

Una vez identificadas las principales funcionalidades de la UEY, en cuanto a las actividades de producción, se asocian dichas funcionalidades a alguno de los agentes del segundo nivel de abstracción ya definidos en la Sección 10.10.1. Luego:

a) **Agente Planificación de Producción.** Tiene las funcionalidades de procesamiento de órdenes, planificación de producción y control de costos de producción.
b) **Agente Control de Procesos.** Tiene las funcionalidades de control de la producción y control de calidad.
c) **Agente Manejo de los Factores de Producción.** Tiene las funcionalidades de control de insumos y energía, procura, control de inventario de productos y despacho de productos.
d) **Agente Manejo de Situaciones Anormales.** Tiene algunas de las funcionalidades de gestión de mantenimiento.
e) **Agente Ingeniería de Mantenimiento.** Tiene algunas de las funcionalidades de gestión de mantenimiento.

En la Figura 10.24 se muestra la definición de los agentes del segundo nivel en relación a la especificación funcional de la UEY.

Figura 10.24: Agentes del segundo nivel de SADIA

3. **Tercer nivel de abstracción**. En el tercer nivel de abstracción se diseña el SMA asociado a cada uno de los agentes del segundo nivel, usando algún marco de referencia basado en agentes. El aspecto fundamental es nuevamente comprender las funcionalidad de cada SMA del tercer nivel para asociar luego un agente con un grupo de funcionalidades. Si se usa el modelo SCDIA para este nivel, entonces tendremos definidos el conjunto de agentes de actuación, observación, controlador, coordinador y especializado donde estarán distribuidas las funcionalidades de este tercer nivel. La última parte de este libro está dedicada a la especificación de agentes del tercer nivel usando el modelo SCDIA.

Referencias

1. 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.

2. A.I. Anosike and D.Z.Zhang. An agent-based approach for integrating manufacturing operations. *Interntional Journal of Production Economics*, 121:333—-352, 2009.
3. Bravo C., J. Aguilar, Rivas R., and Cerrada M. Design of an Architecture for Industrial Automation based on Multi-Agents Systems. In *Proceedings of 16th IFAC World Congress*, July 2005.
4. The PABADIS consortium. PABADIS - White Paper. Thecnical Report, University of Magdeburg, Germany, October 2001.
5. F. Hidrobo, A. Ríos-Bolívar, J. Aguilar, and L. León. An architecture for industrial automation based on intelligent agents. *WSEAS Transaction on Computers*, 12(4):1808–1815, 2005.
6. 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.
7. Axel Klostermeyer. Revolutionising plant automation – the pabadis approach - white paper. Technical Report IST-1999-60016, PABADIS: Information Society Technology, Germany, 2003.
8. A. Lüder, J. Peschke, A. Bratukhin, A. Treytl, A. Kalogeras, and J. Gialelis. The PABADIS PROMISE - Architecture. In *Proceedings of International Congress ANIPLA 2006*, *Methodologies for Emerging Technologies in Automation*, Rome, November 2006.
9. Paulo Leitao. Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22:979—-991, 2009.
10. H. Van Dyke Parunak, Albert D. Baker, and Steven J. Clark. The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design. *Integrated Computer-Aided Engineering*, 8(1):42–58, January 2001.
11. A. Ríos-Bolívar, J. Aguilar, F. Hidrobo, and L. León. Industrial automation architecture: An intelligent agent based approach. In *Proceedings of 4th WSEAS Int. CIMMACS'05*, pages 134–139, Jul 2005.
12. W. Shen, F. Maturana, and D. H. Norrie. MetaMorphII: An Agent-Based Architecture for Distributed Intelligent Desing and Manufacturing. *Journal of Intelligent Manufacturing*, 11(3):273–251, June 2000.
13. S. Turgay. Agent-based FMS control. *Robotics and Computer-Integrated Manufacturing*, 25:470—-480, 2009.
14. L.C. Wang and S.K. Lin. A multi-agent based agile manufacturing planning and control system. *Computers in Industry*, 57:620—-640, 2009.

Capítulo 11
Control de Procesos Basado en Agentes

Resumen: En este capítulo se aborda, de manera descriptiva, una arquitectura para el desarrollo de SMA para el control y monitoreo de procesos industriales, a los fines de garantizar una operación segura y óptima en base a los objetivos de producción.

11.1. Introducción

Para ser altamente competitivas, las empresas de producción deben mantenerse en el *estado del arte* en tecnología, métodos, técnicas, procesos, mercado, etc. a partir del sostenimiento de un mejoramiento continuo y/o planes de proyecciones futuras, y para lograr estas metas, el manejo de la información es de importancia capital. Es de allí que surge la necesidad de disponer de sistemas de control y supervisión muy sofisticados.

El control de procesos industriales consiste en satisfacer ciertos objetivos de regulación, por ejemplo calidad del producto, seguridad del personal, ambiental y de los equipos, control del balance de los materiales, etc; y ciertos objetivos económicos, bajo la presencia de una amplia variedad de perturbaciones de la planta. Las perturbaciones típicas son: cambios en las propiedades de los materiales, especificaciones de productos diferentes, variaciones en la demanda del mercado, fluctuaciones de los precios, etc. Esta clasificación de los objetivos del control en dos categorías orienta las actividades en el diseño de las estructuras de control regulatorio y de la optimización de los procesos.

Existe una diversidad de problemas de control que, por su complejidad y carácter dinámico, no han sido satisfactoriamente considerados o resueltos empleando los enfoques convencionales de la teoría de control. El rápido desarrollo de componentes de hardware de alta capacidad y de TIC lideri-

zan una fuerte necesidad de integración en sistemas automáticos, donde se requieren tareas de supervisión y control altamente complejas [19]. Esto ha obligado a pensar en nuevos paradigmas basados en la teoría de inteligencia artificial distribuida para el diseño de herramientas que implementen control y supervisión inteligente. Así, en problemas complejos de control puede resultar atractivo el uso del aprendizaje por medio de la experiencia o el razonamiento heurístico para la toma de decisiones, análogas a las del razonamiento humano, que permitan producir soluciones confiables. Aquellos casos en los cuales la complejidad del problema o las incertidumbres alrededor del mismo, hacen difícil, sino imposible, el establecimiento detallado de especificaciones, parecen ser los apropiados para adoptar enfoques que utilicen el aprendizaje como herramienta para el diseño de soluciones satisfactorias. La obtención de tales soluciones estará en función de la cantidad y calidad de la información disponible sobre el problema tratado.

En los sistemas de control inteligente se hace uso de diversas metodologías que combinan aspectos de la teoría convencional de control con técnicas emergentes inspiradas en la inteligencia artificial distribuida y una variedad de técnicas de optimización de procesos. Las características generales de un sistema inteligente de control son las relacionadas con sus habilidades para emular capacidades humanas como la planificación, el aprendizaje y la adaptación. El tipo de problemas que justifican el diseño de los sistemas de control inteligente son, en general, de gran escala, de naturaleza híbrida, posiblemente modelados por ecuaciones diferenciales, en diferencias, por modelos de eventos discretos o combinación de todo lo anterior. En el diseño de controladores para tales procesos dinámicos complejos están presentes requerimientos que no son suficientemente contemplados por la teoría convencional de control, excepto el control robusto. (Por ejemplo problemas de incertidumbres en el ambiente) [19].

Los sistemas de control inteligente son diseñados para mantener un rendimiento satisfactorio, en lazo cerrado, de un proceso y su integridad sobre un amplio rango de condiciones de operación. Las características particulares del sistema inteligente deben entonces estar relacionadas con la complejidad del proceso; incluyendo comportamientos no lineales y variantes en el tiempo, aspectos de la dimensionalidad o el número de variables, la complejidad del objetivo de rendimiento deseado, las mediciones imprecisas y la habilidad para soportar la ocurrencia de fallas. Un término alternativo para hacer referencia al control inteligente es el de control autónomo.

Es así como, en el contexto de la inteligencia artificial distribuida, el paradigma de desarrollo de software orientado a agentes permite diseñar sistemas sofisticados y complejos. En consecuencia, y en base a sus propiedades, la tecnología de agentes pueda ser utilizada para satisfacer requerimientos para la automatización de procesos, tales como la supervisión y el control inteligente [20, 1].

Por lo tanto, en este capítulo se describe una arquitectura que permite el desarrollo de SMA para el control y monitoreo de procesos industria-

les, garantizando la operación segura y óptima en base a los objetivos de producción. El esquema de desarrollo se fundamenta, primeramente, en la definición de la arquitectura de implantación del SMA en un ambiente de automatización industrial, y en segundo lugar, en la definición y especificación de los agentes usando la metodología MASINA, la cual ha sido descrita en la Sección 3.3; y el lenguaje UML [1, 11]. En el marco del desarrollo y con el objetivo de fijar las bases que conllevan a la implantación de los agentes, se presenta la arquitectura que debe soportar aplicaciones de control y supervisión de procesos con la filosofía de agentes inteligentes.

11.2. Control inteligente de procesos

Para lograr los objetivos del control regulatorio en industrias de cierta complejidad, se deben cumplir continuamente, labores de:

1. Identificación de los procesos.
2. Síntesis de algoritmos de control.
3. Simulaciones de procesos.
4. Implantación de la política de control.
5. Diagnóstico del proceso.

Estas tareas se recogen, de manera algorítmica, en la Figura 11.1.

Las estrategias de control y supervisión hacen uso intensivo de la información, que indican la disponibilidad de dispositivos y equipos, capacidades y condiciones de operacionales, que inducen el tipo de control a ser ejecutado durante un lapso de tiempo. Así, para un adecuado e inteligente control de las operaciones es necesario disponer de:

- Una plataforma de datos y sistemas de información
- Modelos cada unidad de producción, asociados a la física del proceso
- Modelos asociados a las interacciones entre las diferentes unidades

Los modelos permiten construir sistemas de control, sistemas de apoyo a la toma de decisiones y sistemas para determinar el estado de un proceso, de una unidad de producción o del complejo productivo.

En este contexto, los sistemas de información permiten la realización de la integración de las operaciones mediante una plataforma de datos y transformaciones de los mismos, de tal manera que presten información (movimiento de datos) y faciliten la transmisión de lineamientos entre los diferentes niveles, así como también información horizontal dentro de un mismo nivel, tal como se muestra en la Figura 8.2.

Los sistemas de información respaldan internamente cada nivel cuando mantienen actualizados los datos del proceso, los cuales determinan las condiciones del mismo, su rendimiento y facilitan la implantación de algoritmos para realizar las actividades de control que modifican su comporta-

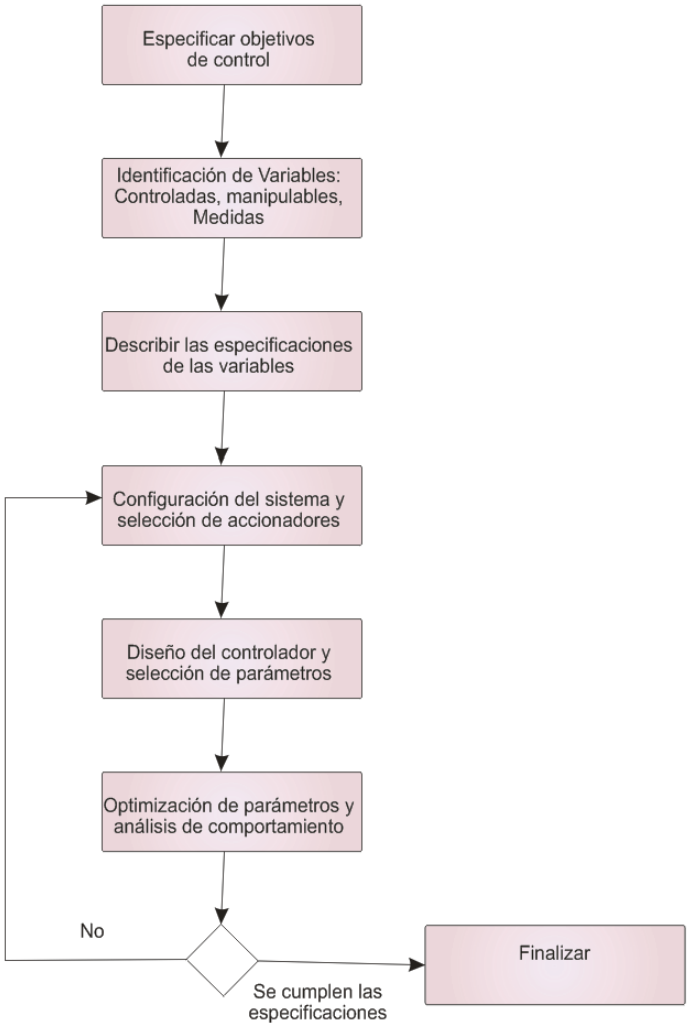

Figura 11.1: Flujograma para el diseño de sistemas de control

miento. En la Figura 11.2 se muestra el funcionamiento de un sistema realimentado que permite la ejecución de acciones de control sobre el proceso. El

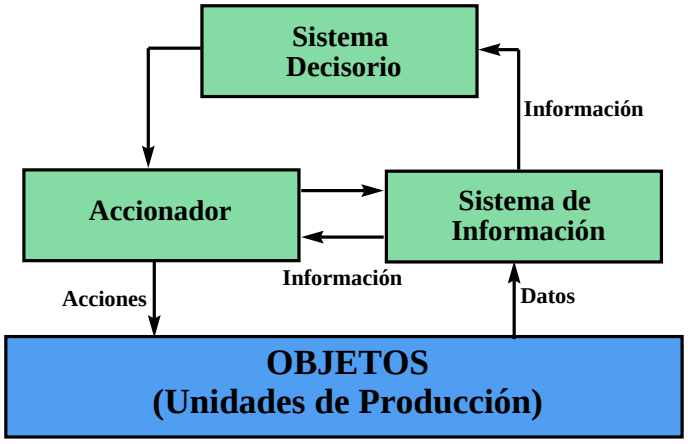

Figura 11.2: Sistema de Información como soporte para la toma de decisiones

sistema de información mantiene los datos de los objetos bajo control, produce información a partir de esos datos para definir estados, condiciones de operación, capacidades del sistema, cumplimiento de metas, etc. La información existente en el sistema de información a su vez, alimenta al sistema actuador, el cual transforma las decisiones en lineamientos o controles hacia el objeto.

Es así que, en la actualidad, es común encontrar esquemas de control de procesos industriales complejos que deben satisfacer múltiples objetivos y por lo tanto deben operar en diferentes modos de control.

Definición 11.1. El *control inteligente* es la habilidad de un sistema de control de operar en múltiples ambientes y con multi-objetivos, partiendo del reconocimiento de la situación específica y sirviéndola adecuadamente, según los objetivos de funcionamiento.

Cada situación específica define el modo de operación y cada uno de los modos tiene un controlador que se ha diseñado para satisfacer los objetivos de desempeño correspondiente a ese modo. La conmutación entre regímenes de operación se logra por una selección de la entrada de la planta a partir de las salidas de un número de controladores en paralelo, cada uno de los cuales corresponde a un modo en particular. Entonces, la conmutación de

modos es la sustitución de la entrada a la planta en virtud de que la salida de un controlador se reemplaza por otro.

Partiendo de la existencia de un sistema informático que permite el desarrollo de aplicaciones novedosas y de alto desempeño, que se dirijan hacia el mejoramiento de las operaciones automatizadas, se pueden desarrollar sistemas que puedan actuar en las diferentes tareas del control y la supervisión inteligente; determinados a partir de los datos presentes en los diferentes plataformas de información. A partir de la detección de cambios en las plantas y la presencia de sistemas que evalúen y suministren mejoras en diferentes escenarios, será posible tomar decisiones de manera óptima en base a las nuevas condiciones. Entonces, el lazo de control se cierra a través de mecanismos que transformen estas decisiones en secuencias de operaciones o consignas a las plantas.

El uso de sistemas inteligentes y las técnicas de control avanzado en los procesos técnicos puede darse en los diferentes niveles de su estructura organizacional. Por un lado, los sistemas inteligentes se pueden usar como herramientas para el apoyo en las tomas de decisiones, para el modelado de los componentes de esa estructura, para la simulación dinámica de los procesos bajo diferentes condiciones, etc. Por otro lado, las técnicas de control avanzado tienen su mayor fortaleza de aplicación en los niveles de control regulatorio y supervisorio y en niveles de optimización de procesos, permitiendo crear mecanismos de detección de fallas, sensores virtuales, o control en algunos lazos difíciles de regular con las técnicas comúnmente utilizadas. Las oportunidades de usar estas metodologías vienen dada por la versatilidad de las mismas para ser implantadas como mecanismos de solución en diferentes problemas, sin importar la complejidad en la información que se maneja. Tanto los sistemas inteligentes como las técnicas de control avanzado tienen su mayor fortaleza en sus capacidades adaptativas que les permita manejar adecuadamente variabilidad en los elementos y suministros, restricciones físicas y cambios en la condición de operación de los procesos [5].

En general, todo desarrollo en sistemas inteligentes y de control avanzado, dentro de una organización, debe considerar:

1. La adecuación de los datos a las algoritmos inteligentes a ser utilizados.
2. Las características cualitativas de los datos.
3. El tipo de información que deben generar y de que manera se deben incorporar a las actividades del proceso productivo.
4. Las herramientas que deben usarse para su desarrollo.
5. Las necesidades y capacidades de la organización, las cuales deben ser restricciones de los sistemas a desarrollar, (capacidad de cálculo, tiempos de respuesta, etc.)
6. Cuales son las estructuras de los sistemas de instrumentación y control y su situación de disponibilidad.
7. Las posibilidades de crecimiento dentro de los sistemas existentes.

Como ventajas en la utilización del control inteligente en un ambiente industrial, se puede señalar:

1. La posibilidad de utilizar información incompleta. Partiendo de este hecho como hipótesis de trabajo, los sistemas deben tener capacidades de asociar e inferir información.
2. La incertidumbre en la coherencia e integridad de los datos. La posibilidad de usar fuentes distintas de datos (con valores o formatos de representación diferentes para un mismo dato, etc.) hace necesario manejar incertidumbre e integración como parámetros fundamentales de diseño.
3. La gran cantidad de datos manejados/almacenados. La cantidad de datos para estas actividades permiten desarrollar sistemas inteligentes con un gran valor estratégico que hasta ahora no ha sido explotado. Así, en actividades conocidas como *información a futuro* hay un gran área de posibles aplicaciones. Esto es conocido como *simulación estratégica*, punto aun no explotado en toda su magnitud en el control y supervisión de procesos.
4. La posibilidad de implantación directa sobre mecanismos actuadores. Esto amplifica las oportunidades de utilización de estos sistemas, pero se deberán considerar aspectos, tales como: tiempos de respuestas exigidos, hardware disponible, etc.

Para el desarrollo de los sistemas de control y supervisión es necesario tomar especial cuidado en los siguientes aspectos:

- Los requerimientos de cálculo y complejidades de los sistemas inteligentes. Tanto a nivel de control directo como a nivel gerencial, el proceso de utilización de estos sistemas está limitado por la capacidad de cálculo disponible.
- La motivación de los posibles usuarios. De manera general, la receptividad de los usuarios viene dado por el conocimiento de las virtudes de los sistemas a utilizar. El proceso de capacitación de posibles usuarios viene dado por el conocimiento de estas técnicas. Esto también afectará de alguna u otra manera el éxito en el uso de estos sistemas.
- La posibilidad de disponer de datos que permitan el desarrollo de modelos para ser aplicados en la construcción de sistemas de control inteligente.

Para la caracterización de las oportunidades de desarrollo de estos sistemas, es necesario enumerar, primeramente, una lista general de tipos de aplicación, que posteriormente serán catalogados por: el nivel en la pirámide de automatización, plataforma computacional, tipo de técnica a ser utilizada y las expectativas extraídas de las especificaciones de control y supervisión.

La existencia de un sistema de instrumentación moderno (en el estado del arte de la tecnología), que permita realizar mediciones sobre el proceso físico; con una infraestructura teleinformática robusta que permita conocer

las condiciones y variables de los procesos, son requerimientos imprescindibles para lograr niveles de automatización elevados, tendientes a mejorar el desempeño del proceso productivo, partiendo de la supervisión y el control.

11.3. Sistemas de control con agentes inteligentes

Siguiendo la definición de control inteligente, el desarrollo de sistemas de control y supervisión de procesos industriales se puede abordar a partir de *agentes inteligentes*. En primer lugar, se debe definir la plataforma general de implantación que soporta al SMA. A partir de allí se conciben aplicaciones como SMA especializados, definidos para coordinar, ejecutar y evaluar tareas de control y supervisión necesarias en el procesamiento de la información del proceso y la toma de decisiones. Los agentes de control y supervisión interactúan con un SMA que modelan los elementos de las unidades de producción a través de abstracciones lógicas y funcionales de los procesos. Los agentes se construyen utilizando la metodología MASINA [2, 1], y UML [11].

Los sistemas para la automatización industrial son aplicaciones que se caracterizan por requerimientos bien específicos de desempeño, funcionalidad, productividad y de seguridad operacional. Esto es, la automatización industrial debe satisfacer requerimientos de seguridad, confiabilidad, eficiencia y calidad [4]. Por otro lado, para la automatización de procesos se usan sistemas de software y hardware de gran escala, complejos, distribuidos y persistentes, los cuales son definidos en función de las características de los procesos técnicos a ser controlados y supervisados [12].

El rápido desarrollo de componentes de hardware de alta capacidad y de las tecnologías de información y comunicación (TIC), liderizan una fuerte necesidad de integración en sistemas automáticos, donde se requieren tareas de supervisión y control altamente complejas. Esto ha obligado a pensar en nuevos paradigmas basados en la teoría de inteligencia artificial distribuida para el diseño de herramientas que implementen control y supervisión inteligente. De esa teoría, el paradigma de desarrollo de software orientado a agentes permite diseñar sistemas sofisticados y complejos.

Típicamente, las tareas de automatización de procesos no han sido desarrolladas como una aplicación de las nuevas tecnologías de información, entre las que se encuentran los agentes inteligentes. Sin embargo, algunas investigaciones se han orientado hacía el uso de la tecnología de agentes en la implementación de sistemas de automatización de procesos [3, 6, 17, 18]. En general, las aplicaciones se han caracterizado por la realización de un acoplamiento entre los principios operacionales de los sistemas de automatización de procesos y los agentes inteligentes, permitiendo obtener sistemas distribuidos y de ingeniería complejos.

Los sistemas automatizados se pueden representar mediante diferentes niveles, cada uno de los cuales tiene características operacionales adecuadas: nivel de dispositivos de campo (nivel operacional), para la captura de la información de los procesos, nivel de control supervisorio y optimización (nivel táctico), donde se ejecutan las tareas de control, y nivel de gerencia de los procesos (nivel estratégico), donde se evalúan y desarrollan las estrategias de producción. Esta arquitectura de operación jerárquica permite la distribución de las funcionalidades de las actividades de automatización a través de la descripción de las diferentes tareas operacionales, tácticas y estratégicas. Fundamentalmente y de manera tradicional, la inteligencia reside en los niveles superiores [8, 7].

En este mismo sentido, el paradigma de agentes inteligentes es una manera natural de descomposición de sistemas y una alternativa razonable para implantar las funcionalidades de automatización en los diferentes niveles [7, 17]. Así, los niveles de un sistema automatizado se pueden representar por sub-sistemas y componentes de los sub-sistemas, los cuales son definidos por agentes y comunidades de agentes. Las interacciones entre sub-sistemas y componentes son definidas por mecanismos de cooperación, coordinación y negociación. Por medio de mecanismos explícitos se establecen las relaciones entre sub-sistemas y componentes. Por lo tanto, la inteligencia puede ser distribuida en los distintos niveles. De esta manera, el paradigma de agentes inteligentes es adecuado para cumplir con los requerimientos de los sistemas de automatización actuales, donde la reconfigurabilidad y la flexibilidad, conjuntamente con la inteligencia, son aspectos importantes a satisfacer [9, 10, 16, 18].

