UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

A R A G Ó N

PROTOTIPO DE UN SISTEMA MULT-AGENTE PARA INSTALACIÓN REMOTA DE SOFTWARE

TESIS PROFESIONAL

Que Para Obtener el Título de

INGENIERO EN COMPUTACIÓN

Presenta:

FRANCISCO JAVIER GONZÁLEZ BLANCAS

Director de Tesis:

Jesús Díaz Barriga Arceo

México 2004

VNIVERSIDAD NACIONAL AVTÓNOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN

JEFATURA DE CARRERA DE INGENIERÍA EN COMPUTACIÓN

OFICIO:ENAR/JACO/00287/04.

ASUNTO: Asignación de Jurado

LIC. ALBERTO IBARRA ROSAS
SECRETARIO ACADÉMICO
P r e s e n t e .

Por este conducto me permito presentar a usted el nombre de los profesores que sugiero integren el Sínodo del Examen Profesional del alumno FRANCISCO JAVIER GONZÁLEZ BLANCAS con el trabajo de tesis "PROTOTIPO DE UN SISTEMA MULTIAGENTE PARA INSTALACIÓN REMOTA DE SOFTWARE"

PRESIDENTE:	**M. EN C. JESÚS DÍAZ BARRIGA ARCEO**
VOCAL:	**ING. RICARDO GUTIÉRREZ OROZCO**
SECRETARIO:	**ING. ARTURO OCAMPO ÁLVAREZ**
SUPLENTE :	**ING. CÉSAR FRANCISCO GERMÁN ROSAS**
SUPLENTE:	**ING. RODOLFO VÁZQUEZ MORALES**

Quiero subrayar que el director de tesis es el **M. en C. Jesús Díaz Barriga Arceo**, el cual está incluido con base en lo que reza el reglamento de Exámenes Profesionales de esta Escuela.

Sin otro en particular, me es grato enviarle un cordial saludo.

A T E N T A M E N T E
"POR MI RAZA HABLARÁ EL ESPÍRITU"
San Juan de Aragón, Edo. de México, mayo 20 del 2004.
EL JEFE DE CARRERA

M. EN C. JESÚS DÍAZ BARRIGA ARCEO

c.c.p. Lic. Ma. Teresa Luna Sánchez.- Jefa del Departamento de Servicios Escolares.
M. en C. Jesús Díaz Barriga Arceo. Asesor.
Interesado.

JDA*vjd

Agradecimientos

En el transcurso para la realización de ésta tesis he recibido ayuda, apoyo y confianza de muchas personas a las que deseo expresar mi agradecimiento.

En primer lugar, quiero agradecer al M. en C. José Manuel Cervantes Martínez, quien me brindo la oportunidad de ingresar al proyecto de seguridad que se trabajó en el Instituto Mexicano del Petróleo y del cual forma parte el presente trabajo. A los doctores del CIC Matías y Leonid que fueron los que nos proporcionaron un taller de agentes que nos sirvió para introducirnos a la tecnología de agentes, también a los compañeros que de alguna forma me apoyaron e hicieron que el tiempo que pasé en el Instituto fuera agradable.

Quiero agradecer en especial, a mi compañero y amigo Jorge Martínez Rendón, quien siempre me ha apoyado y más aun durante la realización de éste trabajo con consejos de cómo abordar o tratar los temas, ya que también él desarrollo su trabajo de tesis referente a los sistemas multi-agentes, y al igual que yo iniciamos de cero para desarrollar nuestro respectivo trabajo.

Agradezco al M. en C. Jesús Díaz Barriga Arceo quien es mi director de tesis, y me ha orientado en la forma de continuar los temas del capitulado de la tesis, realizando sus observaciones para culminar de la mejor forma éste trabajo.

Gracias también al Ing. Rodolfo Vázquez Morales por sus comentarios que han enriquecido la tesis.

A mi tía Aida Baza al igual que a mi muy querida amiga Claudia Montejano González les agradezco el haberme ayudado a darle forma a la tesis, porque sin su ayuda el trabajo tendría un mayor número de errores.

Agradezco muchísimo a mi mamá, mi papá y mi hermano quienes siempre me han apoyado en mi desarrollo personal y académico, ya que sin él no hubiera llegado a esta etapa.

Resumen

Esta tesis define el análisis y diseño de un sistema basado en agentes para la instalación remota de software, que pretende implantarse en el Instituto Mexicano del Petróleo para resolver problemas como la disponibilidad de personal, la pérdida en el tiempo que se consume en el traslado del personal de soporte informático de su lugar de trabajo al área del usuario demandante de un servicio, para esto, se ha implementado un prototipo que pone a prueba el funcionamiento del sistema.

Para resolver esta problemática se pensó en la tecnología de agentes debido a que permite realizar aplicaciones autónomas, interactuar con su entorno, proporcionar flexibilidad a los sistemas, además de que, presenta un comportamiento dinámico en su ejecución y se integra fácilmente con otros sistemas.

Para llegar al análisis y diseño del sistema se han proporcionado los fundamentos del paradigma de agentes, como son la teoría, las arquitecturas, los lenguajes y los tipos de agentes. Después, se da un panorama general de las metodologías orientadas a agentes que tienen mayor aceptación para realizar el análisis, diseño y documentación. También se revisan algunas herramientas para crear sistemas multi-agentes, de las cuales se tiene que emplear una para probar el análisis realizado. Luego se presenta propiamente el modelo conceptual del sistema, donde se plasman las características que va a poseer, para culminar con la implantación del prototipo del sistema de instalación remota de software, proporcionando conocimientos que se adquirieron con el trabajo realizado.

Palabras clave: paradigma de agentes, metodologías orientadas a agentes, herramientas para construcción de agentes.

Abstract

This thesis defines the analysis and design of a system based on agents for the remote software installation which seeks to be implanted in the Petroleum Mexican Institute to solve problems like the readiness of personal, the lost time that the personnel of area data processing spend in their transfer from their work place to the user area who requires a service. For these reasons, a prototype has been implemented to prove the operation of the system.

To solve this problem, it has thought to use of the agents technology because it allows to carry out autonomous applications, manipulates its environment, provides flexibility to the systems, as well as, it presents a dynamic behavior in its running and it integrates itself easily with other systems.

To get the analysis and design of the system they had provided the foundations of the paradigm of agents, as it is it the theory, architectures, languages and agents' types. Then, a general panorama of the methodologies is given guided agents that have bigger acceptance to carry out the analysis, design and documentation. Some tools are also revised to create systems multi-agents, since like one has to prove the carried out analysis, it is necessary to use one. Then it is presented the conceptual pattern of the System properly, where the characteristics are captured that will possess, to culminate with installation of the prototype of the system of remote installation of software, providing knowledge that were acquired with the carried out work.

Keywords: agents' paradigm, agent-oriented methodology , tools for agents' construction.

Índice de contenidos.

Lista de figuras.

Entorno del problema

Introducción

El Instituto Mexicano del Petróleo es un “organismo descentralizado de interés público y preponderantemente científico, técnico, educativo y cultural, con personalidad jurídica y patrimonio propios, cuya función será buscar la independencia científica y tecnológica en el área petrolera”, creado en 1965 por decreto del entonces presidente Gustavo Díaz Ordaz para desarrollar investigación y tecnología para la industria petrolera nacional.

Dentro del Instituto Mexicano del Petróleo encontramos una organización estructurada en diferentes programas, dentro de uno de esos programas se encuentran las Direcciones Ejecutivas, donde se localiza la Dirección Ejecutiva de Planeación y Desarrollo Institucional, que en un siguiente nivel se halla la Gerencia de Tecnología Informática, la cuál cuenta con diversas áreas, pero el área en la cual se centrará el presente estudio es de Soporte Informático. El diagrama de la figura 1 muestra la estructura mencionada.

Antecedentes del problema

El campo de estudio de la tesis comprende el área de Soporte Informático de la Gerencia de Tecnología Informática, que brindan sus servicios al área administrativa del Instituto Mexicano del Petróleo.

El servicio que ofrece esta área está encaminado a cuestiones de software, ya que tiene a su resguardo el software que emplea la parte administrativa del Instituto. Por tanto los servicios de instalación de software, configuración de

cuentas de correo electrónico, antivirus institucional, depuración de los equipos, instalación de sistema operativo, solución de problemas con el software, etc., son atendidos por el área en cuestión.

La razón por la cual se realiza el análisis es por la inquietud de los usuarios que sufren retrasos en sus proyectos, debido en algunas ocasiones a la lentitud con que se atienden sus necesidades por parte del área de Soporte Informático que no es oportuna, debido a que en general la cantidad de personal técnico disponible y los tiempos de su traslado al área solicitante son altos, además el personal dedicado a esta labor no destina su tiempo completo a ésta, ya que tiene que realizar otras actividades propias de la Gerencia de Tecnología Informática. El motivo de que existan tareas de tiempo compartido, se debe a que se presentan periodos irregulares en la carga de trabajo en el área de soporte informático y sería un desperdicio de recursos tener al personal inactivo cuando no exista alguna demanda de servicio. Otra problemática que se encuentra inmersa es que cuando al personal de Soporte Informático se le acumula el trabajo de esta área y a su vez tiene que realizar alguna actividad de otra área de la Gerencia de Tecnología Informática que tiene cierta prioridad, es comprensible que va a descuidar o no va a dedicar el tiempo requerido para terminar las actividades en el tiempo que se han planteado en un inicio.

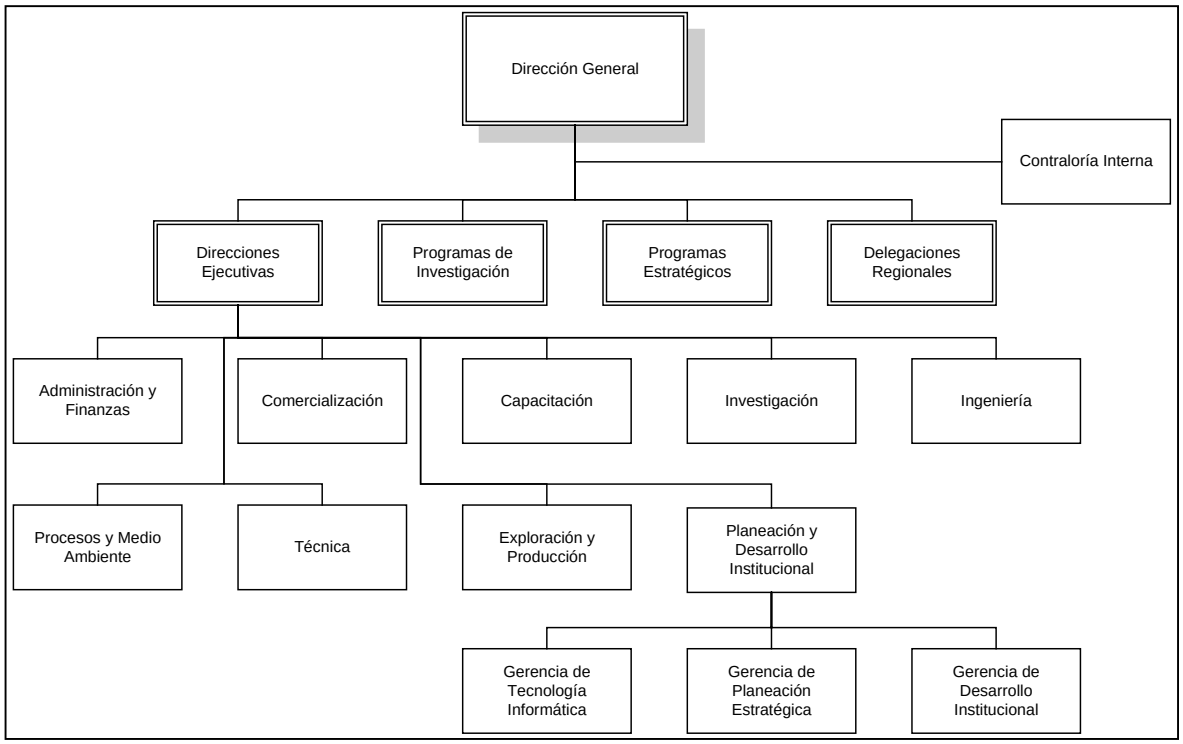

Fig. 1 Estructura Organizacional

Investigación preliminar

En este punto se considera la información de cómo se lleva a cabo el proceso de servicio del área de Soporte Informático actualmente. Para una mayor comprensión del proceso se coloca a continuación en la figura 2 el diagrama de actividades UML que muestra el proceso del negocio. La información contenida en el diagrama se obtuvo por medio de un estudio realizado al personal que realiza las actividades así como por observación directa.