En este contexto, es bien sabido que los controladores de procesos actuales son sistemas autónomos reactivos, por lo tanto constituyen una aplicación natural para evolucionar hacia agentes inteligentes. No es sorprendente, entonces, desarrollar aplicaciones de control de procesos basadas en agentes inteligentes.

Para el control inteligente de procesos es necesario satisfacer múltiples objetivos, para lo cual se deben considerar diferentes escenarios de operación. Esta cualidad demanda que los sistemas de control y supervisión satisfagan requerimientos de adaptabilidad, flexibilidad, autonomía, operatividad concurrente y colaborativa, que son los aspectos que mas resaltan de los agentes inteligentes [3]. Tradicionalmente, las aplicaciones de agentes en control y supervisión de procesos no se diseñan para que satisfagan esas características, por el contrario, se parte de que los controladores son sistemas reactivos y a partir de allí se definen los agentes para las tareas de supervisión de esos controladores [15]. Por consiguiente, mediante el desarrollo de SMA para el control y supervisión de procesos industriales, es posible alcanzar una operación segura, óptima e inteligente, en base a los objetivos de producción.

El esquema de desarrollo se fundamenta, primeramente, en la definición de la arquitectura de implantación del SMA en un ambiente de automati-

zación industrial, y en segundo lugar, en la definición y especificación de los agentes usando la metodología MASINA y el lenguaje UML [1]. Según MASINA, la definición de los agentes que conforman el SMA se realiza en la fase de conceptualización, utilizando diagramas de casos de uso y de actividades de UML. La especificación detallada de los agentes definidos en esta fase se realiza en la fase de análisis a través del modelo de agente, modelo de tarea, modelo de inteligencia, modelo de coordinación (diagrama de interacción de UML) y modelo de comunicación. El diseño de los agentes se realiza utilizando TDSO, la cual incluye el diagrama de clases de UML, tal como ha sido descrito en el Capítulo 3.

En este contexto y con el objetivo de fijar las bases que conllevan a la implantación de los agentes, se presenta a continuación, la fase de conceptualización del Agente Proceso, del Agente Control y del Agente Supervisor, ubicados en el nivel superior de la arquitectura de implantación, y el modelo de coordinación, descrito por diagramas de interacción, definido en la fase de análisis, el cual permite caracterizar las conversaciones entre agentes necesarias para el cumplimiento de los servicios y objetivos del agente dentro del SMA. Asimismo, se presentan algunos diagramas de clases que corresponde a la fase de diseño.

11.3.1. Arquitectura de implantación

En primer lugar, tal como ha sido descrito, los sistemas automatizados se pueden representar mediante diferentes niveles, cada uno de los cuales tiene características operacionales adecuadas: nivel de dispositivos de campo (nivel operacional), para la captura de la información de los procesos, nivel de control supervisorio y optimización (nivel táctico), donde se ejecutan las tareas de control, y nivel de gerencia de los procesos (nivel estratégico), donde se evalúan y desarrollan las estrategias de producción. Esta arquitectura de operación jerárquica permite la distribución de las funcionalidades de las actividades de automatización a través de la descripción de las diferentes tareas operacionales, tácticas y estratégicas. Fundamentalmente, y de manera tradicional, la inteligencia reside en los niveles superiores [14].

En este mismo sentido, el paradigma de agentes inteligentes es una manera natural de descomposición de sistemas, y una alternativa razonable para implantar las funcionalidades de automatización en los diferentes niveles. Así, los niveles de un sistema automatizado se pueden representar por sub-sistemas y componentes de los sub-sistemas, los cuales son definidos por agentes y comunidades de agentes. Tradicionalmente, las aplicaciones de agentes en control y supervisión de procesos no se diseñan para que satisfagan esas características, por el contrario, se parte de que los controladores son sistemas reactivos, y a partir de allí se definen los agentes para las tareas de supervisión de esos controladores [16, 7]. Esto representa una pri-

mera alternativa de aplicación de agentes inteligentes en la automatización industrial, tal como se muestra en la Figura 11.3.

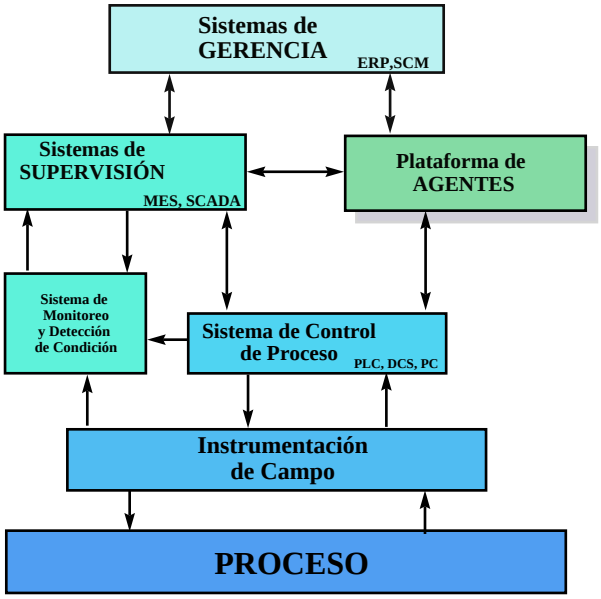

Figura 11.3: Agentes integrados en automatización industrial.

En esta manera de aplicación de agentes persisten todos los sistemas de control en tiempo real que se conocen hoy día (PLCs, SCADA, DCS, etc.), y los mecanismos de monitoreo y supervisión que soportan el diagnóstico del proceso productivo. Los agentes pueden ser aplicados para optimizar la producción. Otra alternativa, consiste en aplicar la filosofía de agentes inteligentes en todos los niveles de la automatización, tal como se muestra en la Figura 11.4.

Las interacciones entre sub-sistemas y componentes son definidas por mecanismos de cooperación, coordinación y negociación. Por medio de mecanismos explícitos se establecen las relaciones entre sub-sistemas y componentes. Por lo tanto, la inteligencia puede ser distribuida en los distintos niveles. De esta manera, el paradigma de agentes inteligentes es adecuado para cumplir con los requerimientos de los sistemas de automatización actuales, donde la reconfigurabilidad y la flexibilidad, conjuntamente con la inteligencia, son aspectos importantes a satisfacer [9, 16, 18].

El uso de la tecnología de agentes para el diseño de sistemas de automatización de procesos facilita la captura de las propiedades inherentes en estos sistemas. Sin embargo, la implantación de los servicios requeridos para so-

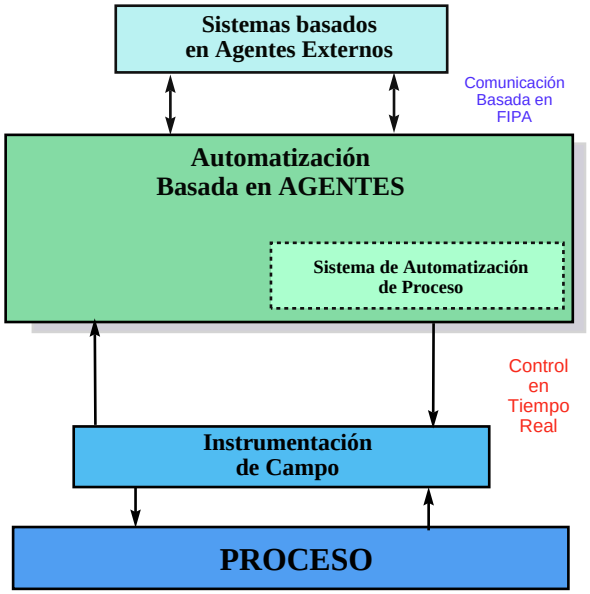

Figura 11.4: Agentes en automatización integrada.

portar los elementos de diseño en SMA puede ser una tarea muy tediosa para usuarios no especializados en desarrollo de software, con restricciones de tiempo real. En este sentido, existen algunas propuestas que presentan la solución a través de un Medio de Gestión de Servicios; entre éstas destaca JADE, la cual es una implantación del estándar FIPA. La principal desventaja de JADE es que no toma en cuenta las restricciones tiempo real para el manejo agentes.

Por otro lado, como ya ha sido mencionado, las principales funcionalidades de un sistema de control y supervisión de procesos son: monitorear las variables de operación, procesar dichas variables para generar comandos de control, transmitir los comandos de control, permitir la reconfiguración de los algoritmos de control, detectar y diagnosticar condiciones anormales de operación. Estas funcionalidades y tareas pueden ser distribuidas y expresadas a través de una estructura de interacción concurrente y colaborativa, de acuerdo al sistema de información.

Desde el punto de vista del uso de agentes inteligentes, se puede definir una arquitectura de funcionamiento, tal como ha sido presentada en la Sección 10.8. Allí se distinguen tres niveles funcionales: *nivel de campo* donde se encuentran los elementos actuadores y sensores; nivel de *Middleware* o Medio de Gestión de Servicios (MGS) y *nivel de aplicaciones* de agentes inteligentes.

De esos niveles, bajo la orientación de implantación de sistemas de control y supervisión, es importante destacar lo referente al sistema operativo base que se utiliza, el cual puede ser cualquiera que permita el manejo del tiempo real. En particular se sugiere el uso Linux más las extensiones tiempo real ofrecidas por RTAI[1] [8]. En la Sección 5.4 se especifica, de manera detallada, las características de funcionamiento para tiempo real.

En la arquitectura, en el nivel superior existen dos comunidades de agentes: *Agentes de Procesos*, los cuales describen los diferentes objetos del medio de producción (abstracciones lógicas y funcionales de los procesos reales). *Agentes de Aplicaciones y Especializados*, conformada por todos los agentes que realizan funciones específicas a nivel de supervisión, control, optimización, visualización, y otras aplicaciones especializadas y/o legadas. Esta comunidad brinda servicios a la comunidad de agentes de procesos. Ambas interactúan con el MGS a través de la capa de interfaz.

11.3.1.1. Control basado en Agentes

Cabe destacar que las actividades de los agentes del nivel superior están basadas en los requerimientos de control de procesos. Estos, a su vez, establecen la necesidad de comunicaciones y tomas decisiones en tiempo real, adecuándose a situaciones no previstas mediante la detección de eventos, [13]. En el diseño del nivel superior podrían estar contemplados los siguientes agentes:

1. Agente Proceso: Modelan los elementos de las unidades de producción. Cada unidad de producción está representada por un Agente Proceso. La composición de un Agente Proceso está basada, por un lado, en una división física del proceso, y por otro lado, en una división funcional de las tareas del agente. Así, un Agente Proceso podría representar desde dispositivos con capacidades de funcionamiento limitadas, como los sensores, actuadores u otros elementos de instrumentación de campo, hasta procesos complejos, tales como una unidad de producción petrolera, una caldera, etc.
2. Agente Control de Proceso: Su tarea fundamental se inspira en la estabilidad y desempeño del proceso controlado. Realiza tareas de entonación, planificación y ejecución de las políticas de control. Su funcionamiento depende los siguientes agentes: Agente Diseño del Control, Agente Ejecutor del Control y Agente Monitor del Desempeño del Control.
3. Agente Diseño del Control: Este agente se encarga de diseñar y/o ajustar planes de control a ejecutar sobre un horizonte de tiempo finito que garanticen el buen desempeño del proceso de producción, en términos de los requerimientos de control (estrategias específicas de control y

[1] Real Time Application Interface for Linux, *https://www.rtai.org/*

parámetros de controlador) y de los requerimientos de procesamiento de control.

4. Agente Ejecutor del Control: Este agente genera las órdenes de control según los lineamientos estipulados en los planes actuales de control y desempeño.
5. Agente Monitor del Desempeño del Control: Este agente se encarga de determinar el desempeño de los planes del control y controladores en ejecución, en términos del cumplimiento de los objetivos planteados en el diseño.

De esta manera, es posible conformar un ambiente de control de procesos basados en agentes. La Figura 11.5 muestra la forma de estructura dicho ambiente.

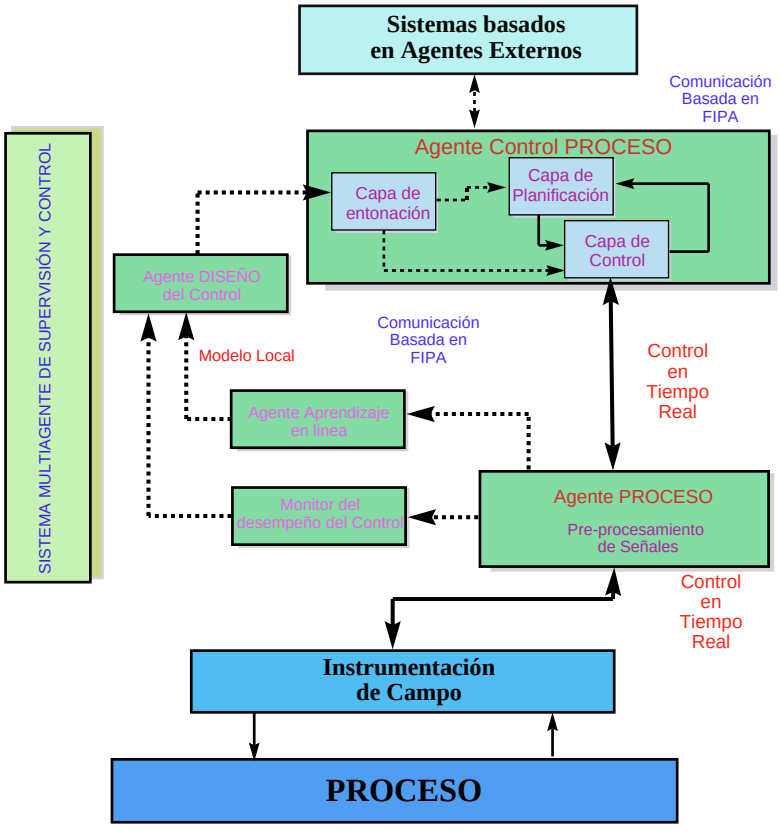

Figura 11.5: Implantación de control con agentes.

A partir de las inter-relaciones entre los diferentes agentes, es fundamental la comunicación entre ellos bajo los siguientes aspectos: la comunicación en tiempo real y la comunicación por eventos. Estos mecanismos de comunicación deben estar soportados por el MGS, descrito en el Capítulo 5.

Tal como ha sido señalado en la Sección 5.2, el Nivel Superior de la arquitectura está conformado por los agentes de aplicación. Allí se definen las comunidades de agentes o SMA en base a sus funcionalidades. En el caso particular de control y supervisión se definen tres agentes: Agente Proceso o Agente de Negocio, Agente Control y Agente Supervisión, podrían definirse otros, tales como agente optimización, aplicaciones especializadas y/o legadas. Estas comunidades interactúan con el Middleware a través de la Nivel Interfaz.

1. **Agente Proceso**: Los Agentes Proceso o Negocio modelan los elementos de las unidades de producción. Cada unidad de producción está representada por un Agente Proceso. La composición de un Agente Proceso está basada, por un lado, en una división física del proceso, y por otro lado, en una división funcional de las tareas del agente. Así, un Agente Proceso podría representar desde dispositivos con capacidades de funcionamiento limitadas, como los sensores, actuadores u otros elementos de instrumentación de campo, hasta procesos complejos, tales como una unidad de producción petrolera, una caldera, etc. Una ventaja que se extrae de esta representación es que, a partir de un modelo del proceso, un Agente Proceso puede ser usado para establecer comparaciones entre el comportamiento real y el comportamiento emulado. Así, éste estaría en capacidad de invocar actividades de manejo de condiciones anormales cuando las comparaciones generan residuos que estén por encima de un nivel mínimo aceptado. Esto con el fin de mejorar el desempeño y minimizar efectos adversos. Dado que el Agente Proceso está muy relacionado con el comportamiento real de los procesos, la transmisión y recepción de información se puede vincular a tareas de tiempo real, tal como se realiza en los sistemas de automatización industrial conocidos: RTU, MTU, SCADA, etc. Es importante destacar que los Agentes Proceso deben ejecutar y solicitar tareas dependiendo de sus roles y funciones. El enviar y recibir solicitudes se vinculan a las comunicaciones asíncronas (dependientes de eventos). El procesar información (conocimiento) es una actividad que puede ser asíncrona o síncrona, dependiendo de las exigencias del proceso y del rol del agente. En las Figuras 11.6 y 11.7 se presentan el diagrama de caso de uso y el diagrama de actividades del Agente Proceso.
   El caso de uso Gestionar Información se encarga de la gestión y procesamiento de la información, asociada y proveniente de los procesos de campo, así como también la suministrada por otros agentes. Los actores con los que interactúa son, en este caso, otros Agentes Proceso y de Aplicaciones. Las tareas básicas de este agente son: recibir información, recibir solicitud, transmitir información, solicitar servicio de procesamiento. Las

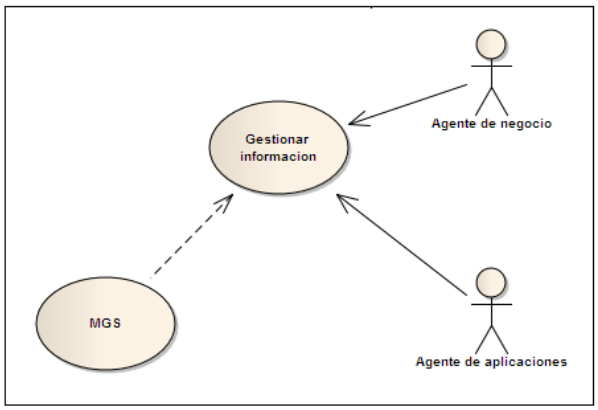

Figura 11.6: Diagrama de caso de uso del Agente Proceso.

interacciones de este agente con otras comunidades de agentes se muestran en la Figura 11.8.

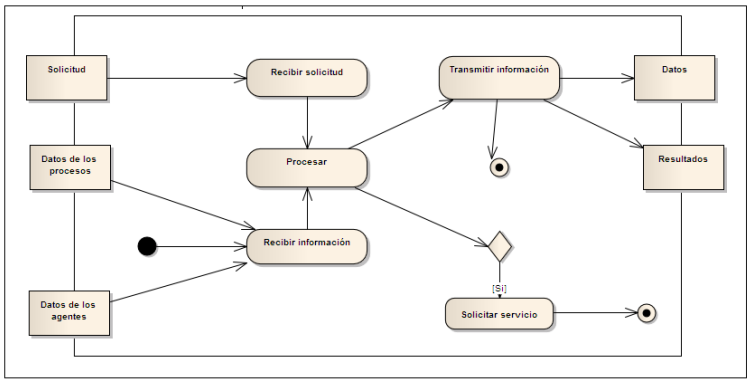

Figura 11.7: Diagrama de actividades del Agente Proceso.

2. **Agente Control de Proceso**: Como se sabe, una de las tareas fundamentales de cualquier sistema de automatización industrial es el control de procesos. Ella se inspira, de igual manera, sobre la base de satisfacer requerimientos de seguridad (estabilidad de los procesos) y de productividad (desempeño de los procesos). Para implantar sistemas de control es necesario disponer de la información de campo, que en el caso basado en agentes, se puede realizar a través del Agente Proceso. El Agente Control se diseña como un SMA conformado por los siguientes agentes:

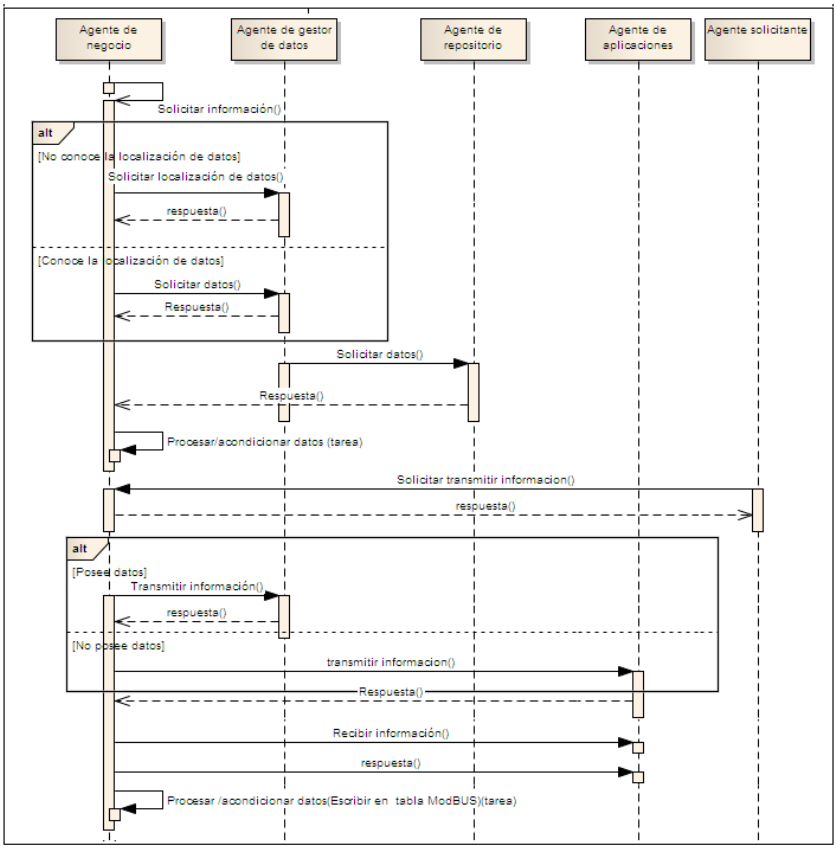

Figura 11.8: Diagrama de interacción del Agente Proceso.

Agente Diseño del Control, *Agente Ejecutor del Control* y *Agente Monitor del Desempeño del Control*.

a) Agente Diseño del Control: Este agente se encarga de diseñar y/o ajustar planes de control a ejecutar sobre un horizonte de tiempo finito que garanticen el buen desempeño del proceso de producción, en términos de los requerimientos de control (estrategias específicas de control y parámetros de controlador) y de los requerimientos de procesamiento de control. El diagrama de caso de uso para este agente se muestra la Figura 11.9.
Las actividades orientadas a definir las tareas de control se estructuran de acuerdo al diagrama de actividades que se muestra en la Figura 11.10. Como se puede observar, se definen dos tareas: *Planificar control* y *Ajustar controlador*. El diagrama de interacción para la actividad Planificar control se muestra en la Figura 11.11.

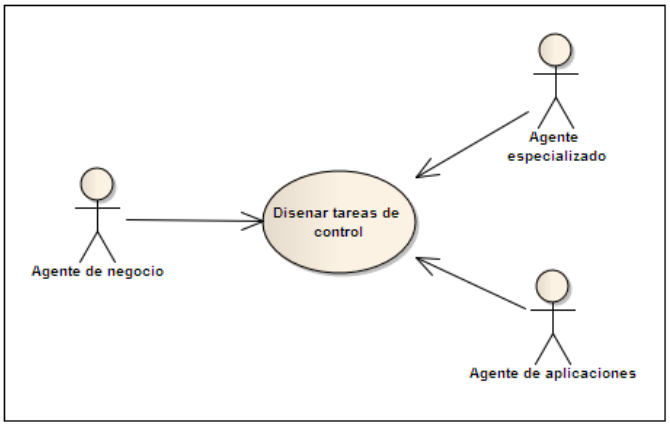

Figura 11.9: Diagrama de caso de uso del Agente Diseño del Control.

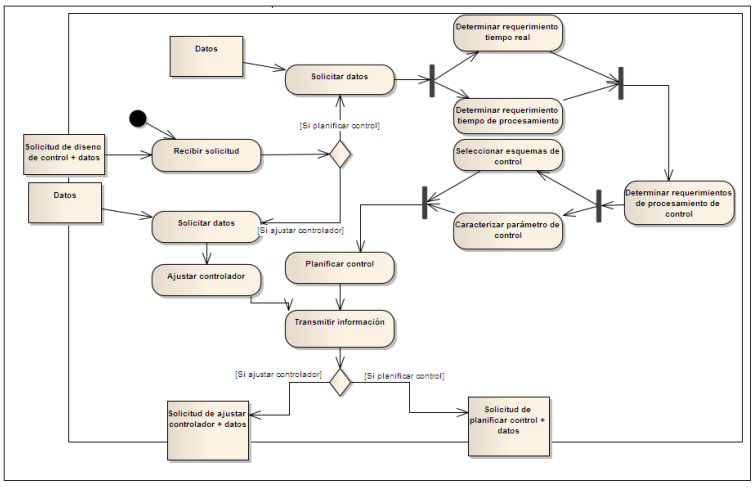

Figura 11.10: Diagrama de actividades del Agente Diseño del Control.