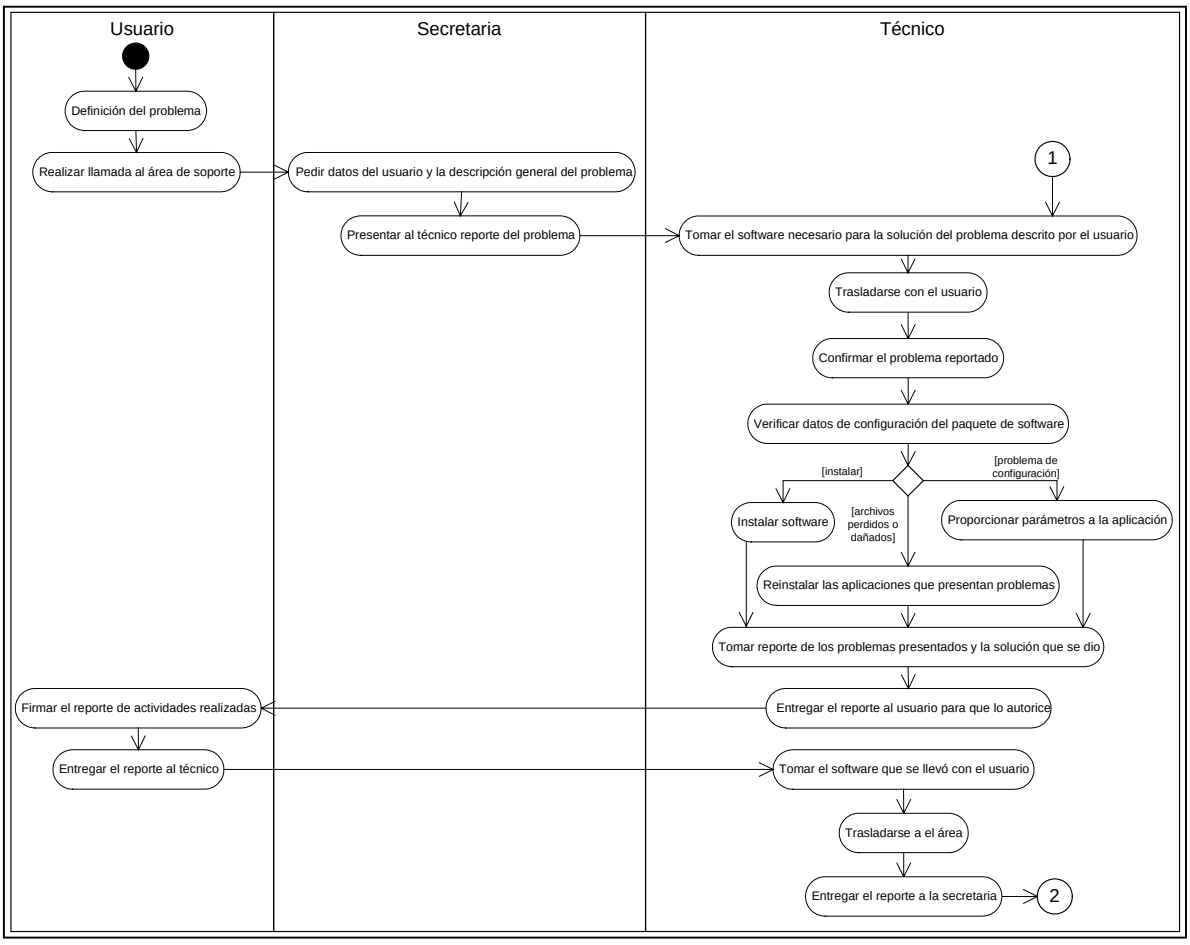

Fig. 2 Diagrama de actividades "Proceso del negocio"

Características de los recursos con que cuenta la organización

Ya descrita la problemática y el procedimiento de cómo se lleva a cabo el servicio que proporciona el área de Soporte Informático, prosigue la descripción de los recursos con que cuenta la Institución.

El Instituto Mexicano del Petróleo cuenta con un inventario próximo a las 5,000 computadoras personales dentro de sus diferentes programas, de las cuales para

la parte administrativa a la que brinda el servicio el área de Soporte Informático cuenta con un aproximado de 900 computadoras. El sistema operativo con que cuentan estas computadoras es Microsoft Windows variando en las diferentes versiones que existen del mismo. Por ejemplo, existen algunos equipos con versiones 95, 98, Me, pero la mayoría de los equipos ahora cuentan con Microsoft Windows XP. Todas las computadoras cuentan con tarjetas de red trabajando con tecnología Ethernet, que se conectan a la Intranet con que cuenta el Instituto para compartir los recursos.

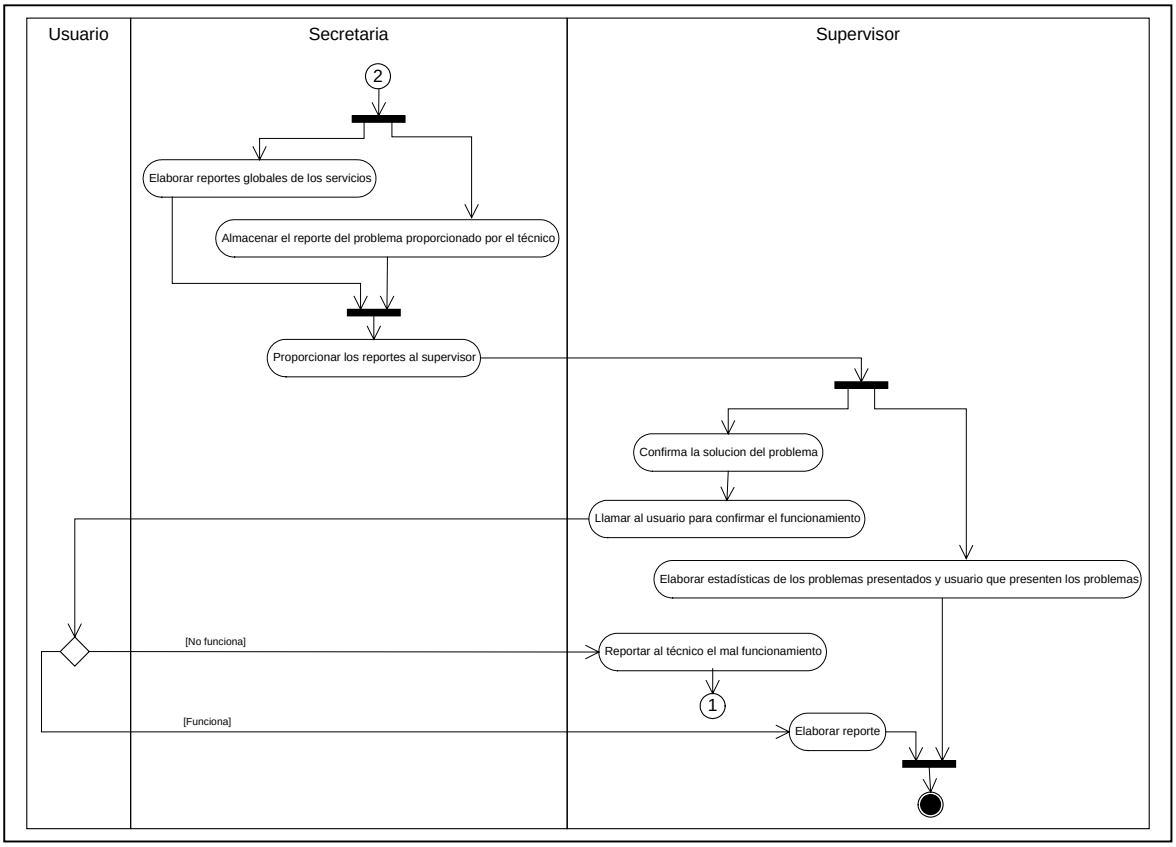

Fig. 2 Diagrama de actividades "Proceso del negocio" (Continuación)

La Intranet sigue las normas del cableado estructurado. La topología física es tipo estrella con 5 switches enlazados en forma redundante. La conexión del Backbone está conformado con fibra óptica, la cual tiene una velocidad de transmisión de 622 Mbps empleando la tecnología ATM. El cableado vertical también está conformado por fibra óptica y tecnología ATM, en tanto que el cableado horizontal lo constituye el cable de par trenzado categoría 5 con tecnología Ethernet y Fast-Ethernet, que puede alcanzar una velocidad de transmisión de 100 Mbps.

Descripción de la problemática encontrada

Con base en los datos que se muestran en el diagrama de actividades del negocio se presentan los puntos en donde se han encontrado problemáticas. Estos son:

- Al iniciar el proceso se necesita que la secretaria esté disponible, ya que es la encargada de tomar los reportes de los servicios que solicitan los usuarios del área administrativa.
- Pasar el reporte de los datos del usuario y del equipo que presenta la problemática al personal encargado de realizar los servicios.
- El personal que presta el servicio de soporte informático tiene que estar disponible en el momento en el cual le ha hecho llegar el reporte.
- Pérdida en el tiempo de traslado del lugar del personal de soporte informático al área del usuario demandante.
- El usuario debe encontrarse en su lugar de trabajo en el momento en el que el personal de soporte informático acuda a resolver el problema de software que presente la máquina.
- En muchas ocasiones los procesos que tiene que realizar el técnico son repetitivos o triviales.
- Los procesos que se tienen que realizar detrás de lo que es el servicio propiamente también son repetitivos.

Visión de la mejora en los procesos

Después de que se ha descrito el proceso de cómo son atendidas las solicitudes de los usuarios y de haber identificado cuales son los puntos en que se tienen algunos problemas al ofrecer los servicios que proporciona el personal de Soporte Informático, a continuación se marcan las situaciones en las cuales se pretende incidir para eliminar las deficiencias detectadas. Así, se pretende:

- Instalar software de forma oportuna.
- Configurar el software eficazmente.
- Personalizar el equipo del usuario oportunamente.
- Solucionar problemas detectados por el usuario en su equipo de cómputo.
- Automatizar los procesos de administración de la información recabada durante el proceso de servicio.
- Eliminar la necesidad de que el responsable se encuentre disponible para realizar el servicio, ya sea que se trate del personal del área de soporte o del usuario.

De acuerdo a los puntos que se pretenden optimizar, indagando en métodos para desarrollar aplicaciones que realicen labores de forma autónoma, se ha pensado en alguna aplicación que cuente con un mecanismo que permita, por las

características concebidas del sistema, que con una serie de parámetros interactúe con su ambiente y si se presenta con alguna otra problemática durante su ejecución, permita darle solución o seguir su trabajo. Las características que se mencionan pueden ser brindadas por un enfoque relativamente nuevo, llamado “agentes”.

Aunque es posible desarrollar el sistema con un enfoque orientado a objetos, no se contaría con la flexibilidad, el comportamiento autónomo y la evolución del comportamiento dinámico que se ha detectado en el enfoque de agentes. Por ejemplo un agente ejecuta un razonamiento autónomo elegido de una serie de alternativas; presenta la capacidad de ser activo, ésto es, que es capaz de actuar dirigido por objetivos que posee; permite cambiar su comportamiento en tiempo de ejecución, además de que es capaz de enfrentarse con situaciones no previstas o inesperadas; permite la asociación entre agentes para enfrentarse a una problemática o la realización de un objetivo.