Otra tarea importante de este agente es ajustar los parámetros del controlador en ejecución sobre un horizonte de tiempo finito, que garantice el buen desempeño del proceso de producción en términos de los requerimientos de control y de procesamiento de control. Para ello se construye un diagrama de interacción, el cual se muestra en la Figura 11.12.

b) Agente Ejecutor del Control: Este agente genera las órdenes de control según los lineamientos estipulados en los planes actuales de control y esquemas usados (todo esto es especificado por el agente descrito previamente). Para esto, el agente recibe un requerimiento de control,

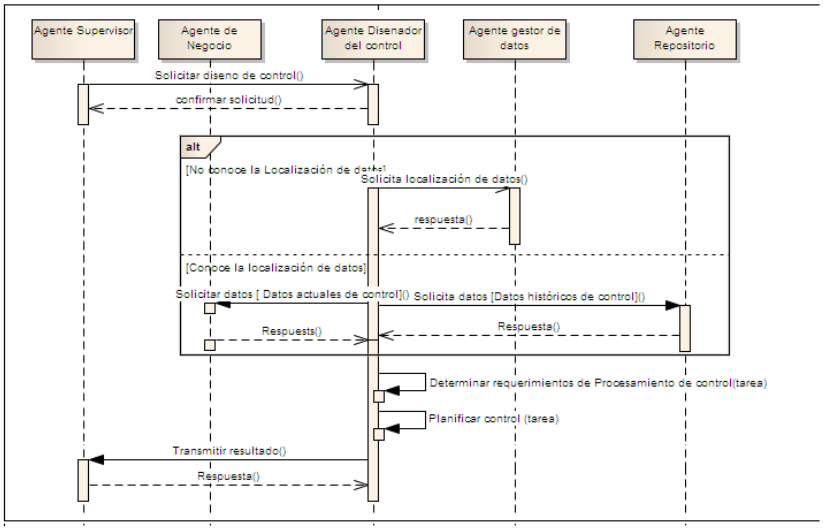

Figura 11.11: Diagrama de interacción del Agente Diseño del Control.

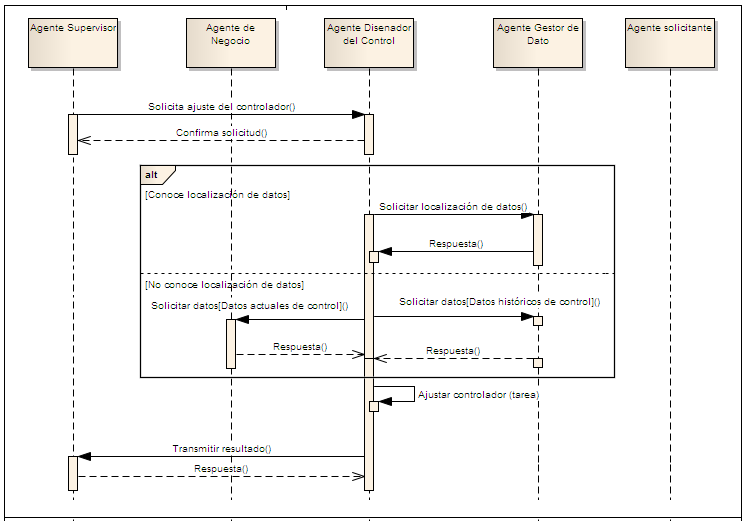

Figura 11.12: Diagrama de interacción para el ajuste del control.

y en base a los datos y conocimiento disponibles genera órdenes de control. El caso de uso para este agente se muestra en la Figura 11.13. Las actividades orientadas a definir las tareas de control se estructuran de acuerdo al diagrama de actividades que se muestra en la Figura 11.14, donde se observa que el rol fundamental es *generar acciones de*

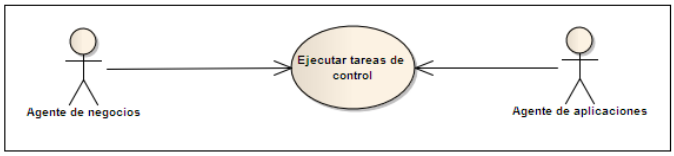

Figura 11.13: Diagrama de caso de uso del Agente Ejecutor del Control.

control. El diagrama de interacción para este agente se muestra en la Figura 11.15.

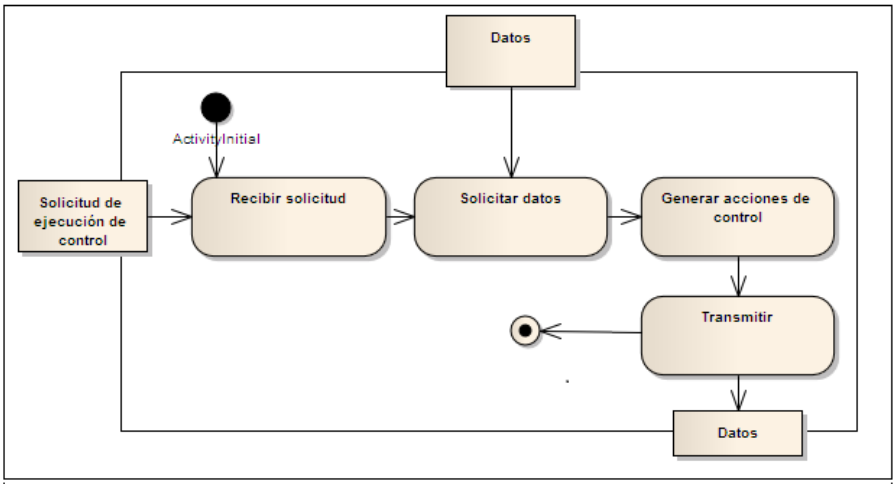

Figura 11.14: Diagrama de interacción del Agente Ejecutor del Control.

c) Agente Evaluador del Control: Este agente se encarga de determinar el desempeño de los planes del control y controladores en ejecución, en términos del cumplimiento de los objetivos planteados en el diseño. El caso de uso para este agente se muestra en la Figura 11.16.
Las actividades orientadas a definir las tareas de control se estructuran de acuerdo al diagrama de actividades mostrado en la Figura 11.17.
Tal como se puede observar en la Figura 11.17, las tareas básicas del agente son: *Evaluar desempeño del plan de control* y *Evaluar desempeño del controlador*, cuyos diagramas de interacción se presentan en las Figuras 11.18 y 11.19, respectivamente.

3. **Agente Supervisor**:
Las aplicaciones de supervisión y mantenimiento son parte fundamental en los procesos de automatización; su adecuada integración con los sistemas de control permite alcanzar los objetivos con altos desempeños de

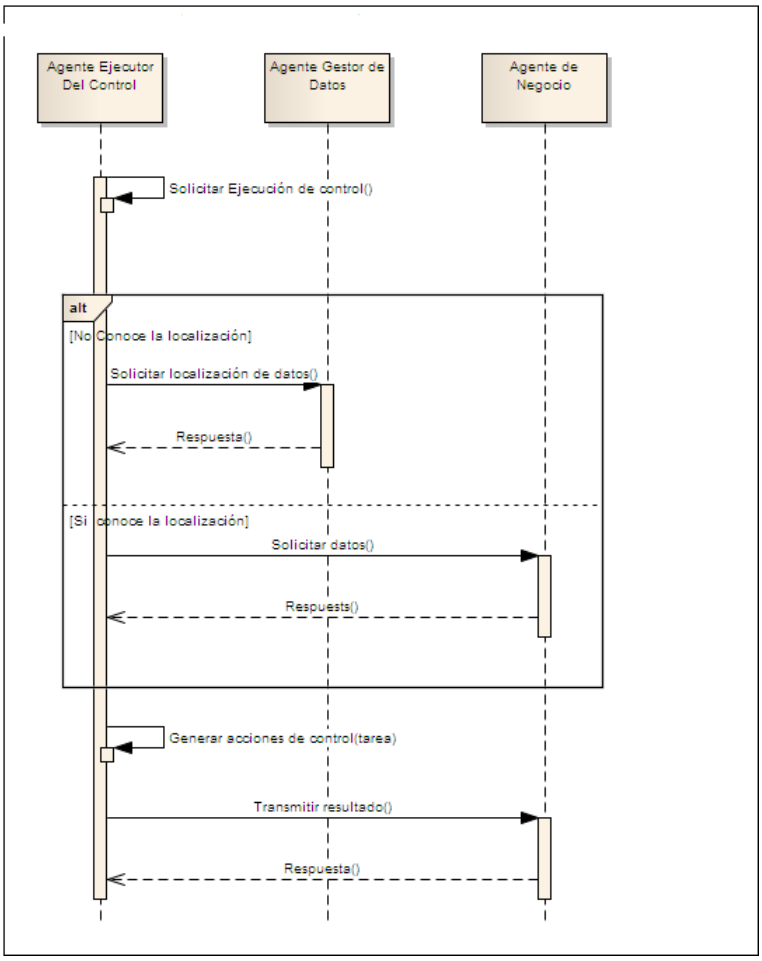

Figura 11.15: Diagrama de actividades del Agente Ejecutor del Control.

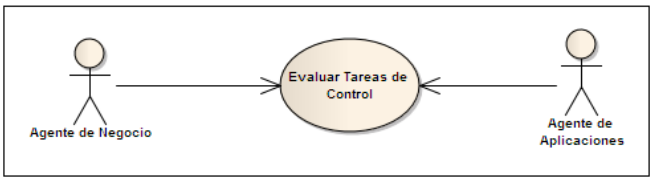

Figura 11.16: Diagrama de caso de uso del Agente Monitor del Desempeño del Control.

los procesos de producción. La descripción del Agente Supervisor en el marco de esta arquitectura de control se describe en la Sección 12.2.

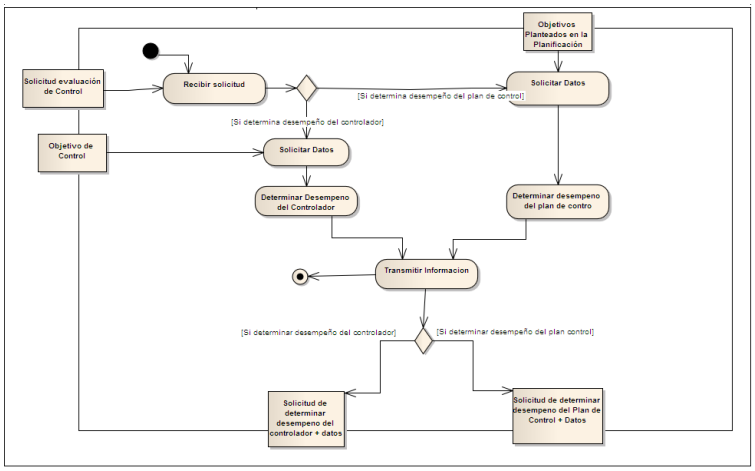

Figura 11.17: Diagrama de actividades del Agente Monitor del Desempeño del Control.

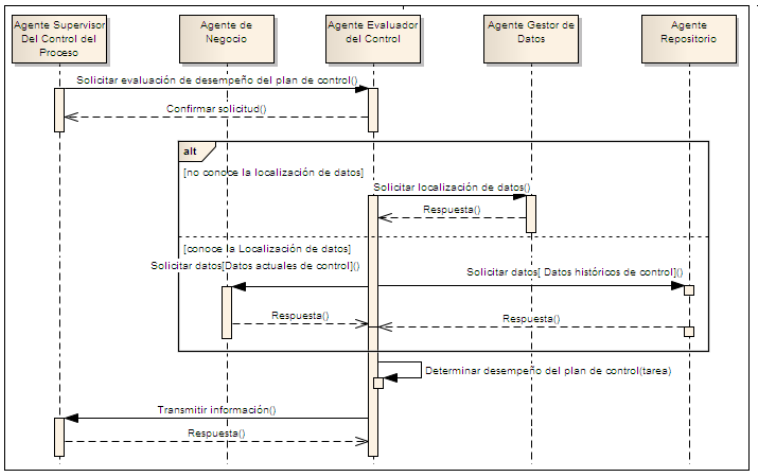

Figura 11.18: Diagrama de interacción para evaluar el desempeño del plan de control.

Para la implantación en las redes de control, estos agentes se conciben como objetos de software cuyo diseño usa las herramientas para el desarrollo de sistema a objetos, teniendo como punto de partida la definición de los diagramas de clases. Una vez definidos los diagramas de clases, se procede con la definición del universo de clases y de los tipos abstractos de datos (TAD), seguido de la especificación formal de la clase, métodos y operaciones de dicha clase. De esta manera se conforma la plataforma para la implementación del SMA, que se constituyen en un modelo de agentes de sistemas inteligentes de control y supervisión para plataformas de auto-

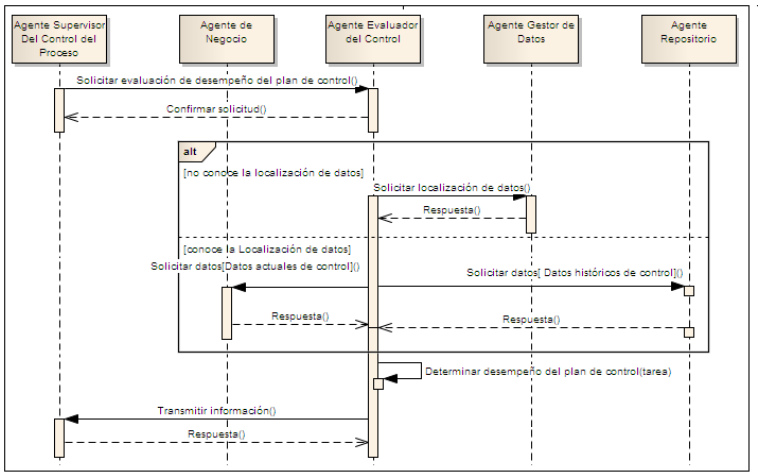

Figura 11.19: Diagrama de interacción para evaluar el desempeño del controlador.

matización industrial; el cual permite coordinar, ejecutar y evaluar tareas de control y supervisión, necesarias para el procesamiento de la información del proceso y para la toma de decisiones. El SMA se organiza en una arquitectura integrada/distribuida en el contexto de la automatización de procesos.

Referencias

1. J. Aguilar, M. Cerrada, and F. Hidrobo. A methodology to specify multiagent systems. *Lecture Notes in Artificial Intelligence, Springer-Verlag*, 4496:92–101, 2007.
2. J. Aguilar, M. Cerrada, G. Mousalli, and F. Rivas. Masina: Una metodología para especificar sistemas multiagentes. *Publicación ULA*, 2004.
3. J. Aguilar, M. Cerrada, G. Mousalli, and F. Rivas. A multiagent model for intelligent distributed control systems. *Lecture Notes in Artificial Intelligence, Springer-Verlag*, 3681:191–197, 2005.
4. J. Aguilar, A. Rios, F. Rivas, O. Teran, L. Leon, and N. Perez. Definición de dominios y paradigmas en una arquitectura de automatización industrial. Technical Report 05-04, Fundacite-Mérida, Mérida, 2004.
5. Joseph Aguilar-Martin. *Inteligencia Artificial para la Supervisión de Procesos Industriales*. Consejo de Publicaciones, ULA, 2007.
6. Alexei Bratoukhine, Yoseba Penya, and Thilo Sauter. Intelligent software agents in plant automation. Technical report, Vienna University Of Technology, Wien Austria, 2002.
7. C. Bravo, J. Aguilar, F. Rivas, and M. Cerrada. Design of an architecture for industrial automation based on multi-agents systems. In *Proceeding of the 16th IFAC World Congress*, Prague, 2005.

8. F. Hidrobo, A. Ríos-Bolívar, J. Aguilar, and L. León. An architecture for industrial automation based on intelligent agents. *WSEAS Transaction on Computers*, 4(12):1808–1815, 2005.
9. Nicholas Jennings and Stefan Bussmann. Agent-based control systems. *IEEE Control Systems Magazine*, June:61–73, 2003.
10. Axel Klostermeyer. Revolutionising plant automation – the pabadis approach - white paper. Technical Report IST-1999-60016, PABADIS: Information Society Technology, Germany, 2003.
11. OMG. Unified Modeling Language specification, 2003. Version 2.0, June 2003 via `http://www.omg.org`.
12. Jim Pinto. Instrumentation & control on the frontiers of a new millennium. *Journal Instruments & Control Systems*, 1(2):78–90, 2000.
13. A. Ríos-Bolívar, M. Cerrada, F. Hidrobo, J. Aguilar, and J. Durán. Towards control and supervision systems based on intelligent agents. In *Proc. of 2nd WSEAS International Conference on Dynamical Systems and Control*, pages 36–41, Bucharest, Romania, 2006.
14. A. Ríos-Bolívar, F. Hidrobo, M. Cerrada, and J. Aguilar. Control and supervision system development with intelligent agents. *WSEAS Transactions on Systems*, 6(1):141–148, 2007.
15. I. Seilonen, T. Pirttioja, P. Appelqvist, A. Halme, and K. Koskinen. An approach to process automation based on cooperating subprocess agents. In *1st Int. Conf. on Applications of Holonic and Multi-Agent Systems (HoloMAS 2003)*, Prague, Czech Republic, September 2003.
16. Ilkka Seilonen, Teppo Pirttioja, and Pekka Appelqvist. Agent technology and process automation. Technical report, Helsinki University of Technology, 2003.
17. T. Wagner. An agent-oriented approach to industrial automation systems, 2002.
18. Thomas Wagner. Applying agents for engineering of industrial automation systems. In M. Schillo *et al.*, *editor*, MATES 2003, *page 62–73. Springer-Verlag*, *Berlin Heidelberg*, *2003.*
19. *Lingfeng Wang and Kay-Chen Tan.* Modern Industrial Automation Software Design. *IEEE Press*, *2006.*
20. *G. Weiss.* Multiagent Systems. *The MIT Press*, *1999.*

Capítulo 12
Supervisión de Procesos Basada en Agentes

Resumen: En este capítulo se aborda, de manera descriptiva, el desarrollo de SMA para la supervisión de procesos industriales.

12.1. Introducción

El crecimiento de la demanda de productos que satisfagan las exigencias de calidad del mercado en tiempos adecuados, ha requerido que los procesos de producción mantengan altos niveles de eficiencia y confiabilidad. Para ello, la automatización de procesos se ha visto como una necesidad para la mejora de la eficiencia de la industria, incrementando a su vez la complejidad de los procesos, disminuyendo la confiabilidad de todo el sistema y en general, de los niveles de seguridad.

Con el fin de satisfacer los requerimientos de confiabilidad y seguridad en sistemas complejos, es necesario disponer, además de operadores altamente capacitados con experiencia y habilidades, de soluciones para enfrentar estos problemas de capacidad, a la vez de soportar no sólo las tareas de los operadores sino las tareas en los otros niveles de la pirámide de automatización. Estas soluciones es lo que se conoce como *sistemas supervisores*, cuya meta principal es incrementar la confiabilidad de los procesos usando conceptos de control automático con funcionalidades eficientes de las interfaces hombre-máquina.

La supervision es entendida, entonces, como el control y operaciones de monitoreo de un sistema, incluyendo todo aquello que asegure confiabilidad y seguridad. Un sistema de supervisión es un enlace que permite la cooperación entre el control automático y los niveles superiores de la pirámide de automatización. El rol de un sistema de supervisión es coleccionar datos

desde los procesos y tomar decisiones sobre nuevas direcciones que pueden tomar dichos procesos, lo cual está soportado por funciones de [17]:

- Planificación y programación a alto nivel, que permita decidir sobre estrategias y procedimientos ante diferentes situaciones (normales y anormales).
- Observación (monitoreo) del proceso y ajustes menores en línea.
- Diagnóstico de fallas.
- Intervención sobre el control cuando se alcanzan ciertos estados, para lograr nuevos objetivos de control.
- Aprendizaje a través del uso de las experiencias pasadas para mejorar los procesos de conocimiento y actualizar los planes globales de la empresa.

En la comunidad de control y automatización, los sistemas de supervisión tienen diferentes interpretaciones, lo que que finalmente ha permitido que se desarrollen aplicaciones de supervisión orientadas a diferentes objetivos y estructura del sistema. Por un lado, las supervisión orientada a tareas de control donde, en general, los sistemas supervisorios son concebidos como generadores de *puntos de operación* de alto nivel para controladores a bajo nivel, a partir del monitoreo de su desempeño y, por otro lado, la supervisión orientada al diagnóstico de fallas que ha tenido especial atención como tarea de un sistema de supervisión. Cualquiera sea la orientación o interpretación, los objetivos se reducen a:

1. Permitir el manejo del proceso.
2. Identificar en tiempo real la tendencia del proceso.
3. Gestionar los datos para realizar análisis de históricos, que permitirán mejorar la eficiencia del proceso, mejorar la confiabilidad, optimizar los recursos y prevenir la degradación del proceso.

Como quiera que sea su orientación, un sistema de supervisión es un sistema que evalúa si un comportamiento satisface un conjunto de especificaciones de desempeño, diagnostica las causas, planifica acciones y ejecuta las acciones planificadas.

Alcanzar los objetivos de supervisión en sistemas complejos y altamente automatizados es una tarea compleja que requiere de la incorporación de mecanismos que permitan:

- Autonomía en el proceso de la toma de decisiones.
- Capacidades adaptativas, según los cambios que emerjan naturalmente de la dinámica propia de los procesos internos y externos.
- Capacidades de aprendizaje, de acuerdo a la experiencia adquirida.
- Posibilidades de acceso al conocimiento e información distribuida localmente.

Estos requerimientos, en resumen, hacen que los sistemas complejos tengan características *inteligentes*. La incorporación de estas habilidades a los

sistemas de supervisión para lograr los objetivos propuestos es lo que define como *Supervisión Inteligente* de procesos.

Mucho se ha discutido en la comunidad de automatización lo que significa la incorporación de inteligencia en los procesos, que va desde simplemente usar técnicas inteligentes tales como lógica difusa, redes neuronales ó algoritmos genéticos, para realizar algunas tareas de supervisión [14, 4], hasta las propuestas que presentan arquitecturas y modelos para concebir sistemas de supervisión inteligente, como un todo [12, 8, 19, 13, 16]. Es aquí donde el uso de agentes inteligentes se presenta como un alternativa para dotar a los sistemas de supervisión de propiedades estructuralmente inteligentes y emergentes. El paradigma de agentes inteligentes es una manera natural de descomposición de sistemas y una alternativa razonable para implantar las funcionalidades de la supervisión inteligente, donde la reconfigurabilidad y la flexibilidad, junto con la inteligencia, son aspectos importantes a satisfacer [1].

En general, las aplicaciones de agentes en supervisión de procesos se diseñan a partir de los sistemas reactivos ya existentes, y a partir de allí se modelan los agentes para permitir la comunicación entre dichos sistemas y extraer información para ir creando su propio conocimiento. Con el fin de diseñar sistemas de supervisión concebidos desde su inicios como sistemas inteligentes, los autores han propuesto esquemas para el desarrollo e implantación de SMA para la supervisión de procesos industriales, dotados de las capacidades de autonomía, interacción colaboración, adaptabilidad, entre otras, que permiten la operación segura y óptima en base a los objetivos de producción.

En este capítulo se presentan diferentes enfoques de sistemas de supervisión basada en agentes, que abordan tanto la orientación hacia el control supervisorio, como la orientación hacia el manejo de fallas.