Se desea que el sistema cuente con alguna de estas características mencionadas anteriormente como la asociación con otros agentes para enfrentarse al problema del dominio de instalar software cosa que en el paradigma orientado a objetos no esta contemplado; debe ser autónomo para resolver las dificultades encontradas en la medida de lo posible, para que el personal de la Gerencia de Tecnologías de Información dedique su tiempo a otras actividades que requieran mayor atención; debe ser capaz de cambiar su comportamiento en tiempo de ejecución debido a que existe una colaboración entre agentes y éstos tendrían que esperar cierta información para continuar con su tarea, cosa que con un enfoque orientado a objetos tampoco se tiene contemplado; debido a que no es posible o práctico especificar el comportamiento del sistema caso por caso se pretende el empleo de agentes ya que los objetivos son los que marcan la pauta de sus tareas y para el caso del enfoque de objetos no es así; otro punto por el cual se ha decidido la implementación con agentes, es debido a que puede extenderse fácilmente con otros sistemas de agentes y la flexibilidad que proporciona es mayor, en cambio en el enfoque de objetos ésto tampoco requeriría un mayor trabajo para adaptar su interacción con otros sistemas.

Las cualidades que han sido mencionadas son las que han motivado en dar solución a la problemática detectada en el área de Soporte Informático empleando el enfoque de agentes.

Para un conocimiento general del contenido de cada uno de los capítulos e introducirse al tema de interés, se muestra un resumen de cada capítulo.

El Capítulo 1 “Fundamentos de Agentes” describe de una forma general el contexto de agentes.

El Capítulo 2 “Estudio de Herramientas para el Desarrollo de Agentes” presenta una serie de metodologías existentes y de herramientas que se encuentran en el mercado para el diseño y desarrollo de sistemas Multi-agentes.

El Capítulo 3 “Modelo conceptual del Sistema” describe el diseño del sistema basado en agentes para la instalación remota de software empleando la metodología MAS-CommonKADS.

El Capítulo 4 “Prototipo del Sistema” refleja parte de la implantación del prototipo Multi-agente.

El Capítulo 5 “Conclusión y Futuros trabajos” se presentan, como dice el título, las conclusiones de la tesis y los trabajos futuros.

Anexo A “Agent UML” introduce al lector a un conocimiento acerca del A-UML, un lenguaje para la representación de agentes, el cual es empleado en la representación de algunas partes del diseño del sistema.

Anexo B “Manual de Instalación del prototipo” se presenta de forma más detallada cómo se realiza la instalación del prototipo y los requerimientos que se necesitan para su funcionamiento.

CAPÍTULO 1
Fundamentos de agentes

1.1 Introducción

En este capítulo se presenta una visión general sobre el paradigma de agentes. En primer lugar se proporciona una breve reseña histórica del desarrollo de agentes (sección 1.2). En el siguiente apartado se muestran las perspectivas sobre el estudio de los agentes (sección 1.3), desplegando información de las áreas de investigación que han surgido referente al campo. La última parte del capítulo enumera una serie de entornos donde son útiles los agentes (sección 1.4).

1.2 Historia

Los agentes de software forman parte de los sistemas multi-agentes (Multi-Agent Systems MAS), los cuales a su vez pertenecen al campo de la Inteligencia Artificial Distribuida (DAI) uno de los tres grandes campos de la Inteligencia Artificial. Los otros dos campos son solución de problemas distribuida (Distributed Problem Solving, DPS) e inteligencia artificial paralela (Parallel AI, PAI) (Fig.1.1). Por lo tanto, los agentes de software, heredan muchas de las motivaciones, objetivos y beneficios potenciales de la Inteligencia Artificial Distribuida como: la modularidad, la velocidad propia del paralelismo, y la precisión debida a la redundancia. También heredan de la inteligencia artificial la operación al nivel del conocimiento y la facilidad de mantenimiento (Huhns & Singh, 1994).

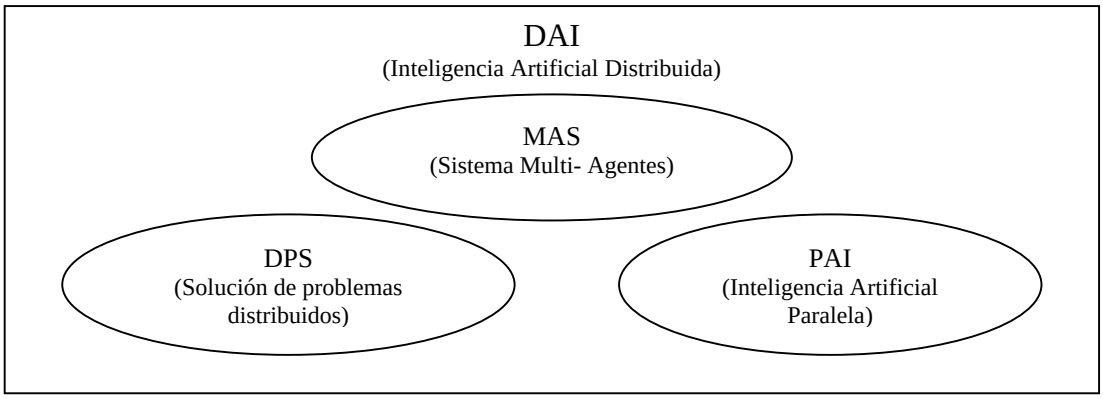

Fig. 1.1 Campos de la Inteligencia Artificial Distribuida

El concepto de agente se remonta a los primeros días de investigación en Inteligencia Artificial, en la época de los 70's. Carl Hewittis en 1977 propuso el concepto de un objeto auto-contenido, interactivo y de ejecución concurrente llamado actor. Este objeto tenía algún estado interno encapsulado y podría responder a mensajes de otros objetos similares.