12.2. Un agente de supervisión para el control y la confiabilidad

En la Sección 10.8, los autores han propuesto una arquitectura que permite la integración de aplicaciones de automatización sobre plataformas distribuidas basadas en SMA, que apunta hacia la integración del conocimiento, información y datos distribuidos con el fin de proveer mecanismos que soporten la toma de decisiones. En dicha arquitectura, la capa superior refiere a una comunidad de agentes, llamados Agentes de Aplicaciones, que implementan funciones específicas de automatización industrial, entre ellas la de supervisión.

Bajo esta propuesta, el Agente Supervisor se concibe como un SMA compuesto por tres agentes: Agente Supervisor de Control, Agente Supervisor de Confiabilidad y Agente Supervisor de Tareas, ver Figura 12.1.

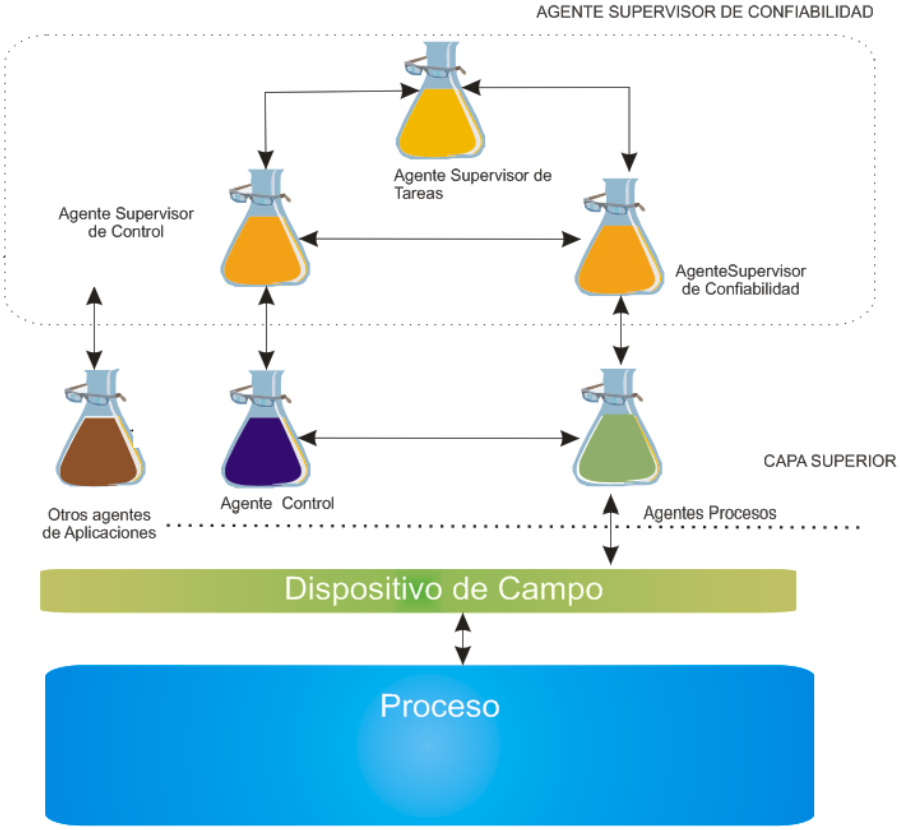

Figura 12.1: Arquitectura de referencia par la supervisión de procesos basada en agentes

A continuación se describen dichos agentes, los cuales han sido especificados usando la metodología MASINA, propuesta por los autores y presentada en detalle el Capítulo 3. Se presentan principalmente los diagramas de actividades de cada agentes, siendo esta actividades vistas, en general, como servicios ofrecidos a otros agentes de aplicaciones. Los diagramas de interacción permiten visualizar las comunicaciones necesarias entre agentes para cumplir con las actividades.

1. **Agente Supervisor de Control**: Este agente supervisa el desempeño del proceso de producción a partir de los datos y conocimientos relacionados con las tareas de control y de manejo de los factores de producción, generando índices operacionales que permitan medir tal desempeño. Como consecuencia, este agente puede solicitar el ajuste de los planes de control actuales ó de los parámetros del controlador para compensar salidas

erróneas del proceso. El caso de uso de este agente se muestra en la Figura 12.2.

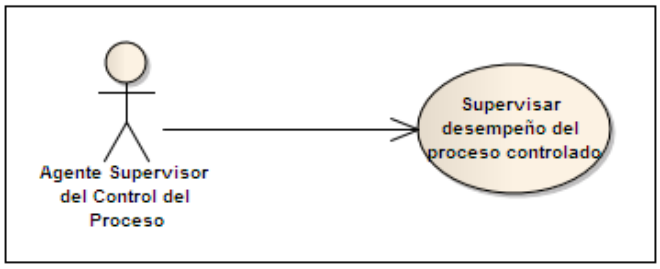

Figura 12.2: Diagrama de caso de uso del Agente Supervisor de Control.

La Figura 12.3 muestra el diagrama de actividades de este agente. Una de las actividades fundamentales señaladas en dicho diagrama es la actividad *Estimar índices operacionales*. Esta actividad (servicio) se activa cuando el agente recibe la solicitud de supervisión, seguidamente solicita datos de la gestión de control y mantenimiento, y aplica métodos que resulten en índices que permitan medir el desempeño del proceso de producción. El diagrama de interacción, para llevar a cabo esta actividad de estimación de los índices operacionales, se muestra en la Figura 12.4, la cual resalta la interacción de este agente con los otros agentes de sistema.

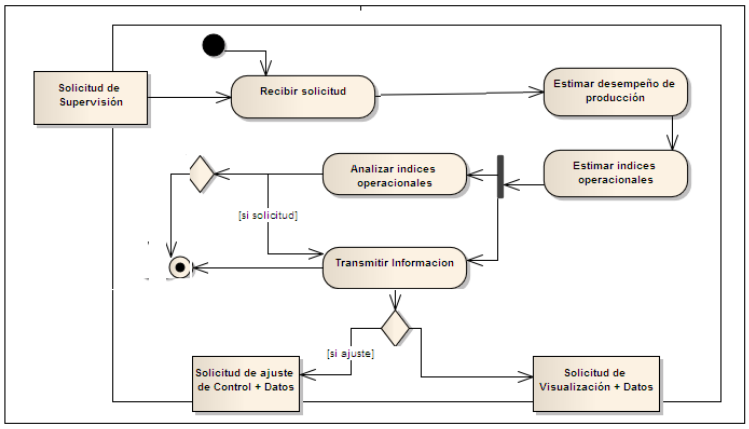

Figura 12.3: Diagrama de actividades del Agente Supervisor de Control.

Otras actividades que realiza el Agente Supervisor de Control son:

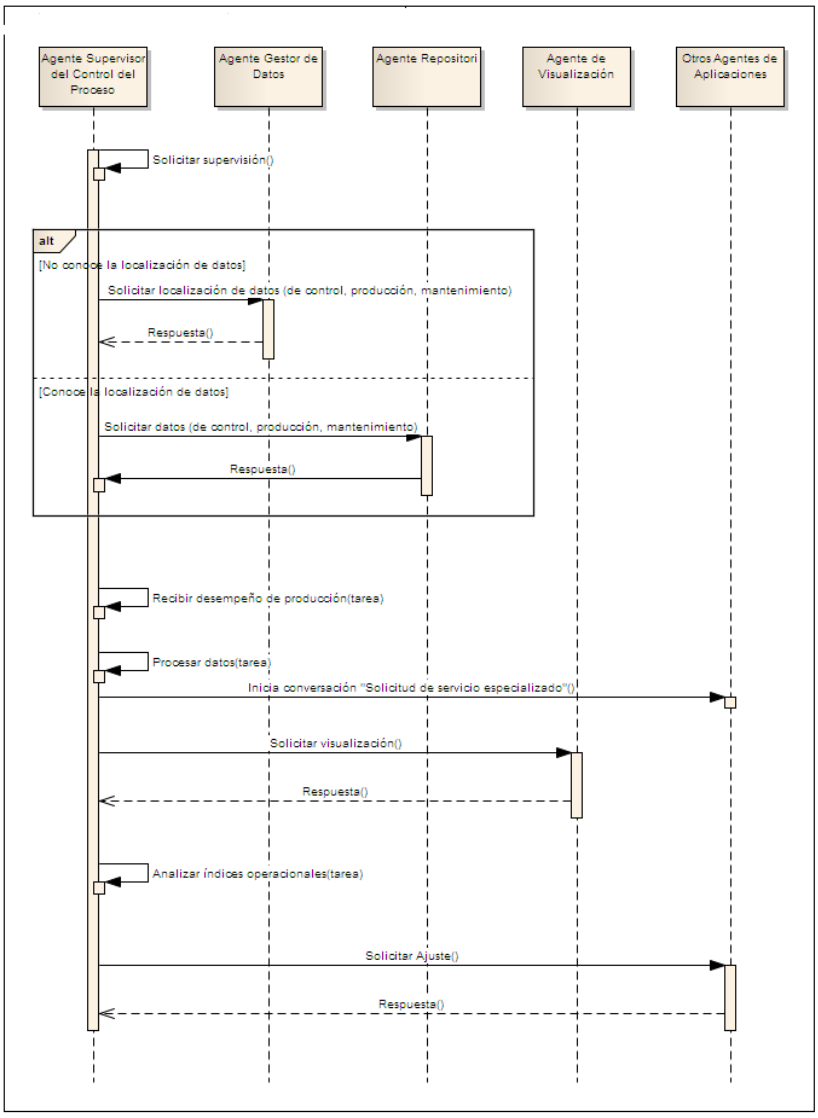

Figura 12.4: Diagrama de interacción para estimar índices operacionales.

- **Estimar desempeño de producción**: el agente recibe la solicitud de supervisión, solicita datos de la gestión de producción y aplica métodos que resulten en índices operacionales que permitan medir el desempeño del proceso de producción.
- **Analizar índices operacionales**: analiza los resultados de los índices operacionales, resultando en la transmisión, a los agentes pertinentes,

de solicitudes de ajuste de planes y/o parámetros de control, producción y/o mantenimiento, que permitan mejorar el desempeño del proceso de producción.

2. **Agente Supervisor de Confiabilidad**: Este agente supervisa las actividades relacionadas con la confiabilidad del proceso. Para ello, debe tener acceso a datos históricos y de tiempo real, debe conocer los planes actuales de mantenimiento, así como los esquemas de manejo de fallas que se están usando. Este agente puede solicitar, en consecuencia, la aplicación de tareas específicas de mantenimiento y/o cambios en los planes actuales de mantenimiento.
   El caso de uso del agente Supervisor de Confiabilidad se muestra en la Figura 12.5 y el diagrama de actividades del mismo en la Figura 12.6. Una de las actividades fundamentales es la de *Manejar Fallas*. En este caso, el agente puede aplicar métodos de detección, diagnóstico y predicción de fallas, según un plan de mantenimiento preventivo o por evento. El diagrama de interacción de esta actividad se muestra en la Figura 12.7, la cual resalta la interacción de este agente con los otros agentes de sistema, a fin de cumplir con dicho servicio.

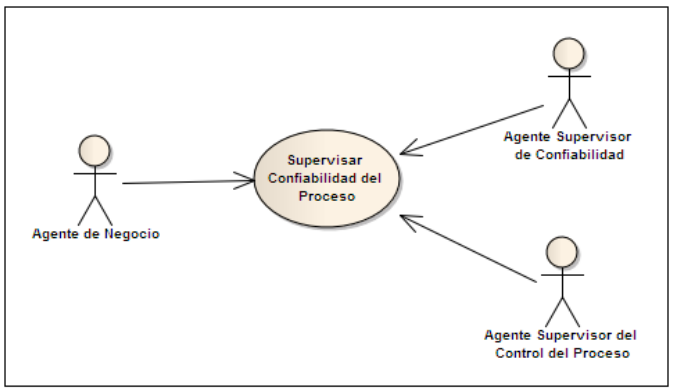

Figura 12.5: Diagrama de caso de uso del Agente Supervisor de Confiabilidad.

Otros servicios (actividades) ofrecidos por este agente son:

- **Monitorear proceso**: el agente monitorea las variables representativas del proceso, analiza sus tendencias y genera alarmas, si es necesario. Esta información es utilizada con fines de control ó de aplicación de tareas de mantenimiento.
- **Monitorear mantenimiento**: estima índices de confiabilidad del proceso en términos del desempeño de los planes de mantenimiento, disponibilidad de repuestos, costos de mantenimiento, correcta ejecución de los planes y tiempo asociado.

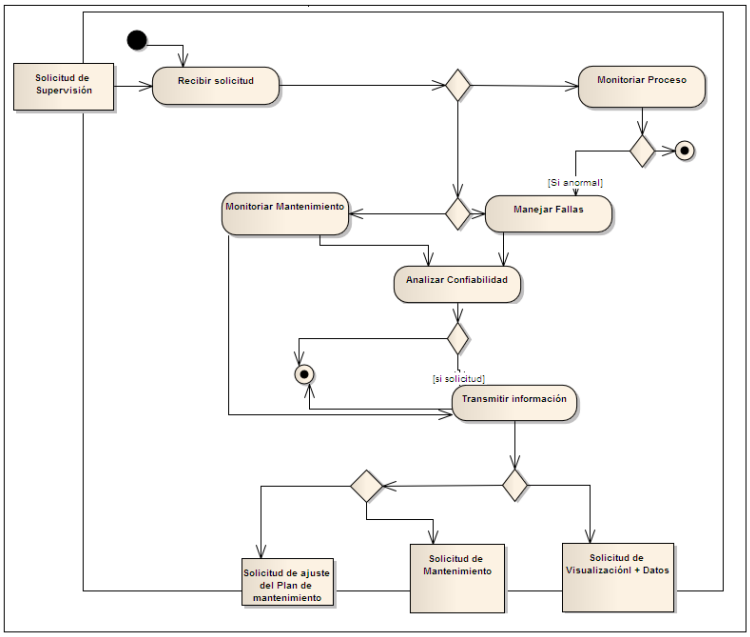

Figura 12.6: Diagrama de actividades del Agente Supervisor de la Confiabilidad.

- **Analizar confiabilidad**: el agente evalúa la confiabilidad del proceso en términos del desempeño de los planes de mantenimiento, la coordinación de la ejecución de los mismos, índices de confiabilidad, entre otros. La información obtenida es utilizada con fines de planificación táctica y operacional.

3. **Agente Supervisor de Tareas**: Una de las características fundamentales de los SMA es la posibilidad de desarrollar mecanismos para autoevaluarse y aprender de experiencias. En el contexto de control y supervisión, el Agente Supervisor de Tareas ofrece dos servicios fundamentales orientados a desarrollar esta característica: *Evaluación de la toma de decisiones* y *Evaluación del desempeño de las tareas*. Para el primer servicio, el agente debe revisar los datos históricos relacionados con los índices operacionales y de confiabilidad, y debe aplicar métodos para evaluar como han sido las consecuencias derivadas de la toma de decisiones con respecto a los cambios solicitados por los agentes en los planes de control ó mantenimiento. Para el segundo servicio, el agente evalúa el desempeño de las aplicaciones de supervisión basadas en las propiedades de los servicios solicitados (confiabilidad del servicio, calidad del servicio, etc.). El diagrama de actividades de este agente para el servicio *Evaluación de la toma de decisiones* se muestra en la Figura 12.8

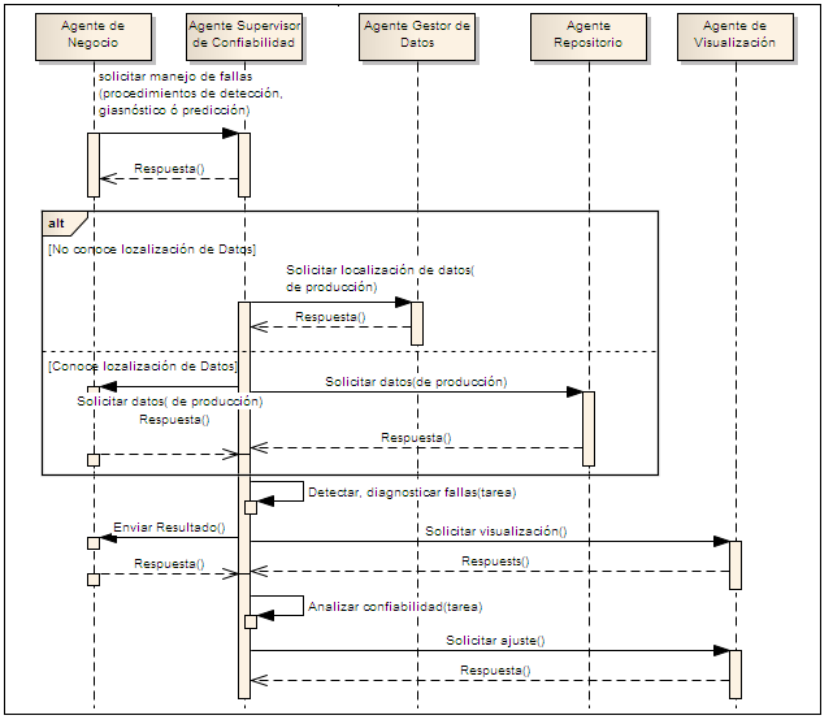

Figura 12.7: Diagrama de interacción para manejar fallas.

12.3. Agente para el manejo de fallas

Las tareas de mantenimiento que se ejecutan en procesos de producción han alcanzado los primeros lugares de importancia a fin de elevar los niveles de confiabilidad y disponibilidad de la planta. En los ambientes automatizados, las fallas impactan el servicio y la calidad del producto, en consecuencia, la gestión de mantenimiento debe asegurar el cumplimiento de la función de la planta, por consiguiente, las principales tareas deben ser preventivas apuntando a la detección, el diagnóstico y la predicción (las tareas sobre condición), seguidas por las tareas planificadas (las tareas sobre tiempo), y por último, se consideran las tareas de reparación que restauran la funcionalidad de equipo (tareas correctivas). En el desarrollo de un programa de mantenimiento debe estipularse qué tareas se necesitan hacer y cuándo se deben hacer, así como garantizar la disponibilidad de recursos que permitan ejecutar dichas tareas según los cronogramas estipulados. Todo esto es guiado por el conocimiento extenso sobre mecanismos de falla, las experiencias y la información proporcionadas por los diseñadores de equipos, los niveles de inventarios, los requerimientos de producción, en-

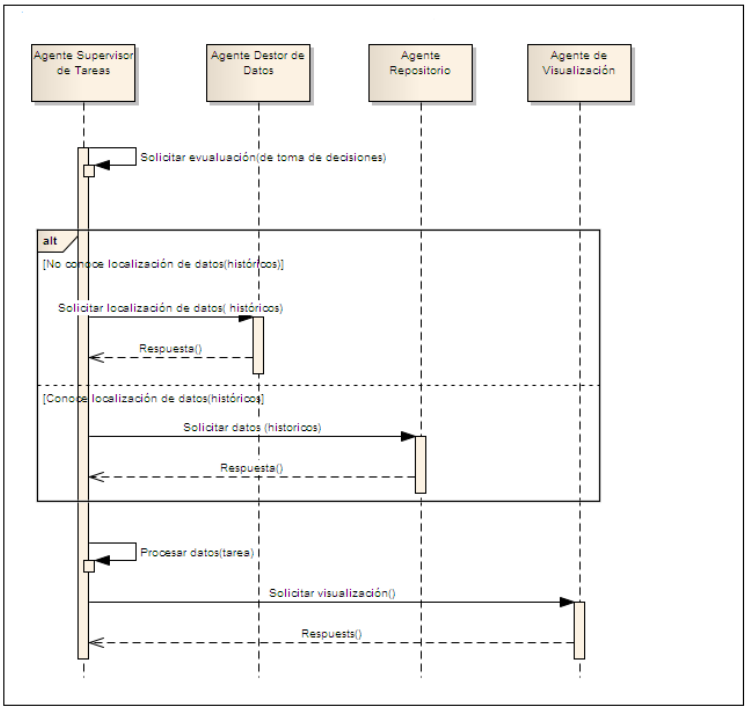

Figura 12.8: Diagrama de interacción para la evaluación de la toma de decisiones.

tre otros. Así, los datos e información que permiten eficazmente manejar la tarea de detección-diagnóstico-predicción, debe estar disponible.

Teniendo en cuenta los aspectos anteriores, un programa de mantenimiento debe considerar los siguientes aspectos [11, 18]:

- Mecanismos para la potencial detección de fallas, basado en el desarrollo de nuevas tecnologías en el mantenimiento preventivo.
- Tecnologías para análisis de falla, el cual están soportadas por mecanismos que permiten reportar las fallas y analizarlas (la causa raíz de las fallas).
- El manejo de la información que apunta al desarrollo de sistemas de información que permiten guardar el registro de mantenimiento correctivo sobre el equipo, el registro de análisis de falla (histórico de detección, diagnóstico y predicción), las alarmas, y los costos de mantenimiento.
- El sistema de ayuda a la toma de decisiones, que usan la información disponible para proponer nuevos programa de mantenimiento acordes a los objetivos actuales de la empresa.

Una revisión acerca del impacto de la gestión de mantenimiento en el desempeño de las industrias se presenta con detalle en [5] y referencias in-

ternas. Fundamentalmente, se resalta la necesidad de integrar las estrategias de mantenimiento con los aspectos operacionales apuntando hacia una estrategia integrada que considera planificación, programación y recursos. En [9] se presentan diferentes enfoques para el desarrollo de sistemas comerciales de mantenimiento integrado. Principalmente, estos sistemas están compuestos por módulos que soportan la planificación del mantenimiento, pero no ofrecen una solución integrada en los niveles de supervisión. Desde un punto de vista distribuido, el concepto de *e-Maintenance* nace con el fin de integrar el conocimiento, la información y los datos distribuidos [6, 16, 10]

Considerando las propiedades y características de los SMA, éstos se perfilan como una vía para la concepción de modelos integrados de gestión de mantenimiento [15, 7]. Los SMA permiten implementar procesos de cooperación, negociación y coordinación entre los principales actores que tienen el conocimiento, como lo son la gestión de producción, el control y el mantenimiento. En esta sección presentamos un modelo de agentes para la gestión de mantenimiento desarrollado por los autores en [5]. El modelo nace de una arquitectura de referencia que permite incorporar un SMF en el nivel supervisorio del modelo piramidal de la Figura 10.19. El sistema para el manejo de fallas, desde el punto de vista de su arquitectura, se muestra en la Figura 12.9:

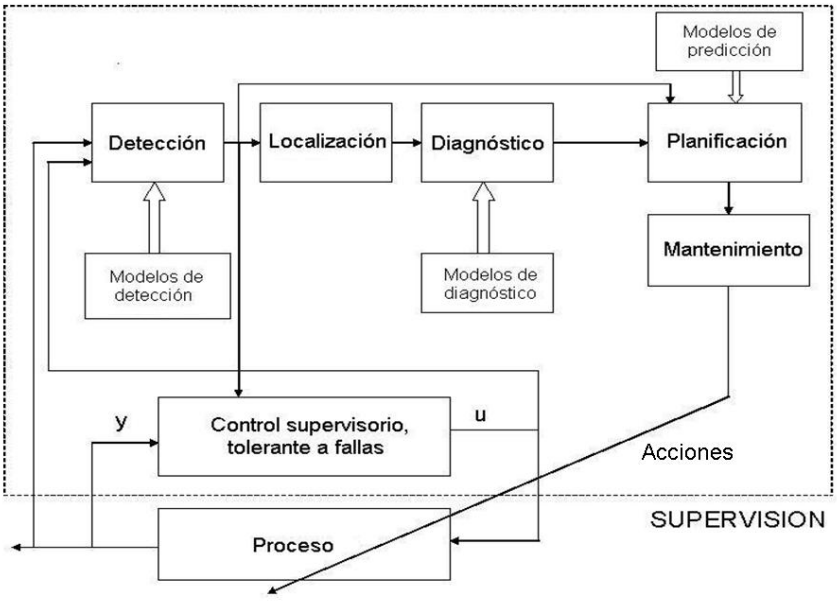

Figura 12.9: Arquitectura de referencia para manejo de fallas

El SMF debe permitir la ejecución de actividades relativas a:

- La detección confinamiento y detección de fallas abruptas o incipientes.
- La predicción de fallas funcionales a partir de fallas incipientes.
- La planificación de tareas de mantenimiento preventivo.
- La ejecución del plan de mantenimiento que también involucra el ejecutar planes de contingencia en el caso de fallas abruptas o inesperadas (mantenimiento correctivo).