Podemos dividir la investigación de agentes en dos ramas principales, la primera en el periodo comprendido entre 1977 y 1990 y la segunda desde 1990 hasta nuestros días. En el primer periodo las investigaciones se concentraron principalmente en agentes del tipo repartidores, con modelos simbólicos internos, también en aspectos de discusión macros como teoría de arquitectura, teoría de lenguaje, la interacción y la comunicación entre agentes, la descomposición y la distribución de tareas, coordinación y cooperación, resolución de conflictos vía negociación, etc., su objetivo fue especificar, el análisis y el diseño de sistemas integrados incluyendo múltiples agentes colaborativos. Estos puntos de agentes, enfatizaban en las sociedades de agentes más que en agentes individuales.

Sin embargo, a partir de los años noventa, adicionalmente a la investigación en aspectos macros, se hace evidente e inconfundible, que el rango de los tipos de agentes siendo investigados y desarrollados es ahora mucho más amplio e incluyen una preocupación por la integración de bases de datos y sistemas de inferencia. La utilización de los agentes se está dando en situaciones con tipos de comunicaciones complejas y diversas; sistemas en los que no es práctico o posible especificar el comportamiento caso por caso (cuando los objetivos marcan la pauta de las tareas a realizar); sistemas de negociación, cooperación y competencia entre entidades; en la creación de sistemas autónomos. En el caso particular de los MAS son muy modulares permitiendo la distribución, extensibilidad y flexibilidad [1].

1.3 Perspectivas en el estudio de agentes

Dentro del área de agentes han emergido principalmente cuatro importantes campos de investigación [2]:

- La **teoría de agentes** determina qué es un agente, propiedades que deben de tener los agentes, y la utilización de formalismos matemáticos para representar y razonar con ellos. En la actualidad hay investigadores que han seguido paradigmas diferentes a la lógica, para enriquecer el comportamiento de los agentes, en el sentido de incorporar, por ejemplo, representaciones de tiempo y acción. También como señalan Wooldridge y Jennings, explicar cómo se relacionan aspectos de obligaciones, deseos, intenciones, etc. con la información que posee el agente.
- La **arquitectura de agentes** tiene que ver con aquellos aspectos que conducen a la construcción de sistemas de computación cómo son tanto las arquitecturas de software cómo de hardware que permiten reflejar las propiedades enunciadas por los teóricos.
- Los **lenguajes de agentes** son los sistemas de software que permiten programar agentes con los términos que han sido desarrollados en el campo de la teoría de agentes.
- La **tipología de agentes** estudia a los agentes desde la perspectiva de su campo de aplicación, más que su arquitectura.

1.3.1 Teoría de agentes

Las teorías de agentes son especificaciones para conceptualizar los agentes. Aunque no hay una definición universalmente aceptada del término de agente hay coincidencia en aceptar la autonomía como aspecto clave y el resto de cualidades, depende del ámbito de definición.

La definición tomada de D.R.A.E [3] señala: " *persona o cosa que produce un efecto." "Persona que obra con poder de otra"*

Para Wooldridge [4] un agente *"es una entidad independiente y completa en si misma, implementada en hardware, software, o una mezcla de ambas, con propiedades como autonomía, sensibilidad al entorno, iniciativa, adaptabilidad, movilidad y/o racionalidad."*

"Para Shoham [5] un agente es una entidad que tiene un estado interno formado de componentes mentales, tales como creencias, capacidades, elecciones y compromisos"

Mientras que Maes [6], dice que un agente es un sistema comprometido en alcanzar metas en un medio ambiente complejo y cambiante.

Por su parte, Wooldridge y Jennings [7] se refieren a un agente como aquel sistema de cómputo que tiene las propiedades de ser autónomo, auto-contenido, reactivo, pro-activo, típicamente controlado de manera central, y que es capaz de comunicarse con otros agentes a través de un lenguaje.

Según la definición progresiva de Russell y Norving [8] un agente *"es algo que demuestra que percibe su entorno a través de sensores y actúa sobre dicho entorno mediante efectores"*

La definición que da FIPA *"Un Agente es el actor fundamental en una plataforma de agentes que combina una o más capacidades de servicio en un modelo de ejecución unificado e íntegro que puede incluir acceso al software externo, usuarios humanos y medios de comunicación."* [9]

La dificultad que presenta definir un agente ya sea porque los investigadores, no son dueños del término agente o debido a que la comunidad de software lo ha usado en campos heterogéneos, no impide definir las características que forman el concepto de agente, mismas que pueden observarse en la tabla 1.1.

Partiendo de la idea de que cualquier programa pueda ser denominado como agente, se pueden distinguir dos nociones extremas de agentes [10]:

Una noción débil de agente consiste en definir un agente como una entidad que es capaz de intercambiar mensajes utilizando un lenguaje de comunicación de agentes, lo que implica autonomía, reactividad, pro-actividad y sociabilidad.

Una noción más fuerte o restrictiva es la enunciada por Shoham en su propuesta de programación orientada a agentes (AOP), donde un agente contiene nociones mentales además de racionalidad, veracidad y adaptabilidad o aprendizaje, incrustando las características presentadas en la noción débil de un agente.

En base a las referencias mencionadas de agentes la definición que se propone de un agente es:

Una entidad independiente que trabaja en función a un servicio solicitado por otra entidad o en función de sus creencias, manipulando herramientas propias o herramientas de otros, de acuerdo a sus conocimientos o a su forma de racionalizar los problemas encontrados, además es capaz de tener una comunicación a través de un lenguaje.

Se puede observar que de entre las diferentes propiedades que posee un agente quizá la más importante sea la de autonomía. Esta propiedad es requisito para que se dé otra propiedad igualmente importante, la adaptación, que tiene que ver con el comportamiento inteligente del agente.