Un modelo simplificado de la Figura 12.9, se muestra en la Figura 12.10, donde se resaltan los dos bloques funcionales principales. Un bloque realiza las tareas de Monitoreo y Análisis de Fallas (MAF) y otro bloque realiza las tareas para la Gestión del Mantenimiento (GM). El SMF y la gerencia de mantenimiento son bloques entrelazados a través de índices de productividad, recursos humanos y financieros, almacén, etc. Por otro lado, el SMF también actúa con el proceso tolerante a fallas, siendo éste el receptor final de las tareas de detección, localización, diagnóstico y decisión.

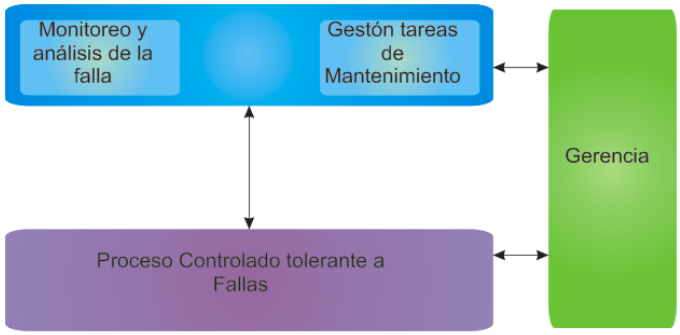

Figura 12.10: Bloques funcionales del SMF

A partir del modelo de referencia de la Figura 12.9, y siguiendo la metodología MASINA, se determinan los actores y casos de uso asociados, presentados en la Tabla 12.1. La identificación de los actores permite, posteriormente, la definición de los agentes que pueden asociarse a un actor o conjunto de actores, ó de modo inverso asociar a un actor un conjunto de agentes.

De igual manera, se determina el flujo de actividades que se debe realizar para continuamente ejecutar los planes de mantenimiento que pueden ir desde la aplicación de tareas de detección y diagnóstico (tareas de mantenimiento sobre condición), hasta la reparación de equipos (tareas de mantenimiento sobre condición). Estas actividades serían consideradas como *acciones* a ejecutar sobre el proceso. Si las *acciones* no son ejecutadas, el SMF debe redefinir el plan de mantenimiento. Si una falla abrupta ocurre

(desviación funcional significativa) el SMF debe emprender una tareas de detección urgente, esto es, no contemplada en el plan de mantenimiento en ejecución. La Figura 12.11 muestra el diagrama de actividades del SMF.

Actor	**Descripción**	**Caso de Uso**
Detección	Recibe información del proceso, e identifica si el sistema se encuentra en un estado inválido. Analiza si el estado inválido es debido a la presencia de una falla, y la clasifica como falla incipiente o abrupta	Monitorea Sistema Identifica Estado del Sistema
Localizador	Localiza la región donde ocurre la falla incipiente	Ubicar Falla
Diagnosticador	Identifica el modo de falla, sus causas y consecuencias. Identifica nuevos modos de fallas y sus posibles causas y consecuencias	Analiza Falla
Predictor	Predice el lapso de tiempo en el cual una falla incipiente puede apuntar a una falla funcional	Predice Falla
Planificador	Planifica la realización de tareas de mantenimiento basado en prioridad y disponibilidad de recursos. También, replanifica las tareas de mantenimiento no concluidas o no realizadas.	Planifica Tarea Replanifica Tarea
Ejecutor	Ejecuta la tarea de mantenimiento, según el caso, y propone planes de contingencia en el caso de no poder realizar la tarea	Ejecuta Tarea Reporta Tarea

Tabla 12.1: Actores y casos de uso del SMF

12.3.1. SMA basado en SCDIA para el SMF

El modelo de agentes que implementa del modelo de referencia para manejo de fallas de la Figura 12.9 se presenta en la Figura 12.12. Dicha figura muestra que el SMA usa el modelo de referencia SCDIA de la Sección 10.9. Dado que los requerimientos funcionales del agente de manejo de fallas se refiere a la gestión de mantenimiento y actividades asociadas, este agente de manejo de fallas representa en el segundo nivel de abstracción del modelo SADIA al Agente Ingeniería de Mantenimiento, y su descripción en el tercer nivel de abstracción de SADIA será el SMA de la Figura 12.12. A objeto de usar el modelo de referencia SCDIA, el problema de manejo de fallas ha sido adaptado como un problema genérico de control en lazo cerrado. En dicha figura se muestra que el SMF basado en agentes interactúa con el SCDIA a través del MGS.

Basado en el modelo SCDIA, y a partir de la tabla de actores, se definieron los siguientes agentes:

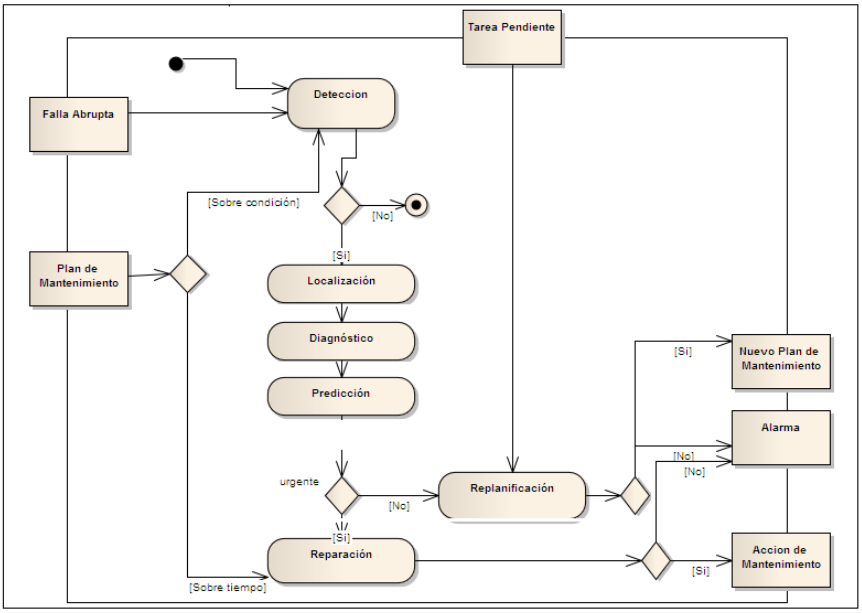

Figura 12.11: Diagrama de actividades del SMF

1. **Agente Especializado Detector:** desempeña tareas de detección, usando para ello técnicas específicas conocidas por el agente.
2. **Agente Especializado Localizador:** se encarga de localizar la parte del sistema en donde ocurre una falla.
3. **Agente Especializado Diagnosticador:** desempeña tareas de diagnóstico usando para ello técnicas específicas conocidas por el agente.
4. **Agente Especializado Predictor:** desempeña tareas de predicción usando para ello técnicas específicas conocidas por el agente.
5. **Agente Coordinador:** recolecta información de otros agentes y toma la decisión sobre la realización del mantenimiento correctivo (reparación). En caso contrario, efectúa la planificación (o replanificación) de la(s) tarea(s) de mantenimiento preventivo a realizar.
6. **Agente Controlador:** propone los tareas a seguir para la ejecución del plan de mantenimiento propuesto por el agente Coordinador, sobre un horizonte de tiempo determinado.
7. **Agente Actuador:** ejecuta las acciones de mantenimiento.
8. **Agente Observador:** recolecta información del proceso e identifica si el sistema se encuentra en un estado inválido, observa el estado del mantenimiento y detecta las fallas abruptas en el sistema.

En esta descomposición de agentes, se observa que el actor Planificador se distribuye entre los agentes Controlador y Coordinador. Todo lo relacionado con la recolección de información de los actores es una funcionalidad del

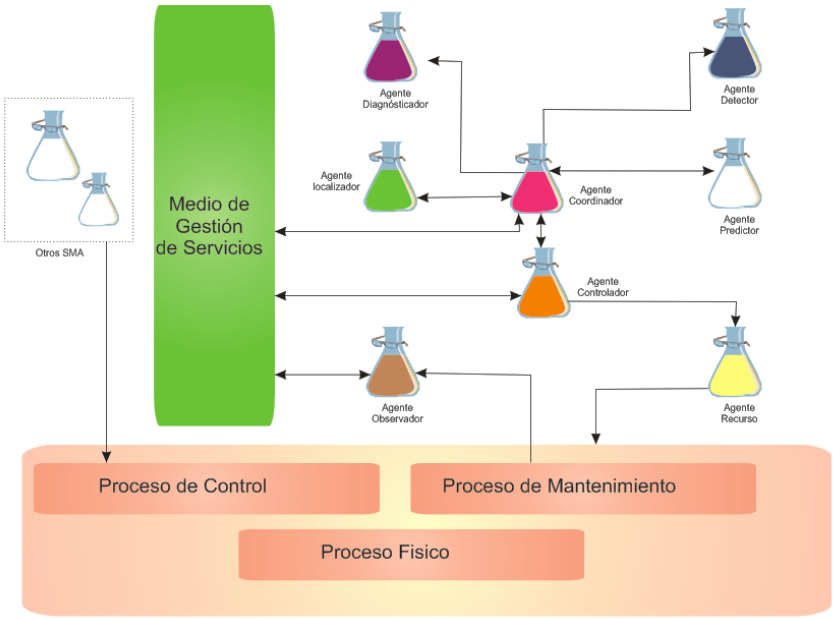

Figura 12.12: Modelo de agentes para el SMF

agente Observador y lo referente a la ejecución de acciones sobre el proceso de cualquiera de los actores es una funcionalidad del agente Actuador.

12.3.2. Especificación de los agentes del SMF

Una vez definidos los agentes y sus funcionalidades, la especificación del agente considera primeramente el modelo del agente , seguido por el modelo de tareas, el modelo de coordinación y el modelo de comunicación.

1. **El modelo del agente**. Este modelo especifica las funcionalidades del agente, sus objetivos y servicios, en tablas descriptivas de los parámetros necesarios para caracterizar al agente. La tablas 12.2 y 12.3 ejemplifican esta especificación para el caso del agente Coordinador.

Nombre	Coordinador
Tipo	Agente de software
Papel	Tomar decisión para la planificación de las tareas de mantenimiento y coordinar todo lo que tiene que ver con las tareas del SMF.
	Continua en la próxima página

Tabla 12.2 –Finaliza de la página anterior

Descripción	Recolecta la información obtenida por los agentes Detector, Localizador, Diagnosticador y Predictor de la(s) instancia(s) del sistema; basado en esta información toma la decisión sobre la realización de tareas de mantenimiento en el sistema, también efectúa la replanificación de la(s) tarea(s) de mantenimiento preventivo a realizar.

Tabla 12.2: Descripción del agente

Nombre	Planificar las tareas de mantenimiento a realizar en las instancias del sistema según la disponibilidad de recurso.
Tipo	Objetivo por condición.
Parámetro de entrada	Datos e información de los otros agentes del sistema y de otros SMA.
Parámetro de salida	El plan de mantenimiento y/o orden de mantenimiento correctivo (tareas urgentes).
Condición de activación	Recibir información de los agentes especializados o del agente Observador.
Condición de finalización	Establecer un macro-plan de mantenimiento a realizar en el sistema.
Condición de éxito	Obtener un macro plan de mantenimiento para lograr niveles de confiabilidad aceptables en las instancias supervisadas.
Condición de fracaso	No cumplimiento de la condición de éxito.
Lenguaje de representación	Lenguaje natural.
Descripción	El agente Coordinador evalúa los diferentes escenarios para la elaboración ó replanificación de los planes generales de mantenimiento preventivo y/o ordenar la ejecución de mantenimiento correctivo (tareas urgentes).

Tabla 12.3: Objetivo del agente

Nombre	Proporcionar plan de mantenimiento
Tipo	Gratuito, concurrente
Parámetros de entrada	Datos e información provenientes de los agentes especializados y del agente Observador
Parámetros de salida	Macro-plan de mantenimiento
Lenguaje de representación	Lenguajenatural

Tabla 12.4: Servicio del Agente

El agente Coordinador presta tres servicios al SMA:

- Proporcionar plan de mantenimiento.
- Modificar plan de mantenimiento.
- Llamar a agentes especializados.

La Tabla 12.4 especifica solamente al servicio *Proporcionar plan de mantenimiento.*

2. **El modelo de tareas**. Las tareas que deben ejecutar los agentes se derivan de la Tabla 12.1 de casos de uso. Para el caso del agente Manejo de Fallas que se esta especificando, la Tabla 12.5 muestra las tareas definidas.

Agente	Tareas
Observador	Identificar fallas funcionales abruptas Calcular índices de funcionamiento Determinar el estado del mantenimiento
Detector	Llevar estadísticas sobre la ocurrencia de fallas Seleccionar técnica de detección Reajustar modelos de detección Incorporar nuevos modos de fallas
Localizador	Ubicar falla
Diagnosticador	Identificar modos de fallas y sus causas Llevar estadísticas de los modos de fallas Llevar estadísticas sobre las causas de las fallas Realizar análisis sobre las consecuencias de las fallas en el sistema Reajustar modelos de diagnóstico Caracterizar nuevos modos de fallas Incorporar nuevas causas de fallas
Predictor	Calcular curvas de confiabilidad en equipos Generar índices de confiabilidad del proceso Incorporar nuevos modelos de predicción
Controlador	Proponer la programación (scheduling) de mantenimiento Modificar el programa de mantenimiento
Coordinador	Proponer plan de mantenimiento Evaluar recursos Ordenar realización de tareas especializadas Ordenar mantenimiento correctivo Modificar plan de mantenimiento preventivo
Actuador	Ejecuta las tareas de mantenimiento Ejecuta planes de contingencia

Tabla 12.5: Tareas del Agente Manejo de Fallas

La caracterización detallada de cada tarea se hace en el modelo de tareas, describiendo los aspectos importantes de dicha tarea. En el caso del agente Coordinador, la Tabla 12.6 especifica la tarea *Evaluar Recurso*.

Tarea	Evaluar Recurso
Objetivo	Evaluar los recursos disponibles (mano de obra, repuestos, entre otros) en el sistema para la realización de las tareas de mantenimiento correctivo ó tareas urgentes.
Precondición	La información proveniente del agente Base de Datos debe ser clara, precisa y completa.
Frecuencia	Relativa a la realización de una tarea urgente o de modificar el plan de mantenimiento
Continua en la próxima página	

Tabla 12.6 –Finaliza de la página anterior

Descripción	A través de esta tarea el agente Coordinador evalúa los recursos disponibles para la realización de una tarea de mantenimiento correctivo (ó tareas urgentes) en el sistema, ó para la modificación del plan de mantenimiento.

Tabla 12.6: Modelo de Tarea

3. **El modelo de coordinación**. El modelo de coordinación describe las conversaciones que deben darse entre los agentes del sistema para realizar sus tareas y ofrecer sus servicios. Siguiendo la metodología MASINA, para cada conversación deben especificarse un conjunto de parámetros mínimos que definen a dicha conversación. Para el agente Manejo de Fallas se definieron seis conversaciones necesarias que deben darse para el funcionamiento del SMA:

   - Mantenimiento por condición.
   - Tareas de mantenimiento.
   - Tareas urgentes.
   - Replanificación de tareas.
   - Estado de mantenimiento.
   - Identificar falla funcional.

   La Tabla 12.7 especifica la conversación *Replanificación de Tarea*. Dicha conversación se inicia cuando el agente Actuador reporta al SMA la no ejecución de una acción de mantenimiento sobre el proceso. El agente Coordinador recibe esta información e inicia la conversación con el agente base de datos a fin de recolectar la información necesaria para replanificar la tarea pendiente (recursos requeridos y costos, entre otros). La Figura 12.13 ilustra en un diagrama de interacción la conversación presentada.

Conversación	Replanificación de Tarea
Objetivo	Replanificar tareas que no se efectuaron en el sistema.
Agentes participantes	Coordinador, Base de Datos, Humano.
Iniciador	Agente Coordinador.
Actos de Habla	Buscar Tareas Pendientes Alarma Enviar Plan de Mantenimiento.
Precondición	Tener que realizar replanificación de una tarea de mantenimiento
Condición de Terminación	Replanificación de la tarea y almacenamiento en el nuevo plan de mantenimiento. Si no se logra la replanificación, se da una alarma.
Descripción	Mediante esta conversación, el agente Coordinador busca la información alojada en el agente Base de Datos para replanificar las tareas de mantenimiento pendientes en el sistema, y realizar un nuevo plan de mantenimiento. Si la tarea es urgente y no se puede replanificar se da una alarma, de lo contrario sigue en espera.

Tabla 12.7: Modelo de Coordinación

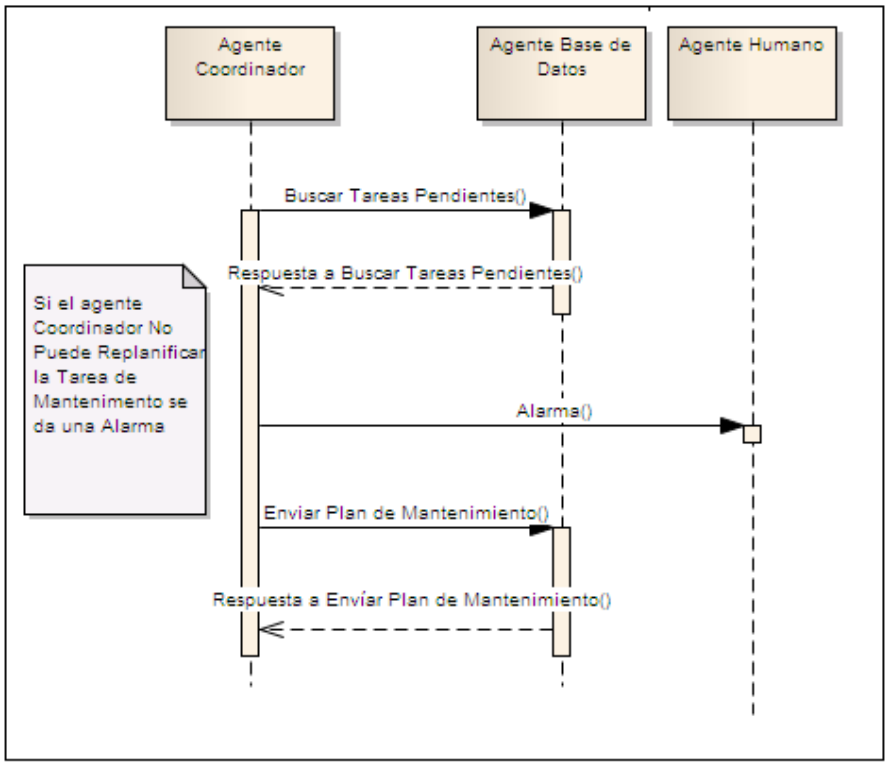

Figura 12.13: Diagrama de interacción de la conversación *Replanificación de Tareas*

4. **El modelo de comunicación**. Según la metodología MASINA, el modelo de comunicación describe los actos de habla (interacción) asociados a cada conversación. La Tabla 12.8 especifica el acto de habla *Buscar Tarea Pendiente*, asociada a la conversación *Replanificación de Tarea*. Como puede observarse en dicha tabla, este acto de habla también esta asociado a la conversación *Tareas Urgentes* definida para el agente Manejo de Fallas.

Acto de Habla	Buscar Tareas Pendientes
Nombre	Buscar Tareas Pendientes
Tipo	Requerimiento de información.
Conversación asociada	Tareas Urgentes Replanificación de Tarea
Objetivo	Buscar en la base de datos las tareas de mantenimiento pendientes a realizar en el sistema.

Continua en la próxima página

Tabla 12.8 –Finaliza de la página anterior

Agentes Participantes	Coordinador Base de Datos
Iniciador	Coordinador
Precondición	Existen tareas pendientes en el sistema
Condición de terminación	El agente Coordinador recibe la información sobre las tareas de mantenimiento pendientes en el sistema.
Descripción	El agente Coordinador solicita y recibe la información sobre las tareas de mantenimiento pendientes que están almacenadas en la base de datos.

Tabla 12.8: Modelo de Comunicación

Otros aspectos que deben ser especificados en los modelos del agente son los esquemas de coordinación y comunicación que se usan, los cuales dependen del tipo de servicio que prestan y las restricciones a las que están sometidos los agentes. En general, los esquemas de coordinación por omisión son de tipo centralizada, a través de mecanismos de pases de mensajes para establecer la comunicación entre los agentes que participan en la conversación.

12.4. Agente para el manejo de situaciones anormales

Una situación anormal es una condición, evento o circunstancia que involucra la pérdida de la capacidad de un elemento de realizar una función asociada a él, que hace que las operaciones se desvíen de su estado de operación normal. Las consecuencias pueden ser tanto mínimas como de alcances catastróficos. Esas situaciones podrían extenderse, desarrollarse y cambiar en el tiempo, aumentando la complejidad de los requerimientos de intervención. Por eso, para resolver esas dificultades es importante entender los factores que las generan. En ese sentido, si bien estos pueden darse aisladamente, en la mayoría de los casos ese tipo de situaciones son el resultado de la interacción entre múltiples variables.

Una situación anormal produce efectos (síntomas) que son las señales que permiten identificarla. La meta del manejo de situaciones anormales en la industria es mantener el desempeño operacional, mantener la disponibilidad continua de los activos de producción, reducir la carga al operador, y minimizar y reducir los costos de operación. De esta manera, la disponibilidad de un Sistema de Manejo de Situaciones Anormales (SMSA), es de gran importancia para detectar dichas situaciones en forma temprana y recomendar cursos de acción a seguir antes que ellas impacten la operación.

Es importante resaltar que una situación anormal no implica necesariamente la existencia de una falla, y su aparición puede deberse a cambios en las condiciones de producción y operación que, de no detectarse temprana-

mente, si podrían conducir a la aparición de una falla en el sentido del caso presentado en la Sección 12.3.

Un sistema de manejo de fallas debe considerar, al menos, tres bloques o flujos:

1. **Detección y diagnóstico**. El flujo correspondiente a detección y diagnóstico contempla las siguientes funcionalidades:

   - Obtención de información de estado: monitoreo constante del estado del proceso.
   - Detección de síntomas: detección de señales que puedan identificar la ocurrencia de una situación anormal.
   - Ejecución de pruebas: pruebas realizadas para corroborar la ocurrencia de una situación anormal.
   - Identificación de situación: identificación de la situación anormal que está ocurriendo, entre las almacenadas en los modelos de datos disponibles.
   - Evaluación de criticidad: evaluación de la criticidad de la situación anormal que está ocurriendo.
   - Alarma de parada segura: emisión de una señal para el comienzo de la parada segura del proceso, en caso de que la situación lo amerite.
   - Alarma de protección fuego y gas: emisión de una señal para la activación de los procesos de protección de fuego y gas, en caso de que haberse producido un siniestro en las instalaciones.
   - Registro de eventos: registro de las situaciones anormales que se presentan, las acciones tomadas y demás eventos ocurridos, con el fin de alimentar la experiencia del sistema.

2. **Tratamiento**. El flujo correspondiente al tratamiento contempla las siguientes funcionalidades:

   - Identificación de causas: establecimiento de las causas de la situación anormal que está ocurriendo.
   - Evaluación de consecuencias: determinación de las consecuencias que puede acarrear la situación anormal que se está presentando.
   - Ejecución de acciones: ejecución de las acciones correctivas necesarias para solventar la situación anormal que está ocurriendo.