Características	Descripción
Comunicación	La habilidad de intercambiar mensajes.
Interoperación	La habilidad de intercambiar información y servicios con otros agentes programas y por lo tanto resolver problemas.
Movilidad	La habilidad de cambiar de posición física en una red.
Autonomía	La habilidad de operar sin intervención directa de humanos o de otras cosas, teniendo control sobre acciones y estados internos.
Adaptación	La habilidad para ajustarse al medio ambiente, o a otras entidades para mejorar su actuación a futuro.
Situado	La habilidad de interactuar continúamente con el medio ambiente.
Pro-activo	La habilidad de exhibir comportamiento dirigido a metas, es decir, tomar iniciativas.
Eficiencia (exitoso**)**	La habilidad de alcanzar metas.
Reactivo	La habilidad del agente de responder rápidamente a los eventos (perturbaciones) que suceden en el ambiente.
Predicción	La habilidad de predecir cómo realizar tareas cuando el agente (deliberativo) tiene un modelo bastante exacto de cómo es su mundo y cómo trabaja.
Interpretación	La habilidad de interpretar correctamente los datos (sensoriales) que se reciben del medio ambiente.
Racional	El agente es racional (deliberativo) cuando puede elegir ciertos comandos que supone o predice le permitirán alcanzar sus metas.
Metas	Denotan las opciones que el agente actualmente tiene. Son un subconjunto de sus creencias.
Introspección	La habilidad de examinar y ser autoreflexivo de las propias ideas y planes.
Intención	Plan parcial de acciones que el agente se compromete a ejecutar para alcanzar sus metas.
Decisión	La obligación que tiene un agente consigo mismo.
Toma de decisiones	El agente selecciona entre opciones alternativas.
Creencias	Hechos que el agente tiene del estado actual del mundo y de la posibilidad de un curso de acción logrando ciertos efectos.
Deseos	Noción abstracta que especifica preferencias sobre estados futuros del mundo o cursos de acción.
Planeación	Proceso de implantación de intenciones.
Obligación	Compromiso que adquiere un agente con otro agente para realizar algún hecho más que tomar una acción.
Veracidad	Se asume que un agente no comunica información falsa a propósito.
Benevolencia	Se asume que un agente está dispuesto a ayudar a otros agentes si esto no entra en conflicto con sus propios objetivos.

Tabla 1.1 Características que puede tener un agente

1.3.2 Arquitectura de agentes

En el diseño arquitectónico de una aplicación utilizando el paradigma de agente se plantean en la literatura dos enfoques: el diseño de agentes monolíticos, en el que la aplicación o servicio a ofrecer lo dá un sólo agente autónomo y el modelo multi-agente, en el que existen un conjunto de agentes en una determinada estructura que conforman el servicio ofrecido, pudiendo hacer cada uno una parte del trabajo o haciendo todos el mismo pero desde distintas perspectivas.

Antes de seguir con la clasificación de arquitecturas se presenta una referencia del significado del término arquitectura de agentes.

"Una metodología particular para construir agentes. Especifica cómo... el agente puede ser descompuesto en un conjunto de módulos componentes y cómo estos módulos pueden interactuar. El conjunto total de módulos y sus interacciones deben proveer una respuesta a la pregunta de cómo el dato monitoreado y el estado interno del agente determinan las acciones... y estados internos futuros." [11]

Como se plantea una arquitectura de agentes comprende:

- Una división del sistema en módulos.
- La relación de los distintos módulos entre sí.

Una clasificación de arquitectura de agentes comúnmente mencionada es la basada en el monitor de acción del agente (procesamiento empleado) que se describe en esta tesis.

1.3.2.1 Arquitecturas para agentes Deliberativos

Este tipo de arquitecturas sigue la corriente de la IA simbólica, que se basa en la hipótesis de los sistemas de símbolos-físicos en la cual un sistema de símbolos físicos capaz de manipular estructuras simbólicas puede exhibir una conducta inteligente, es decir, utiliza el modelo de símbolos para mantener una representación explícita e interna del mundo empleando estados mentales. La problemática que presenta es la de cómo describir los objetivos y medios de satisfacerlos, y cómo realizar la traducción del nivel de conocimiento al nivel simbólico.

Las arquitecturas para agentes deliberativos (Fig. 1.2) suelen basarse en la teoría clásica de planificación de inteligencia artificial:

Dado un estado inicial, un conjunto de planes y un estado objetivo, la deliberación del agente en determinar qué pasos debe encadenar para lograr su objetivo, es llevada a cabo por medio de un módulo central (administrador) que

interactúa con los demás módulos que contienen los conocimientos, objetivos, deseos, intenciones, razonando para planificar las acciones a ejecutar.

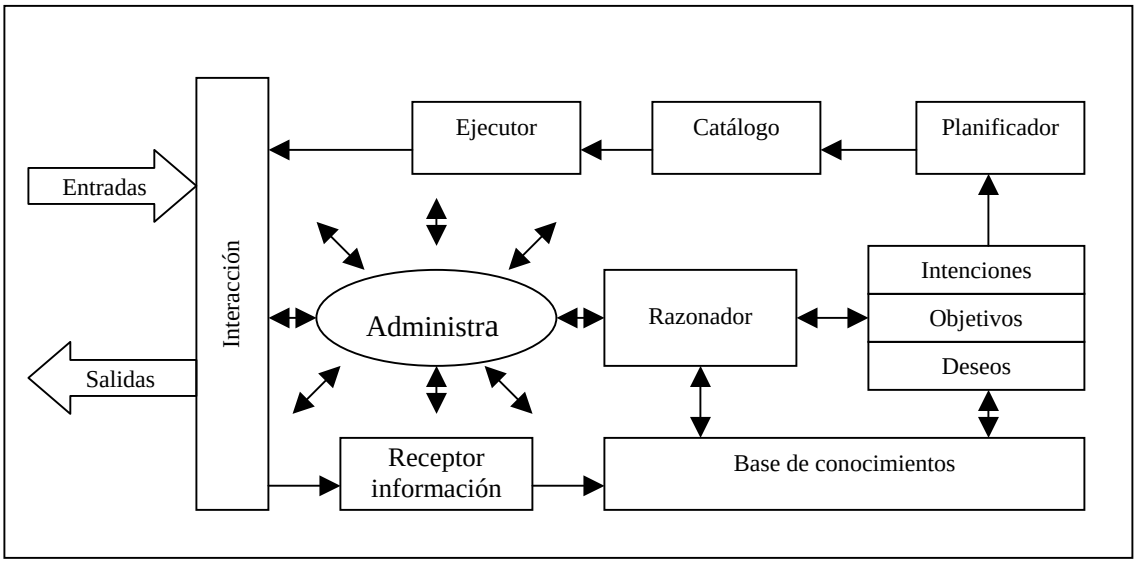

Fig. 1.2 Arquitectura para agentes deliberativos

Este tipo de arquitecturas son utilizadas para representar aspectos significativos de un problema en un dominio determinado, para seleccionar una solución de entre diferentes alternativas, por ejemplo los sistemas expertos basados en conocimientos, con representación y razonamiento simbólico pueden ser ocupados en esta arquitectura.