3. **Manejo de alarmas**. El flujo correspondiente al manejo de alarmas contempla las siguientes funcionalidades:

   - Emisión de alarmas: emisión de diferentes alarmas según criticidad de la situación anormal.
   - Cambio de estado de alarmas: incremento ó disminución en los niveles de alarma en caso de que una alarma no haya sido atendida, o que la situación anormal haya empeorado, o si una situación anormal no ha sido solventada pero está siendo atendida.

- Evaluación del proceso: evaluación del desempeño del proceso una vez tomadas acciones para solventar una situación anormal.

12.4.1. SMA basado en SCDIA para el SMSA

En esta sección se presenta un modelo genérico basado en agentes, denominado Manejo de Situaciones Anormales (MSA) desarrollado en [3, 2]. Este agente implementa las funcionalidades de un SMSA y forma parte del conjunto de agentes definidos en el segundo nivel de abstracción del modelo de referencia SCDIA. Debido a que una situación anormal no necesariamente implica una falla, este agente conjuntamente con el agente Manejos de Fallas descrito en la Sección 12.3 se complementan para llevar a cabo lo relacionado con la gestión de seguridad y mantenimiento en procesos industriales.

La arquitectura del agente MSA se basa en el modelo de referencia SCDIA en el tercer nivel de abstracción del modelo SADIA. El SMA está compuesto por los cinco agentes del SCDIA, cada uno con tareas y servicios específicos para el manejo de situaciones anormales. A modo de ilustración, este agente es instanciado para el caso del manejo de situaciones anormales del agente Pozo definido en el primer nivel de abstracción del ejemplo de la Sección 10.10.3.

Tal como se describe en la Figura 10.21, los pozos LAG para poder realizar la función de extraer los hidrocarburos desde los yacimientos hasta la superficie, tienen instalados una serie de elementos y dispositivos de campo, además, están asociados a múltiples de levantamiento y estaciones de flujo. En la operación diaria del pozo LAG, la combinación de ciertos factores y circunstancias no deseadas en los elementos presentes en el pozo generan la ocurrencia de situaciones anormales que afectan la producción de los mismos.

El manejo de situaciones anormales en pozos con LAG implica realizar una observación constante del estado del pozo, esto es, observar las variables funcionales del pozo y verificar que sus valores se encuentran dentro de una banda de operación permitida. Si ocurre que el estado se aleje de la banda de operación permitida, se está en presencia de una situación anormal, la cual se debe diagnosticar. Para ello se identifican sus causas, consecuencias, y se debe generar un curso de acción a seguir en el tratamiento de la situación anormal.

Basado en el modelo SCDIA, se definieron los siguientes agentes:

1. **Agente Observador:** recolecta los datos provenientes del sistema SCADA, y de las bases de datos que puedan aportar información acerca del estado de los procesos que se desarrollan en el pozo monitoreado. Además, de ser necesario, podrá preprocesar y/o validar los datos, calcular promedios y estimadores, hacer observación de estados y cualquier otra operación

para obtener la información requerida por los demás agentes para realizar el diagnóstico, aislamiento y tratamiento de situaciones anormales. El agente se encarga de transmitir la información de estado a los agentes respectivos del SMA.

2. **Agente Controlador:** recibe la información de estado emitida por el agente Observador y compara las condiciones actuales del proceso con las condiciones deseadas para el mismo. En caso de que se alejen de una cierta banda de tolerancia, ejecuta reglas almacenadas dentro de un motor de inferencia, lo que puede resultar en órdenes de activación de alarmas, emisión de órdenes para ejecución de aplicaciones de diagnóstico, caracterización de condiciones de operación, entre otros. Cambia valores establecidos para condiciones de operación (como por ejemplo, valores nominales para variables de proceso). Ejecuta modelos de detección y diagnóstico de situaciones anormales, determina las causas de las mismas y evalúa sus consecuencias, a partir del modelo de datos alimentado por el agente de Ingeniería de Mantenimiento.
3. **Agente Actuador:** activa alarmas y las hace visibles para cada actor involucrado con la resolución del problema (operadores SCADA, ingenieros de optimización). Además, produce cambios en el SCADA (como por ejemplo el cambio de consignas para controladores o la activación de una alarma para la parada de planta), crea y/o modifica agendas para medidas de pozos y ejecuta flujos de trabajo vinculados con planes de mantenimiento correctivo (elaborados, como se verá más adelante, por el agente coordinador), que conlleven a la resolución de las situaciones anormales que se presenten, incluyendo medidas de mantenimiento correctivo.
4. **Agente Coordinador:** emite las solicitudes de servicios a los agentes especializados. Además, interactúa con el agente Ingeniería de Mantenimiento para solicitarle planes de mantenimiento correctivo, y con agente Actuador para indicarle que hacer. Por otro lado, supervisa el funcionamiento del motor de inferencia del sistema y lo modifica en caso de ser necesario, ejecuta pruebas que permiten identificar y localizar modos de falla. Establece, en conjunto con el agente Ingeniería Mantenimiento, las agendas de mantenimiento correctivo.
5. **Agente Especializado Analizador:** realiza análisis detallado basado en modelos matemáticos de los pozos (generados por los ingenieros de optimización) sobre alguna posible situación anormal que se este presentando.
6. **Agente Especializado Caracterizador:** caracteriza el comportamiento de las variables de los pozos (presión de cabecera, presión de tubería, rata de inyección de gas, presión de múltiple de gas lift) como continuas, intermitentes y erráticas.

12.4.2. Especificación de los Agentes del SMSA

Siguiendo la metodología MASINA, en esta sección se presenta la especificación del agente a partir de sus funcionalidad.

1. **El modelo del agente**. Este modelo especifica las funcionalidades del agente, sus objetivos y servicios, en tablas descriptivas de los parámetros necesarios para caracterizar completamente al agente. La tablas 12.9 y 12.10 muestran la especificación para el caso del agente Controlador. Además de los parámetros comúnmente descritos, se da información acerca de los modelos de referencia que pueden asociarse al agente.

Nombre	Controlador
Posición	Tercer nivel de SADIA
Marco de Referencia	Agente del SCDIA
Descripción	Evalúa el comportamiento del pozo monitoreado, comparándolo con el comportamiento deseado, y puede emitir posibles recomendaciones para el manejo de una situación anormal.

Tabla 12.9: Descripción del agente

Nombre	Evaluar Condición
Descripción	El agente Controlador tiene como objetivo evaluar la condición del pozo, para identificar si esta ocurriendo una situación anormal, y si es así recomendar posibles acciones correctivas para mantener el pozo en las condiciones deseadas.
Parámetros de Entrada	Información de estado Consignas
Parámetros de Salida	Estado del pozo Acciones correctivas Alarmas de parada de planta Alarmas de protección de fuego y gas
Condición de Activación	Al nacer el agente
Condición de Fin	Al eliminarse el agente
Condición de Éxito	El agente evalúa correctamente la condición del pozo
Condición de Fracaso	Ocurren situaciones anormales que no pueden solventarse

Tabla 12.10: Objetivo del agente

El agente Controlador tiene definidos tres servicios que presta al SMA:

- Recomendar acciones correctivas.
- Emitir orden para activar alarma de protección de fuego y gas.
- Emitir orden para activar alarma de parada segura.

La Tabla 12.11 especifica el caso del servicio *Recomendar acciones correctivas*.

Nombre	Recomendar acciones correctivas
Descripción	El agente a partir de la información de estado y de las consignas establecidas podrá recomendar posibles acciones que deben ejecutarse para solventar las posibles situaciones anormales que estén ocurriendo.
Parámetros de Entrada	Información de estado Consignas
Parámetros de Salida	Recomendación de posibles acciones correctivas.

Tabla 12.11: Servicio del agente

2. **El modelo de tareas**. Las tareas que deben ejecutar los agentes del SMA se muestran en la Tabla 12.12. La realización de dichas tareas deben garantizar las funcionalidades del agente.

Agente	**Tareas**
Observador	Obtener información de estado Solicitar datos al Agente Gestor de Datos del medio de gestión de servicios Acondicionar datos provenientes del proceso.
Controlador	Diagnóstico Identificación de causas Identificación de consecuencias Evaluación de criticidad Cambios de parámetros de entonación
Coordinador	Solicitud de ejecución de servicio de caracterización Solicitud de ejecución de servicio de análisis detallado
Actuador	Ejecución de acciones correctivas Registro de estatus de alarma Cambios de estado de alarmas
Analizador	Análisis de situación anormal
Caracterizador	Caracterización de comportamiento de variables del pozo

Tabla 12.12: Modelo de Tareas

La caracterización detallada de cada tarea de hace en el modelo de tareas, donde se describen los parámetros más importantes para cada una de las tareas definidas. En el caso del agente Controlador, la Tabla 12.13 especifica la tarea *Diagnóstico*. Además de los parámetros comúnmente definidos, también se especifican las tareas requeridas por la tarea analizada, las cuales constituyen las subtareas.

Nombre	Diagnóstico
Objetivo	Diagnosticar situación anormal a partir de síntomas
Descripción	Se identifican las situaciones anormales, posibles causas y sus posibles consecuencias, a partir de los síntomas detectados.
Precondición	Información de estado del proceso
Subtareas	Identificación de causas Identificación de consecuencias

Tabla 12.13: Modelo de Tarea (Diagnóstico)

3. **El modelo de coordinación**. El modelo de coordinación describe las conversaciones que deben darse entre los agentes del sistema para el logro de los objetivos grupales y ofrecer sus servicios. La Tabla 12.14 especifica la conversación *Obtención del Control*, del agente Controlador. Dicha conversación se inicia cuando el agente Observador reporta al agente Controlador la información sobre el estado del proceso. El agente Controlador solicita al agente Coordinador la realización de pruebas específicas para poder tomar decisión acerca de la situación anormal que esta ocurriendo. Finalmente, al agente Controlador puede solicitar al agente Actuador la realización de alguna acción específica. La Figura 12.14 ilustra en un diagrama de interacción la conversación presentada.

Conversación	Replanificación de Tarea
Objetivo	Determinación de las acciones de control
Agentes participantes	Observador, Coordinador, Actuador
Iniciador	Agente Observador
Actos de Habla	Obtención de información de estado de pozo Solicitud de servicios de diagnóstico Orden de ejecución
Precondición	Estado del proceso
Condición de terminación	Orden de ejecución de acción o error
Descripción	El objetivo del agente controlador es obtener las acciones pertinentes ante la presencia de una situación anormal, para ello interactúa con los agentes observador, coordinador y actuador.

Tabla 12.14: Modelo de Coordinación

4. **El modelo de comunicación**. El modelo de comunicación describe los actos de habla (interacción) asociados a cada conversación. Para la conversación *Obtención del Control* de la Figura 12.14 se tienen dos actos de habla:

   - Solicitud de prueba.
   - Orden de ejecución.

   La Tabla 12.15 especifica el acto de habla *Solicitud de prueba*.

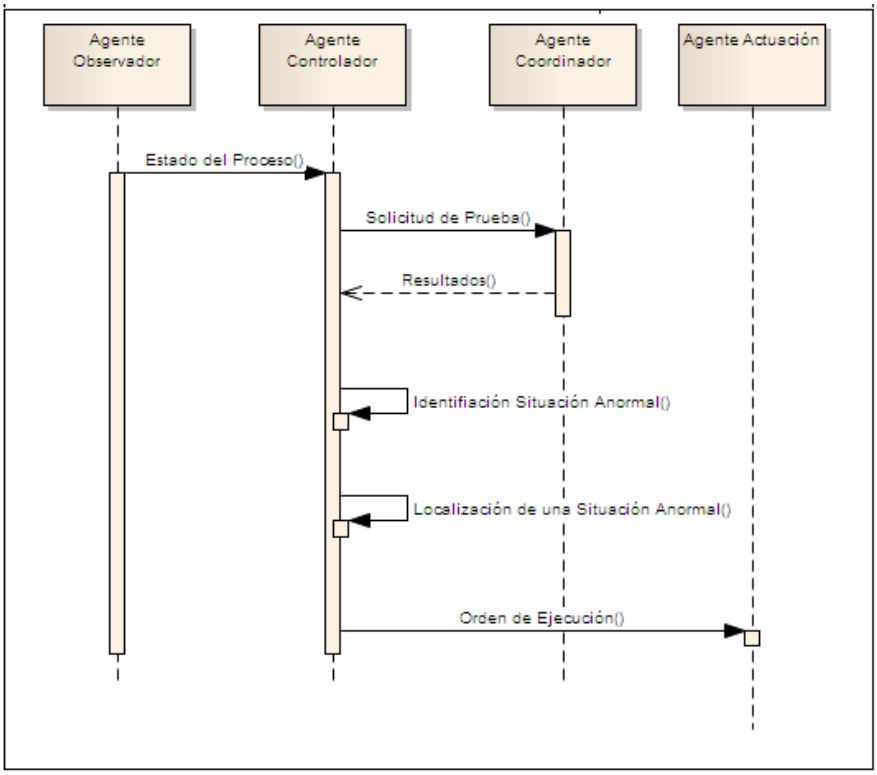

Figura 12.14: Diagrama de interacción de la conversación *Obtención del Control*

Nombre	Solicitud de prueba
Tipo	Solicitud de información
Objetivo	Solicitud de prueba
Agentes Participantes	Controlador, Coordinador
Iniciador	Controlador
Precondición	Solicitud de análisis de situación anormal
Condición de terminación	Aceptación de la solicitud
Conversaciones asociadas	Obtención del control
Descripción	El agente Controlador solicita al Coordinador la realización de un análisis detallado de la situación anormal.

Tabla 12.15: Modelo de Comunicación

5. **El modelo de inteligencia.** Otro modelo a especificar en el SMA es el Modelo de Inteligencia, el cual, en el caso de agentes con capacidades inteligentes, describe los mecanismos de razonamiento, adaptación y aprendizaje. Inicialmente los agentes de un SMA pueden ser agentes reactivos, pero se le puede integrar capacidades inteligentes. Para el caso del agente Manejo de Situaciones Anormales, se especifica el modelo de inteligencia

del agente Controlador, el cual describe los mecanismos de razonamiento, aprendizaje y representación del conocimiento. El motor de inferencia del agente controlador está formado por reglas de producción, cada regla consiste de una parte SI (condición) y otra parte ENTONCES (acción). Las listas de partes SI son un conjunto de condiciones en una cierta combinación lógica. La Tabla 12.16 presenta los parámetros generales que especifican el modelo de inteligencia del agente, tanto en el mecanismo de razonamiento como en el mecanismo de aprendizaje.

Mecanismo de Razonamiento	
Fuente de Información	Agente de Observación Agente Coordinador Medio de Gestión de Servicios
Fuente de Activación	Intervención
Tipo de Inferencia	Deductiva
Estrategia de Razonamiento	Sistema experto
Mecanismo de Aprendizaje	
Tipo	Adaptativo
Técnica de Representación	Reglas
Fuente de Aprendizaje	Exitos o fracasos
Mecanismo de Actualización	Las experiencias previas son utilizadas para actualizar el conocimiento.

Tabla 12.16: Modelo de Inteligencia

En general, las reglas del agente Controlador representan el conocimiento disponible para determinar, entre otros:

- El flujo de producción de un pozo, por medio del sensor de flujo.
- Diferencias entre la señal de pozo en producción (obtenida de la regla anterior) y la categoría del pozo almacenada en las bases de datos.
- Ocurrencia de una ruptura de la línea de producción mientras el pozo está en operación.
- Cambios en el método de levantamiento artificial por comparación del método de diseño artificial con el método de operación.
- Obstrucción del flujo de gas de levantamiento en un pozo y, por lo tanto, el pozo no lo está recibiendo.

Adicionalmente a los modelos presentados, al SMA se le debe especificar los esquemas de coordinación y comunicación usados, los cuales, en general, dependen del tipo de servicio que prestan y las restricciones a las que están sometidos los agentes. Como ya se ha mencionado en secciones anteriores, los esquemas de coordinación por omisión son de tipo centralizada, a través de mecanismos de pases de mensajes para establecer la comunicación entre los agentes que participan en la conversación.

Referencias

1. Rios A., Cerrada M., Narciso F., Hidrobo F., and Aguilar J. Implantando Sistemas de Control Inteligentes. *Ciencia e Ingenieria*, 29(3):249–260, 2008.
2. J. Aguilar, F. Prato, and C. Bravo. Desarrollo de un Sistema Multiagente de Manejo de Situaciones Anormales para un Pozo de Levantamiento Artificial por Gas. In *Proceeding of the XXXII Conferencia Latinoamericana de Informática*, Agosto 2006.
3. C. Bravo, J. Aguilar, E. Colina, and F. Rivas. Sistema Multiagentes para Tratamiento de Situaciones Anormales en Procesos Industriales. In *Proceedings of the V Congreso Nacional de la Asociación Colombiana de Automática*, Julio 2003.
4. E. Camargo, J. Aguilar, A. Ríos, F. Rivas, and J. Aguilar-Martin. Intelligent supervision systems for improving the industrial production performance in oil wells. In *Proccedings of CIMMACS Congress 2010-Advances in Computational Intelligence, Man-Machine Systems and Cybernetics*, pages 289–296, 2010.
5. 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.
6. M. Hung, K. Chen, R. Ho, and F. Cheng. Development of an e-Diagnostics/Maintenance Framework for Semiconductor Factories with Security Consideration. *Advanced Engineering Informatics*, 17(1):165–178, 2003.
7. B. Iung. From Remote Maintenance to MAS-based E-maintenance of an Industrial Process. *International Journal of Intelligent Manufacturing*, 6(1):59–82, 2003.
8. N. Jennings and S. Bussmann. Agent-based Control Systems. *IEEE Control Systems Magazine*, pages 61–73, 2003.
9. I.S Kim. Computerized Systems for on-line Management of Failures: a State-of-the-Art Discusión of Alarm Systems and Diagnostic Systems Applied in the Nuclear Industry. *Reliability Engineering and System Safety*, 4:279–295, 1994.
10. J.B. Leger, B. Iung, A. Ferro De Beca, and J. Pinoteau. Innovative Approach for new Distributed Maintenance System: Application to Hydro Power Plants of the REMAFEX Project. *Computer in Industry*, 38(1):131–148, 1999.
11. J. Moubray. *Reliablity-Centered Maintenance*. Industrial Press, 1992.
12. 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.
13. 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.
14. C. Parra-Ortega, E. Colina-Morles, and E. Chacón Ramírez. Design Framework for Intelligent Supervision of Industrial Processes. *WSEAS Transaction on Systems*, 7(7):616–625, 2008.
15. H. Panetto R. Yu, B. Iung. A Multi-Agents Based E-maintenance System with Case-based Reasoning Decision Support. *Engineering Application of Artificial Intelligence*, 16:321–333, 2003.
16. M. Rao, H. Yang, and H.Yang. Integrated Distributed Intelligent System Architecture for Incidents Monitoring and Diagnosis. *Computer in Industry*, 37(1):143–151, 1998.
17. T. B. Sheridan. Supervisory Control. In G. Salvendy, editor, *Handbook of Human Factors*, pages 1243–1268. John Wiley, New York, 1987.
18. I. Ushakov and R. Harrison. *Handbook of Reliability Engineering*. John Wiley, New York, 1994.
19. 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.

Capítulo 13
Aplicaciones de Planificación

Resumen: En este capítulo se presenta el diseño de sistemas basado en SMA para el manejo de la planificación en la producción industrial.

13.1. Introducción

Planificar en la industria, según algunos autores, es el proceso de seleccionar y secuenciar actividades, cuya ejecución persigue uno o más objetivos, y a la vez se satisfacen un conjunto de restricciones del dominio [2, 12]. La planificación lleva consigo la programación de actividades, la cual es el proceso de asignar recursos y tiempos al conjunto de actividades del plan. Estas asignaciones deben satisfacer un conjunto de reglas o restricciones, que reflejan las relaciones entre actividades y limitaciones de disponibilidad de algunos recursos compartidos.

En general, un sistema de planificación de la producción debe generar y ejecutar una secuencia de acciones necesarias para satisfacer los requerimientos de cada servicio a prestar [2, 7, 12, 9]. La secuencia de acciones (plan detallado) involucra la asignación de los recursos a lo largo del tiempo, hasta la completa ejecución del plan. Además, este sistema también debe supervisar la ejecución del plan, y en caso de alguna desviación, debe realizar dinámicamente las modificaciones al plan en ejecución (replanificación). Un buen plan es aquel que permite realizar ajustes sin comprometer las metas globales.

Así, la planificación es un proceso sistemático en el que primero se establece una necesidad, y en base a ésta se desarrolla la mejor manera de enfrentarse a ella, dentro de un marco estratégico, que permite identificar las prioridades y los principios funcionales (reglas del negocio).

Para la elaboración del plan (*planning*) y la programación de actividades (*scheduling*) del proceso de producción, es necesario acceder a la información interna (capacidad de producción, producción actual, infraestructura instalada, etc.) y externa (demanda, situación actual del mercado, etc.) al proceso de forma integrada, caracterizar el proceso de producción (cuánto se va a producir, cuáles son los insumos que se requieren para la producción, entre otros aspectos), evaluar sus posibles efectos, reconocer y gestionar los requerimientos (precondiciones), tener en consideración las restricciones del sistema (excepciones), optimizar la distribución de los recursos limitados, como requerimientos básicos.

Un objeto de negocio puede requerir contratar los servicios a otro objeto de negocio, con el fin de poder ejecutar sus acciones para alcanzar las metas pautadas. Para este fin, es necesario contar con un módulo responsable de recibir y gestionar los requerimientos del negocio provenientes de los objetos de negocio generadores de requerimientos. Así, un objeto de negocio alcanzará sus metas u objetivos contratando, en caso que lo requiera, los servicios a otros objetos de negocio, y ejecutando sus propias acciones. Las metas u objetivos alcanzados por cada objeto de negocio, permitirá que se logre alcanzar y cumplir el plan global de la organización. El siguiente paso a la caracterización del sistema es la elaboración del plan, en el cual se debe determinar el plan general y, posteriormente, la programación de actividades a ser ejecutadas. Durante la programación de actividades se contempla la asignación y manejo de recursos y productos, así como el manejo de los desechos. Una vez determinado el plan general y la programación de actividades, es necesario contar con un módulo capaz de ejecutar el plan y mantener una continua supervisión del desempeño del mismo.

13.2. Planificación en entornos de automatización industrial

A partir del modelo funcional del flujo de datos propuesto en [6], se puede comprender cuáles son las funciones que deben ser ejecutadas por las aplicaciones de automatización en el área de planificación. Sus funciones representan la interfaz entre las funciones del sistema de control y la empresa, a través de la planificación de la producción. Un plan de producción detallado dentro de un área se define como una función de control. Las funciones generales de la planificación de la producción, son las siguientes [2, 6, 9]:

- Determinar el plan de producción para cada objeto de negocio.
- Identificar los requerimientos de materia prima a largo plazo.
- Determinar el programa de entrega de los productos terminados.
- Determinar la disponibilidad de productos para la venta.
- Determinar la capacidad de producción.

- Determinar el rendimiento del objeto de negocio.
- Determinar la condición de operación (parte del estado del entorno).
- Optimizar el sistema, mediante el establecimiento de la mejor estrategia (mejor plan) para determinadas condiciones de operación y recursos disponibles, tomando en cuenta todas las restricciones, incluyendo las de tipo económico.
- Generar la secuencia de operaciones (programación de actividades), detallando los equipos, personal, y los mecanismos para la recepción, movilización y despacho de materia prima, productos intermedios, y productos terminados.
- Generar y optimizar las órdenes de movimiento de equipos, materiales, productos y desechos, especificando el material involucrado, la cantidad del material, el origen, el destino, el tiempo inicial y el tiempo final.

Las funciones de planificación de la producción pueden generar o modificar la siguiente información [2, 12]:

- El plan de producción.
- La producción actual (desempeño actual) para cada objeto de negocio.

13.3. Modelo de referencia para la planificación de la producción

Un sistema de apoyo para la planificación de la producción y manejo de los factores de producción debe generar y ejecutar las secuencias de acciones necesarias para satisfacer los requerimientos de cada servicio contratado. La secuencia de acciones (plan detallado) involucra la asignación de los recursos a lo largo del tiempo, hasta la completa ejecución del plan [8].

Este sistema también debe supervisar la ejecución del plan, y en caso de alguna desviación, debe realizar dinámicamente las modificaciones al plan en ejecución (replanificación). Para resolver este problema, se puede plantear una descomposición funcional en tres bloques del sistema de planificación, como se muestra en la Figura 13.1, el cual fue definido en [2].

El primer bloque busca la información relevante, tanto interna como externa, para caracterizar el sistema. El segundo bloque es el núcleo del sistema, éste genera el plan general y el plan detallado; además, supervisa la ejecución del mismo, y realiza la modificación al plan en caso que sea necesario. El tercer bloque tiene, como principal tarea, la ejecución del plan detallado. Este sistema interactúa con el resto de los componentes del negocio (por ejemplo: sistemas de control, sistema de manejo de fallas, sistemas de inventario, etc.), y demanda acciones sobre el proceso. Cada bloque del sistema está compuesto por diferentes módulos que interactúan entre si, siguiendo una secuencia determinada por las reglas del negocio, según el modelo de referencia que se propone en la Figura 13.2.

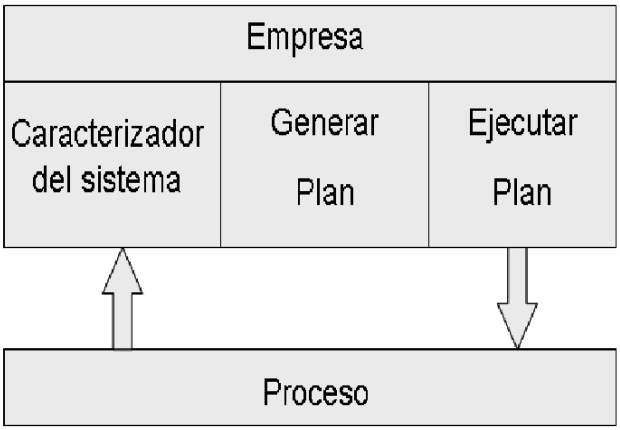

Figura 13.1: Módulos de Planificación

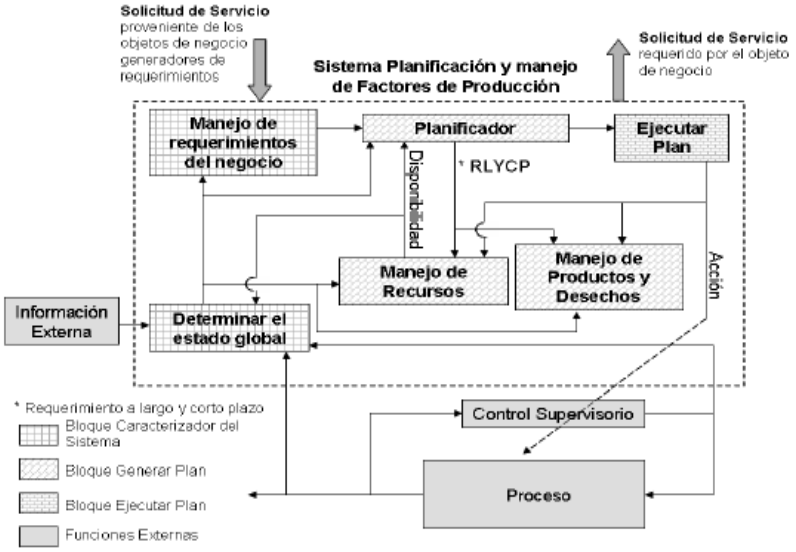

Figura 13.2: Modelo de Referencia

En las siguientes secciones se presentará la descripción de los bloques que se muestran en la Figura 13.2.

13.3.1. Caracterizador del sistema

Está compuesto por dos módulos: Determinar el estado global y Manejo de requerimiento del negocio.

13.3.1.1. Módulo "Determinar el estado global"

Se encarga de obtener y procesar toda la información requerida para la elaboración del plan. El estado del entorno se define para cada área del proceso productivo y puede estar compuesto por los estados de las variables internas del proceso y variables externas al mismo. Las variables externas pueden variar como resultado de los cambios ocurridos en el mercado, mientras las variables internas varían debido a cambios en las variables de entrada (calidad, volumen), fallas en los equipos, fluctuación en el suministro de materia prima y energía, etc. [2, 9]. En general, las variables de interés utilizadas para determinar el estado global actual para la elaboración del plan general, el plan detallado, y el seguimiento del plan son las siguientes:

- Demanda actual: es la cantidad de órdenes de producción recibidas por el objeto de negocio.
- Demanda estimada: es la cantidad de órdenes de producción que estima recibir el objeto de negocio. La demanda estimada es proporcional a los cambios que puedan tener, o se estima que tengan, algunas variables externas al proceso de producción, tales como: ofertas del mercado; cantidad de competidores; niveles de precios del producto en el mercado; niveles de precios de los recursos en el mercado; situación socio-política del estado; situación socio-política a nivel mundial, sobre todo en los clientes y competidores potenciales, situación socio-política de los países suministradores de insumos y/o materias primas; condiciones atmosféricas, etc.
- Estado de las órdenes de producción: cada orden de producción al ser recibida por el objeto de negocio puede tener diferentes estados. Entre los posibles estados se tienen: aceptada, rechazada, en espera, en ejecución, completada, etc.
- Capacidad de producción: es la mayor cantidad sostenible de producto (productos finales y/o productos intermedios) que puede generar un objeto de negocio. La capacidad de producción es directamente proporcional a la capacidad y disponibilidad de los recursos de producción.
- Capacidad de producción comprometida: es la porción de la capacidad de producción que esta actualmente en uso o programada para su uso. La capacidad de producción comprometida es directamente proporcional a las demandas recibidas y aceptadas por el objeto de negocio.
- Capacidad de producción desatendida: es la capacidad de producción que no puede ser atendida debido a factores internos, tales como, indis-

ponibilidad de los equipos, planes de producción deficientes, limitación en los recursos, etc.
- Capacidad de producción disponible: es la porción de la capacidad de producción que puede ser atendida y no se encuentra comprometida. La capacidad de producción disponible es directamente proporcional a la capacidad de producción, e inversamente proporcional a la capacidad de producción comprometida o desatendida.
- Indicadores de rendimiento: permite determinar el desempeño del objeto de negocio (producción actual).
- Disponibilidad de recursos.
- Disponibilidad de productos para la venta y/o uso de otros objetos del negocio.

13.3.1.2. Módulo "Manejo de requerimiento del negocio"

Los objetos de negocio pueden asumir, dentro del proceso productivo, el rol de objeto generador de requerimientos (el objeto solicita uno o más servicios a uno o más objetos de negocio), objeto oferente de servicios, o ambos roles. El término requerimiento del negocio se refiere a los servicios prestados o recursos generados por los objetos de negocio. En caso que el objeto sea generador de requerimientos, el módulo de manejo de requerimientos del negocio debe procesar las órdenes emitidas por el módulo planificador para la contratación de los servicios requeridos por el objeto de negocio, con el fin de garantizar la completa ejecución de sus funciones y el logro de las metas. En caso que el objeto sea oferente de algún servicio, el módulo de manejo de requerimientos del negocio debe procesar las órdenes recibidas desde los objetos generadores de requerimientos, con el fin de aceptar o rechazar las solicitudes de servicio, realizar ofertas, y analizar los resultados de la negociación.

13.3.2. Generar Plan

Este bloque está compuesto por tres módulos: Planificador, Manejo de recursos, y Manejo de productos.

13.3.2.1. Módulo "Planificador"

Se encarga de la elaboración del plan general y el plan detallado del objeto de negocio, a partir de los requerimientos del negocio, modelos del proceso productivo, mecanismos de producción y/o reglas del negocio, métodos

de optimización, estado global, predicciones y/o estimaciones, entre otras especificaciones.

El plan general se refiere a la determinación de las metas de producción del objeto de negocio en función del tiempo (el periodo de tiempo suele llamarse horizonte de planificación), indicando: qué se va a producir, en que cantidad, quién requiere el producto, y cuándo lo requiere. El plan detallado debe generar la secuencia de actividades que debe ejecutar el objeto de negocio a lo largo del tiempo, con el fin de cumplir con las metas de producción establecidas en el plan general. El plan detallado debe al menos establecer:

- Los ajustes (*set-point*) que deben aplicarse a los diferentes componentes del proceso productivo del objeto de negocio, el cuál es asignado al sistema de control y supervisión para los ajustes correspondientes en el nivel proceso.
- La solicitud, asignación, y mecanismos para la entrega y recepción de recursos.
- El programa detallado para la entrega de productos.
- La especificación del mecanismo o estrategia a ser utilizada en cada una de las tareas de producción.

En este módulo también se supervisa la ejecución del plan, y en caso que sea necesario, se realizan las modificaciones al mismo.

13.3.2.2. Módulo "Administrador de recursos"

Se encarga del control de inventarios de los recursos requeridos para la ejecución del plan, en base al modelo de inventario, plan general y plan detallado. Este módulo realiza la asignación de los recursos dentro del proceso productivo, y puede interactuar con las funciones de procura para realizar la solicitud de recursos y para conocer el estado las solicitudes.

13.3.2.3. Módulo "Manejo de productos y desechos"

Se encarga del control de la cuenta de activos circulantes de los productos terminados, en base al modelo de inventario de productos, plan general, y plan detallado. También debe realizar el manejo de los desechos resultantes durante el proceso productivo. Este módulo puede interactuar con el sistema contable para la facturación de productos terminados y entregados.

13.3.3. Ejecutar Plan

Este bloque está compuesto solamente por el módulo Ejecutar Plan.

13.3.3.1. Módulo Ejecutar plan

Se encarga de la ejecución del plan detallado. El plan detallado se define como una secuencia de acciones de control que serán ejecutadas por este modulo.

13.4. Diseño del SMA para el sistema de planificación

En esta sección se desarrolla el SMA del sistema de planificación de la producción propuesto en [2]. Se usan algunos componentes de la metodología MASINA para la descripción del SMA [1]. Además, se parte del supuesto que se tiene un diseño arquitectónico del proceso de automatización basado en agentes [7, 8, 11], tal que el SMA responsable del proceso de planificacion, según el modelo ISA/ANSI, debe comunicarse con los agentes responsables de las tareas de control [3, 5, 13], de supervisión [5, 8], de manejo de fallas [10], y de manejo de las situaciones anormales que se presenten en el sistema [4].

En este primer enfoque se sigue un esquema de planificación centralizado para planes distribuidos (ver la Sección 2.5.4 para más detalles), ya que este SMA es responsable de la tarea de gestión de todo el proceso de planificación en la comunidad de agentes del entorno de automatización. Es decir, este SMA realiza la planificación del proceso de automatización [2, 7, 11], y envía los planes generales a llevar a cabo por el resto de los agentes (de control, de supervisión, de manejo de fallas, etc.). Estos otros agentes elaborarán sus programas específicos de acción, a partir de las consignas recibidas en los planes generales.

13.4.1. Características generales

El SMA para la planificación debe tener la capacidad de ejercer principalmente las siguientes actividades [2, 3, 7, 11, 12]:

- Intercambiar información entre los diferentes niveles de la pirámide de automatización ISO/OSI.

- Monitorear y analizar las variables internas (provenientes desde los diferentes niveles), y externas al proceso, con el fin de determinar el estado global para la elaboración del plan, o la replanificación del plan existente.
- Elaborar el plan general del objeto de negocio.
- Ejecutar el plan detallado del objeto de negocio.
- Ejecutar acciones sobre el plan, y en caso que sea necesario, obtener un nuevo plan de acuerdo al análisis del estado global y contratos recibidos (planificación dinámica).
- Gestionar los recursos asociados al plan.
- Gestionar los productos terminados.
- Estimar y predecir las variables y condiciones que puedan impactar al plan, y por ende, el desempeño del objeto de negocio.

13.4.2. Actores del sistema de planificación

En la Tabla 13.1 se presentan los actores identificados en el sistema de planificación que se esta estudiando.

Actor	**Descripción**	**Función**
Caracterizador	Recibe el estado de las variables internas y externas al proceso y determina el estado global.	Determinar estado global
Negociador	Emite, recibe y procesa los requerimientos del negocio. Gestión de requerimientos recibidos	Gestionar requerimientos a ser emitidos
Planificador	Elabora el plan general del objeto de negocio tomando en cuenta los requerimientos del negocio, el estado global, las estimaciones y/o predicciones, etc. También puede redefinir el plan general en caso que sea necesario. Para la elaboración del plan general, el planificador puede realizar requerimientos a otros objetos de negocio	Elaborar plan general, Supervisar plan general, Redefinir plan general, Generar solicitud

Continua en la próxima página

Tabla 13.1 –Finaliza de la página anterior

Programador	Elabora el plan detallado (programación de actividades) del objeto de negocio en base al plan general, los mecanismos de producción o reglas del negocio, los métodos de optimización, el estado global, etc. También puede redefinir el plan detallado, en caso que sea necesario.	Elaborar plan detallado Supervisar plan detallado Redefinir plan detallado
Administrador de Recursos	Se encarga del control de inventarios de los recursos y la gestión para la adquisición y asignación de los mismos dentro del proceso. Control de inventarios de recursos Manejador de Productos Se encarga del manejo, almacenamiento y distribución de los productos terminados	Control de inventarios de productos
Manejador de Desechos	Se encarga del manejo de los desechos de acuerdo a las normas de Seguridad, Higiene y Ambiente.	Manejo de desechos
Predictor	Realiza las estimaciones a futuro de las variables requeridas por el planificador y el administrador de recursos	Capacidad de producción, Requisiciones por recibir, Restricciones, Suministro de recursos
Ejecutor	Ejecuta el plan de producción y notifica el estado de ejecución del mismo Procesar plan detallado	Reportar estado del plan

Tabla 13.1: Actores del Sistema de Planificación

13.4.3. Modelo de Agente

Con los actores identificados en la fase anterior, se determinan los agentes que dotan al SMA de planificación y manejo de los factores de producción con las funcionalidades que permitan ejecutar sus actividades. Cada actor identificado en la fase de conceptualización puede ser un agente, o puede ser dividido en más de un agente. Así pues, se definen nueve agentes a los cuales hemos denominado como [2]:

- Agente Caracterizador.
- Agente Negociador.
- Agente Planificador.
- Agente Programador.
- Agente Administrador de Recursos.
- Agente Manejador de Productos.
- Agente Manejador de Desechos.
- Agente Ejecutor.
- Agente Predictor.

En la Figura 13.3, se presenta el esquema del SMA de planificación interactuando con otros SMA que ejecutan otras actividades de automatización industrial [4, 5, 3, 10, 11]. Los SMA interactúan entre ellos a través del sistema gestor de servicio como el descrito en el Capítulo 5.

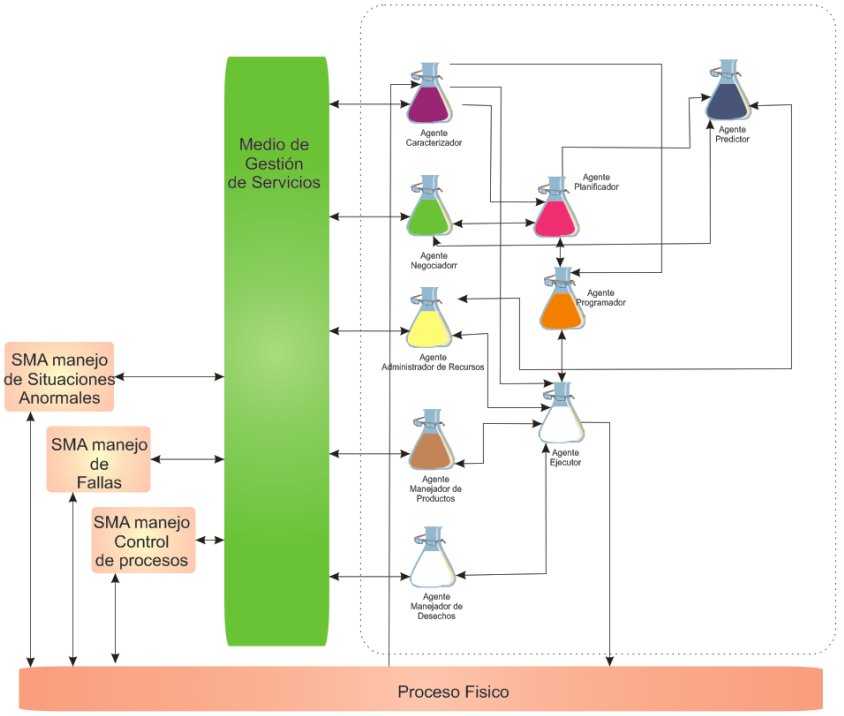

Figura 13.3: SMA de planificación

13.4.4. Modelo de Tareas

Siguiendo la metodología MASINA, a continuación se prsenta, de manera reducida, el diseño del SMA para la planificación de producción industrial.

En la Tabla 13.2 se presenta el modelo de tareas, las cuales debes ser ejecutadas por los agentes definidos en la sección anterior.

Agentes	Tareas
Caracterizador	Obtener el estado de las variables internas, Obtener el estado de las variables externas, Pedir Datos, Determinar el estado global actual, Determinar indicadores de gestión del plan general
Negociador	Elaborar oferta de negocio, Analizar resultado de la negociación, Elaborar requerimiento de negocio, Seleccionar mejor oferente
Planificador	Evaluar el estado global, Definir solicitud de servicio, Obtener plan general de producción, Detectar desviaciones del plan general, Modificar plan general de producción
Programador	Evaluar estado y disponibilidad de recursos, Evaluar estado y disponibilidad de productos, Evaluar estado de desechos, Evaluar mecanismos de producción, Obtener plan detallado de producción, Determinar desviación del plan detallado, Ajustar el plan detallado de producción
Administrador de Recursos	Generar solicitud para la adquisición de recurso, Generar orden de asignación de recurso, Ingresar recurso al sistema, Control del nivel de inventario de recursos
Manejador de Productos	Generar orden para la asignación y despacho de producto, Control del inventario de productos
Manejador de Desechos	Determinar mecanismo para el manejo de desechos, Generar orden para el manejo de desechos
Ejecutor	Ejecutar acción del plan detallado, Determinar el estado y resultados del plan detallado
Predictor	Estimar capacidad de producción a futuro, Estimar demanda (requerimientos por recibir), Estimar restricciones, Estimar fecha de ingreso de recursos

Tabla 13.2: Agentes del Sistema de Planificación

13.4.5. Modelo de Coordinación

Con el modelo de coordinación se describe el esquema de comunicación del SMA; es decir, la conversación, los protocolos y los lenguajes asociados. Para describir la conversación se utiliza el diagrama de interacción UML. Para el problema de planificación se han definido las siguientes conversaciones [2]:

- Diseño del plan general de producción.
- Modificar el plan general de producción por bajo desempeño.
- Recibir requerimientos del negocio.
- Realizar requerimiento del negocio.
- Diseñar el plan detallado.
- Modificar el plan detallado.
- Asignar recurso al plan general.
- Asignar recurso al plan detallado.

- Recibir recurso.
- Entregar producto.
- Eliminar desecho.
- Obtener estimados.
- Ejecutar acción en entidad externa.
- Ejecutar acción en otros sistemas automatizados.
- Recibir información desde otros sistemas automatizados.

A continuación mostramos el diagrama de secuencia UML de la conversación *Diseño del plan detallado*, en la Figura 13.4.

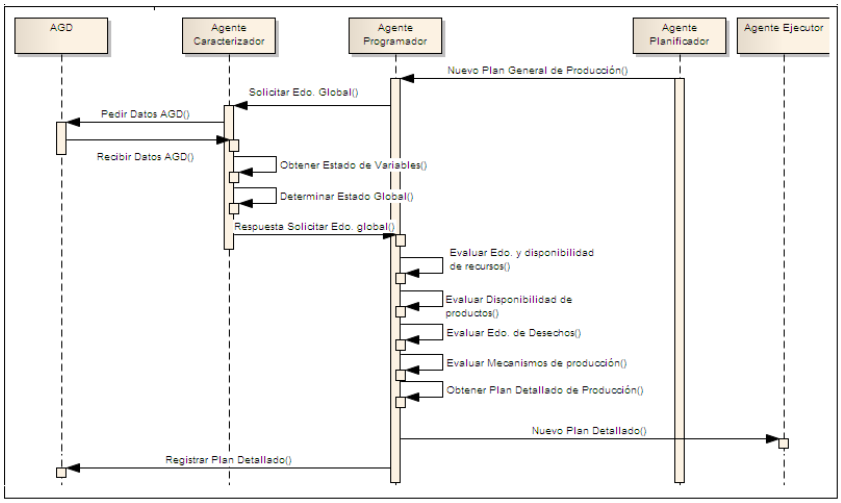

Figura 13.4: Diagrama de secuencia UML de la conversación *Diseño del plan detallado*

El objetivo de esa conversación es diseñar el plan detallado (la estrategia) que asegure cumplir con las metas del plan general de producción. Los agentes que intervienen son: Agentes Planificador, Programador, Ejecutor y el Agente Gestor de Datos (AGD), que forma parte del MGS. En esta conversación se obtiene la combinación, secuencias, y tiempo en la que se deben ejecutar las actividades que aseguren alcanzar la meta del nuevo plan general de producción. El resto de conversación son descritos de la misma manera en [2].

13.5. Otro enfoque de sistema de planificación en automatización

En [2] se propone que los agentes de cada nivel negocien entre si para llegar a acuerdos que permitan cumplir con las metas de producción establecidas; dichos acuerdos representan la lógica del negocio que rige el proceso productivo. Ahora bien, en muchos procesos industriales tal negociación no ocurre en todos los niveles; por ejemplo, la asignación de recursos para cumplir con unas metas establecidas pueden ocurrir en el nivel superior. En virtud de lo anteriormente expuesto, en [12] se propone una nueva descomposición del plan de negocio, comenzando con el plan global de la empresa hasta llegar al plan del lazo de producción. De esta manera, lo que se propone es realizar una modificación a la última fase de construcción del plan global del negocio, tal que se realiza el plan de negocio para el grupo de elementos del lazo de producción como un todo, y no individualmente para cada objeto de negocio. Además, este plan lo realizarían los agentes del nivel superior.

En este caso, se propone realizar un tipo de planificación jerárquica descentralizada para planes distribuidos para el SMA, mucho más fácil de gestionar que la precedente (ver sección 2.5.4 y [15, 14]). Es decir, básicamente se propone que los niveles superiores generen la consigna de planificacion para los niveles inferiores, y a partir de allí, estos niveles realicen su planificación y generen la consigna para los niveles inferiores a ellos. Esto ocurre recursivamente en todo el árbol jerárquico (ver Figura 13.5). Ahora bien, en el último nivel no se construye ningún plan (lazo de producción), sino que es el nivel superior quien lo hace (sólo aquí se usa un enfoque de planificación centralizada para planes distribuidos).

13.5.1. Descripción del Modelo

Para este caso, se propone un modelo funcional para el proceso de planificación de la producción conformado por dos bloques, como se muestra en la Figura 13.6 [12]. El primer bloque corresponde al manejo y gestión de los requerimientos, recursos y productos vinculados al proceso de planificación. El segundo bloque es el núcleo del sistema, responsable de llevar a cabo el proceso de planificación. Cada bloque está conformado por varios módulos, donde cada uno de ellos es responsable de realizar tareas específicas dentro del proceso de planificación.

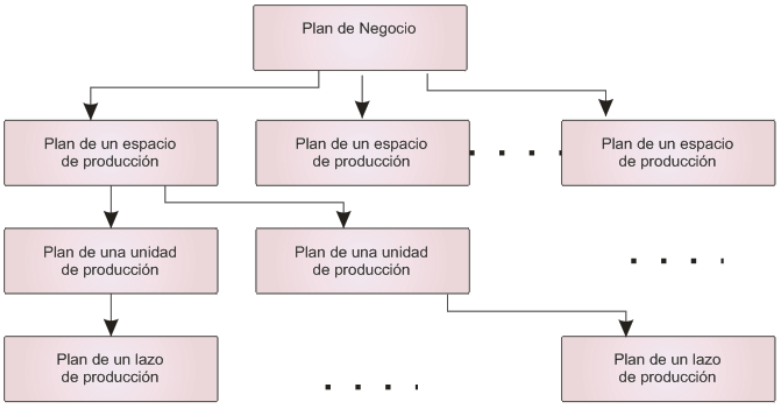

Figura 13.5: Esquema de planificación

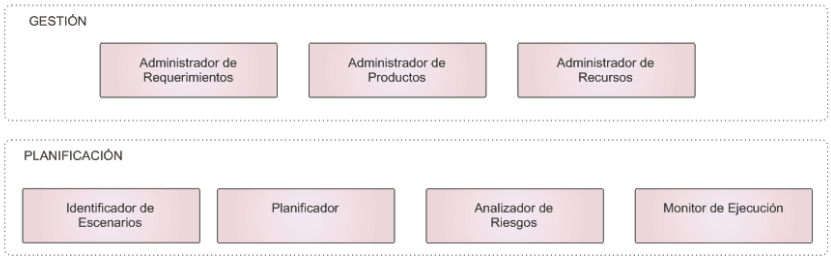

Figura 13.6: Enfoque alternativo de planificación

13.5.1.1. Bloque de Gestión

Este bloque está conformado por tres módulos, para gestionar toda la información requerida por el proceso de planificación:

- **Administrador de requerimientos**: debe procesar las órdenes recibidas desde los entes externos generadores de requerimientos, con el fin de evaluar las solicitudes realizadas, a fin de aceptarlas para procesarlas, o rechazarlas.
- **Administrador de recursos**: se encarga del control de inventarios de los recursos disponibles y requeridos para la ejecución del plan, en base al modelo de inventario y al plan de producción. Este módulo realiza la asignación de recursos dentro del proceso productivo, y puede interac-

tuar con las funciones de procura para realizar la solicitud de recursos y para conocer el estado de las solicitudes.

- **Administrador de productos**: se encarga del control de los productos terminados, en base al modelo de inventario de productos y al plan de producción. También debe realizar el manejo de los desechos resultantes durante el proceso productivo. Este módulo puede interactuar con el sistema contable para la facturación de productos terminados y entregados.

13.5.1.2. Bloque de Planificación

Este bloque está conformado por cuatro módulos:

- **Identificador del Escenario**: encargado de analizar y procesar toda la información referente al escenario actual de producción. El estado del entorno se define para cada área del proceso productivo, y puede estar compuesto por los estados de las variables internas del proceso y variables externas al mismo. Las variables externas pueden variar como resultado de los cambios ocurridos en el mercado, mientras las variables internas varían debido a cambios en las variables de entrada al proceso productivo, fallas en los equipos, fluctuación en el suministro de materia prima y energía, etc.
- **Planificador**: este módulo se encarga de la elaboración del plan de producción a partir de las metas de producción establecidas, los modelos del proceso productivo, los mecanismos de producción y reglas del negocio, los métodos de optimización, el escenario actual, y las predicciones y/o estimaciones del mercado. El plan deberá establecer:

  - La cantidad de producto que se deberá producir, y sus lapsos de entrega, para cumplir con la meta total al finalizar el horizonte de planificación.
  - Los recursos requeridos para cumplir con las metas de producción.
  - La planificación de las actividades de producción y mantenimiento.
  - La especificación de la estrategia a ser utilizada para la ejecución del plan.

  Así mismo, este módulo elabora la programación de actividades. Una vez establecido el plan de producción, el mismo se debe operacionalizar, esto es, se deben definir las tareas específicas a ejecutar y la asignación de los recursos en los tiempos adecuados en un nivel más detallado. De esta manera, la programación de actividades por su parte deberá establecer:

  - La secuencia de actividades de producción por cada unidad componente del proceso productivo.
  - Los puntos de ajustes (*setpoint*) y las estrategias de control de procesos que deben aplicarse a los diferentes componentes del proceso productivo.

  - La solicitud, asignación y mecanismos para la entrega y recepción de recursos.
  - La programación de paros de unidades y de actividades de mantenimiento.

- **Analizador de riesgos**: estudia la probabilidad de que los resultados previstos se produzcan, o bien, determina los valores esperados de los indicadores económicos a partir de la producción actual. Se incluye también, en este módulo, el análisis de sensibilidad, el cuál consiste en determinar variaciones en los indicadores económicos al modificar alguna de las siguientes variables: precios, volúmenes, inversión, costos de operación, etc.
- **Monitor de la ejecución**: una vez determinado el plan de producción y la programación de actividades, se procede a su ejecución. Durante este tiempo es necesario contar con un módulo encargado de monitorear el desempeño del plan, a fin de poder detectar posibles desviaciones que se produzcan, y alertar a tiempo al módulo encargado de realizar los ajustes pertinentes y de replanificar, si fuese necesario.

13.5.2. Arquitectura del modelo multiagente

A partir de las funciones que debe desempeñar el sistema de planificación de la producción, y en concordancia con cada uno de los módulos descritos antes, podemos identificar los agentes que componen el modelo propuesto para la planificación de la producción [12]. Dichos agentes son los responsables de interactuar y ejecutar la secuencia de acciones determinadas por las reglas del negocio que rigen el proceso productivo, a fin de llevar a cabo la planificación de la producción, según la arquitectura que se presenta en la Figura 13.7, propuesta en [12].

En esta arquitectura se proponen los siguientes agentes:

- **Agente Administrador de Requerimientos (AAR1)**: responsable de recibir los requerimientos generados por los clientes, para analizarlos y determinar si pueden ser atendidos. Además, este agente define los objetivos o metas de producción que deben alcanzarse para satisfacer dichos requerimientos.
- **Agente Administrador de Productos (AAP)**: su función principal es la administración y gestión del producto disponible y generado como resultado del proceso productivo. Es responsable de controlar el inventario, validar que los productos generados cumplan con las especificaciones de calidad, y del manejo y despacho de los productos.
- **Agente Administrador de Recursos (AAR2)**: encargado de la administración de los recursos disponibles y necesarios para llevar a cabo la ejecución de los planes. Este agente se encarga de la localización y asigna-

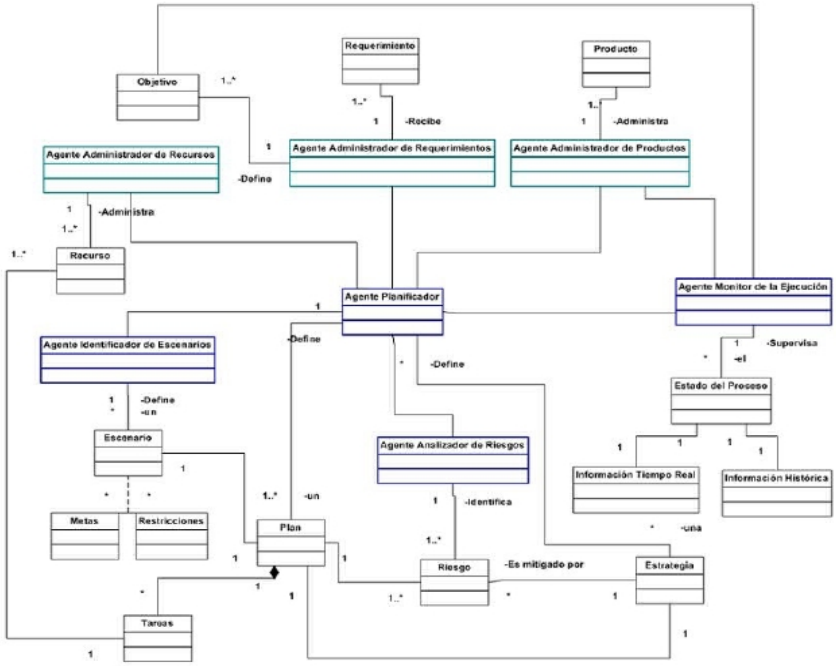

Figura 13.7: Arquitectura de planificación

ción de los recursos requeridos en cada tarea definida en el plan de producción, esta asignación de recursos la realiza en conjunto con el agente planificador.

- **Agente Planificador (AP)**: este agente, en función de las metas u objetivos establecidos por el agente administrador de requerimientos, es responsable de la construcción del plan, es decir, identifica las macro-tareas a realizar para alcanzar dichas metas. Esta fase corresponde a la planificación de las actividades. Por otra parte, este agente también es responsable de programar la realización de las actividades identificadas como parte del plan, y de detallarlas. Si fuese necesario, para esto deberá ubicar en el tiempo (horizonte de planificación) la asignación de los recursos necesarios para llevar a cabo dichas tareas. Para esto último, el agente planificador solicita al agente administrador de recursos la asignación de los recursos requeridos en cada tarea, de forma de garantizar una exitosa consecución del plan. En algunos casos, dependiendo del nivel donde este trabajando este agente, deben realizar sólo tareas de planificación (ejemplo, en los niveles de toma de decisiones) o de programación (ejemplo, en los niveles operacionales). Este agente también tiene como fun-

ción identificar la estrategia a seguir para mitigar los riesgos asociados al plan, identificados por el agente analizador de riesgos, y es responsable de realizar ajustes al plan en caso de ser necesario.

- **Agente Identificador de Escenario (AIE)**: este agente caracteriza en cual escenario se encuentra el proceso productivo, en función de su estado actual, las metas y restricciones existentes, etc. La información generada por este agente es utilizada como insumo por el agente planificador para la construcción del plan.
- **Agente Analizador de Riesgos (AAR3)**: es responsable de identificar los riesgos asociados al plan, analizar sus relaciones y posibles consecuencias. Para lograr su objetivo, este agente realizará cambios en variables claves del proceso, con el fin de identificar si el plan se ve muy afectado por dichos cambios, y de esta manera reconocer las posibles consecuencias que esto originaría en el plan (es decir, hace un análisis de sensibilidad). Finalmente, deberá informar al agente planificador para que defina la estrategia de mitigación de riesgos.
- **Agente Monitor de la Ejecución (AME)**: actúa como observador de la ejecución del plan. Su función es observar el estado del proceso (definido a partir de la información en tiempo real e información histórica), validar que los recursos se están asignando de manera oportuna, y que las tareas se están ejecutando según lo planificado. Este agente debe informar al agente evaluador de escenario el estado de la ejecución del plan.

Con la arquitectura que se propone en [12], lo que se busca es proporcionar un modelo funcional de planificación de la producción basado en agentes, que pueda ser utilizado para apoyar las labores de planificación en cada uno de los niveles presentes dentro de la empresa (gerencial, de optimización, de supervisión, de control del proceso, etc.). Cada nivel del proceso de planificación estará diferenciado, básicamente, por:

- El horizonte de planificación considerado.
- El nivel de detalle de la información utilizada.
- Los requerimientos y metas establecidas.
- Las premisas y los parámetros considerados.
- La naturaleza de las tareas planificadas.
- El tipo de riesgos y estrategias utilizadas.

Todo ello haciendo uso, en cada nivel, del mismo esquema multiagente de planificación.

13.5.3. Funcionamiento del SMA para la planificación de la producción

A continuación se muestra un diagrama de actividad presentado en [12], que ilustra el flujo básico general a realizar por el SMA durante la construc-

ción, ejecución y monitoreo del plan de producción. El color asignado en el diagrama a cada actividad permite identificar el agente responsable de su realización (Figura 13.8):

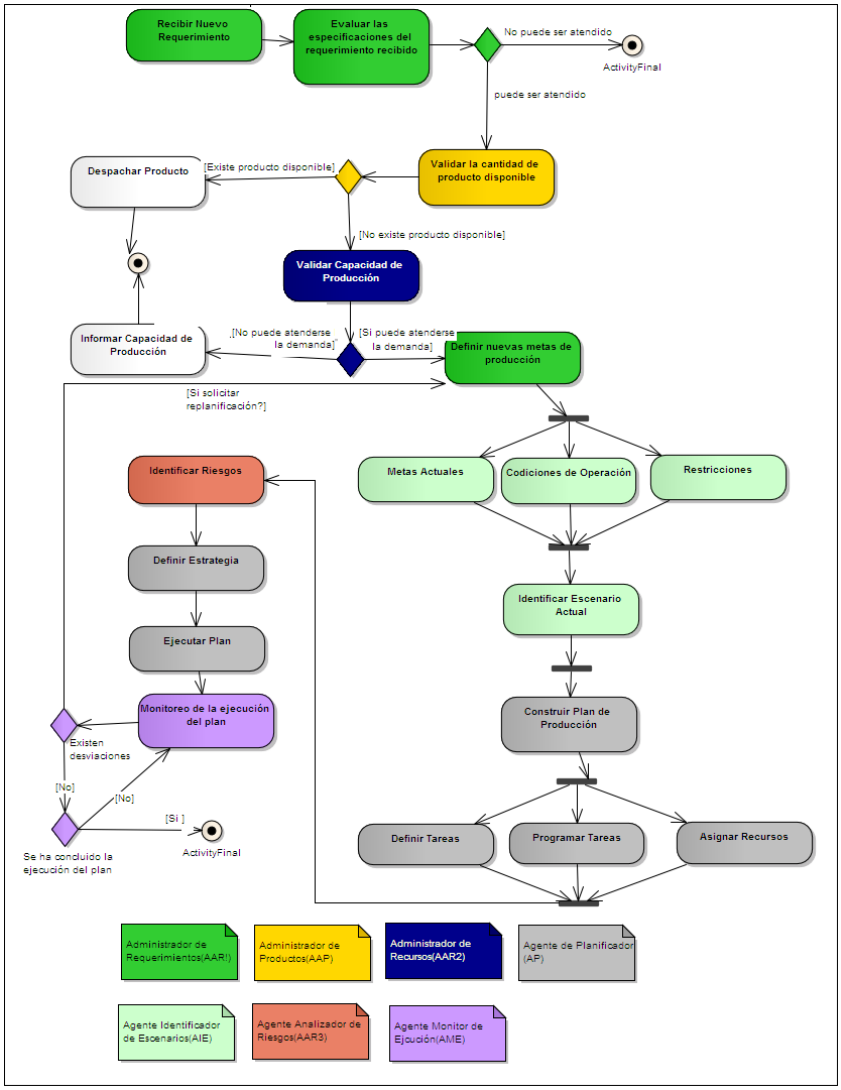

Figura 13.8: Diagrama de actividades de Planificación

En el diagrama se puede observar que el flujo se inicia con la recepción de un nuevo requerimiento por parte del AAR1, en este caso asumiremos que el requerimiento corresponde a una solicitud de determinada cantidad

de producto con ciertas especificaciones. El AAR1 analiza el requerimiento recibido y determina si puede ser atendido, es decir, evalúa que el requerimiento se encuentre dentro de los parámetros y especificaciones que pueden ser atendidas por el sistema de producción. Si el requerimiento puede satisfacerse, el AAP valida la cantidad de producto disponible, si existe producto suficiente se notifica al módulo correspondiente para que inicie las actividades de despacho del producto. En caso contrario, el AAR2 valida la capacidad real de producción del sistema, es decir calcula la capacidad nominal menos la capacidad comprometida para determinar si puede atenderse la demanda en el tiempo establecido. Sino puede atenderse la demanda este agente se encarga de informar la capacidad real de producción. En caso contrario, informa al AAR1, quien define las nuevas metas de producción y notifica al AIE para que realice la categorización del escenario actual en función de las metas actualizadas, las condiciones de operación, y las restricciones existentes. El escenario, aquí definido, creará el insumo para el AP, quién deberá construir el plan, es decir, definir las macro-tareas, así como programar su ejecución en el tiempo, detallarlas en caso de ser necesario, y asignarle los recursos necesarios con el fin de alcanzar las metas. Luego que el plan es construido, el AAR3 identifica los riesgos asociados al plan e identifica sus posibles consecuencias. Esta información es utilizada por el AP para definir las estrategias necesarias para mitigar los riesgos. Finalmente, el plan es ejecutado y el AME se encarga de observar dicha ejecución. De producirse una desviación significativa, la notifica al agente planificador para que realice una replanificación en las actividades que sea necesario.

Este flujo se puede aplicar, de manera general, a los distintos niveles de planificación de la empresa (ver la Figura 13.5), es decir, el SMA mantendrá el mismo esquema de planificación recursivamente, independientemente del nivel donde se implemente. Como se ha dicho antes, la diferencia en la actuación del sistema para cada nivel radicará principalmente en la granularidad de la información y en el horizonte de planificación manejado en cada nivel. Adicionalmente, las salidas de la planificación en un nivel superior representan las entradas de un nivel inferior.

Referencias

1. J. Aguilar, I. Besembel, M. Cerrada, F. Hidrobo, and F. Narciso. Una metodología para el modelado de sistemas de ingeniería orientado a agentes. *Revista Iberoamericana de Inteligencia Artificial, AEPIA*, 12(38):39–60, 2008.
2. J. Aguilar, M. Cerrada, F. Hidrobo, J. Chacal, and Bravo C. Specification of a multiagent system for planning and management of the production factors for automation based on the scdia framework and masina methodology. *WSEAS Transactions on Systems and Control*, 3(2):79–88, 2008.
3. J. Aguilar, M. Cerrada, G. Mousalli, F. Rivas, and F. Hidrobo. A multiagent model for intelligent distributed control systems. *Lecture Notes in Artificial Intelligence, Springer-Verlag*, 3681:191–197, 2005.

4. J. Aguilar, C. Prato, Bravo C., and F. Rivas. A multi-agent system for the management of abnormal situations in an artificially gas-lifted well. *Applied Artificial Intelligence*, *Taylor and Francis*, 23(5):406–426, 2009.
5. J. Aguilar, A. Rios, F. Hidrobo, M. Cerrada, and J. Duran. Control and supervision system development with intelligent agents. *WSEA Transactions on Systems*, 6(1):141–149, 2007.
6. ANSI/ISA. Ansi/isa95.00: Modelo funcional del flujo de datos. Technical report, ISA, 1995.
7. C. Bravo, J. Aguilar, F. Rivas, and M. Cerrada. Design of an architecture for industrial automation based on multi-agents systems. In *Proceeding of the 16th IFAC World Congress*, Praga, Republica Checa, Julio 2005. IFAC.
8. César Bravo, Jose Aguilar Castro, Addison Ríos, Joseph Aguilar-Martin, and Francklin Rivas. Arquitectura basada en inteligencia artificial distribuida para la gerencia integrada de producción industrial. *Revista Iberoamericana de Automática e Informática Industrial*, 8(4):1–15, 2011.
9. César Bravo, Jose Aguilar Castro, Luigi Saputelli, Addison Ríos, Joseph Aguilar-Martin, and Francklin Rivas. An implementation of a distributed artificial intelligence architecture to the integrated production management. *Journal of Natural Gas Science and Engineering*, 0(0):–, 2011.
10. M. Cerrada, J. Cardillo, J. Aguilar, and R. Faneite. Agents-based design for fault management systems in industrial processes. *Computer in Industry*, 58:313–328, 2007.
11. F. Hidrobo, A. Ríos, L. León, and J. Aguilar. An architecture for industrial automation based on intelligent agents. *WSEAS Transactions on Computers*, 4(12):1808–1815, 2005.
12. F. Martínez, C. Bravo, and J. Aguilar. Sistema multiagente para la planificación de la producción en automatización. In *Proceeding of the XIII Congreso Latinoamericano de Control Automático*, pages 959–966, Mérida, Venezuela, Noviembre 2008. IFAC.
13. A. Ríos, M. Cerrada, F. Narciso, J. Aguilar, and F. Hidrobo. Implantando sistemas de control con agentes inteligentes. *Revista Ciencia e Ingeniería*, 29(3):249–260, 2008.
14. G. Weiss. *Multi- agent System: a modern approach to distributed artificial intelligence*. MIT Press, 1999.
15. M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, second edition edition, 2009.

GLOSARIO

Ontología: es el conjunto de términos y sus relaciones que componen el vocabulario de un área dada, así como las reglas para combinar esos términos y relaciones, que definen extensiones a ese vocabulario. Por lo tanto, es un sistema caracterizado por conceptos (o un vocabulario) y relaciones específicas entre ellos, usado como elemento básico (primitivo) para la construcción de sistemas basados en el conocimiento.

Hilo de Ejecución: es la unidad de procesamiento más pequeña que puede ser planificada por un sistema operativo. Así, es una característica que permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente, mediante tareas que puede ser ejecutada en paralelo.

Contenido Semántico: se refiere a la existencia de información inteligible para el receptor, capaz de evocar la representación de un referente, y es directamente evaluable en términos semánticos (tiene su propio sentido, valores veritativos, etc.). En ese contexto, se basa en la idea de añadir metadatos semánticos y ontológicos a la información y conocimiento. Esas informaciones adicionales, que describen el contenido, el significado y la relación de los datos que componen a la información/conocimiento; se deben proporcionar de manera formal, para que así puedan ser usadas automáticamente en el proceso de interpretación de dicha informacion/conocimiento.

Problema NP-Completo: es cualquier clase de problema de cálculo para el cual no se ha encontrado un algoritmo que de una solución eficiente, por poseer un costo factorial o combinatorio, pero que podrían ser resueltos por algoritmos no-deterministas.

Inteligencia Artificial: es la disciplina de las ciencias computacionales, que se encarga de construir procesos, definidos como agentes, que al ser ejecutados sobre una arquitectura física, producen acciones o resultados que maximizan una medida de rendimiento determinada, basándose en la secuencia de entradas percibidas y en el conocimiento almacenado en tal arquitectura.

Inteligencia Artificial Distribuida: es un campo de la Inteligencia Artificial dedicado al estudio de las técnicas y métodos, y el conocimiento necesario para la coordinación y distribución de información (ó conocimiento), y las acciones en entornos multiagentes.

Holón: es un sistema o fenómeno que es un todo en sí mismo así como es parte de un sistema mayor. Cada sub-sistema puede considerarse un holón.

Agente Inteligente: es una entidad capaz de percibir su entorno, procesar tales percepciones y responder o actuar en su entorno de manera proactivamente racional.

Control Automático: es el mecanismo, construido sobre innovaciones tecnológicas, que permite que los procesos técnicos operan en un marco de equilibrio estable de seguridad y productividad.

SCADA: es el acrónimo de *Supervisory Control And Data Acquisiton* (Control Supervisorio y Adquisición de Datos). Constituye un sistema basado en redes de computadores que permite supervisar y controlar, a distancia, variables de proceso, proporcionando comunicación con los dispositivos de campo (controladores autónomos) y controlando el proceso de forma automática por medio de un software especializado. También provee de toda la información que se genera en el proceso productivo a diversos usuarios, mediante una interfaz gráfica, tanto del mismo nivel como de otros usuarios supervisores dentro de la empresa (supervisión, control calidad, control de producción, almacenamiento de datos, mantenimiento, etc.).

Objeto de Negocio: es un tipo de entidad inteligible que es un actor dentro de la capa de negocio de un sistema computacional de múltiples capas. Representa una abstracción que tiene sentido para los actores de una organización, de manera que se utiliza para describir las entidades manipuladas por los actores. Un objeto de negocio se caracteriza por un estado representado por una identidad, un propósito (o conducta), atributos y relaciones que tiene con otros objetos.

Planning: se refiere a un plan. Es un procedimiento o conjunto de acciones ordenadas coherentemente para conseguir un determinado objetivo a largo plazo. Representa un proyecto sistemático que se elabora para alcanzar el objetivo, de manera de dirigir y encauzar las acciones para obtener la meta establecida.

Scheduling: se refiere a una programación de actividades para un lapso de tiempo corto. Así, la programación (scheduling) es el proceso de organizar como ejecutar un conjunto de tareas que se deben realizar inmediatamente (cómo asignar los recursos, etc.).

Índice alfabético