Este tipo de arquitecturas tiene algunos inconvenientes como lo son la estructura rígida, poca capacidad de reacción inmediata, prefiere la demostración matemática de un plan a su eficiencia

Las siguientes son algunas de las arquitecturas deliberativas desarrolladas:

IRMA (Intelligence Resource-bounded Machine Architecture)

IRMA de Bratman, Israel y Pollack (Fig. 1.3), donde la capacidad de razonamiento está limitada por los recursos disponibles, tiempo de procesador principalmente. Estos recursos van desde el número de procesos o hebras requeridos hasta el número de conexiones necesarias con bases de datos. Las aplicaciones pueden modelarse de distintas formas, como son: la abstracción con objetos y agentes [10].

Esta arquitectura tiene cuatro estructuras claves de datos simbólicos: una librería de planes, una representación explícita de creencias, deseos y deliberaciones. Adicionalmente, la arquitectura tiene: un razonador, para

razonamiento acerca del mundo; un analizador de oportunidades, para determinar qué planes deben ser utilizados para lograr las intenciones del agente, que son filtradas por un módulo para que sean pasadas al módulo de deliberación para ejercer las acciones.

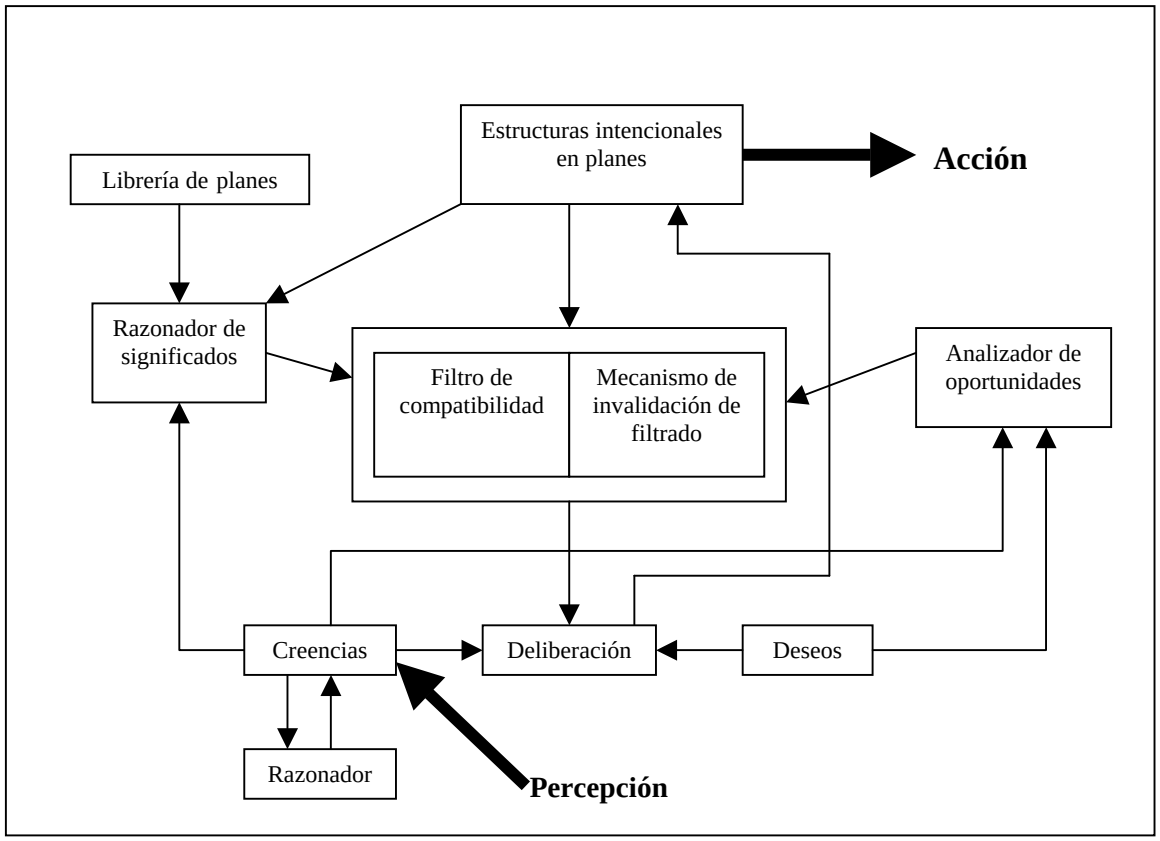

Fig. 1.3 Modelo de la arquitectura IRMA

HOMER

Agente prototipo autónomo con habilidades lingüísticas, capacidades de planeación y acción. A tal agente se le llamó HOMER. La arquitectura de HOMER es específica para el problema de la comunicación en lenguaje natural con el usuario. HOMER toma instrucciones del usuario que pueden contener referencias temporales moderadamente sofisticadas; puede planear cómo realizar sus instrucciones, y puede luego ejecutar sus planes, modificándolos como sea requerido durante la ejecución. El agente además, gracias a su memoria episódica, es capaz de responder a preguntas acerca de sus experiencias pasadas [12].

La arquitectura en la cual HOMER existe es una arquitectura modular (Fig. 1.4). Consiste en una memoria, que suministra información al módulo planificador, que interactúa con procesos reflexivos para generar, de acuerdo a

los planes, una acción o bien información al humano. Esta arquitectura toma en cuenta el entorno que envuelve al agente y la interacción con el usuario de forma separada.

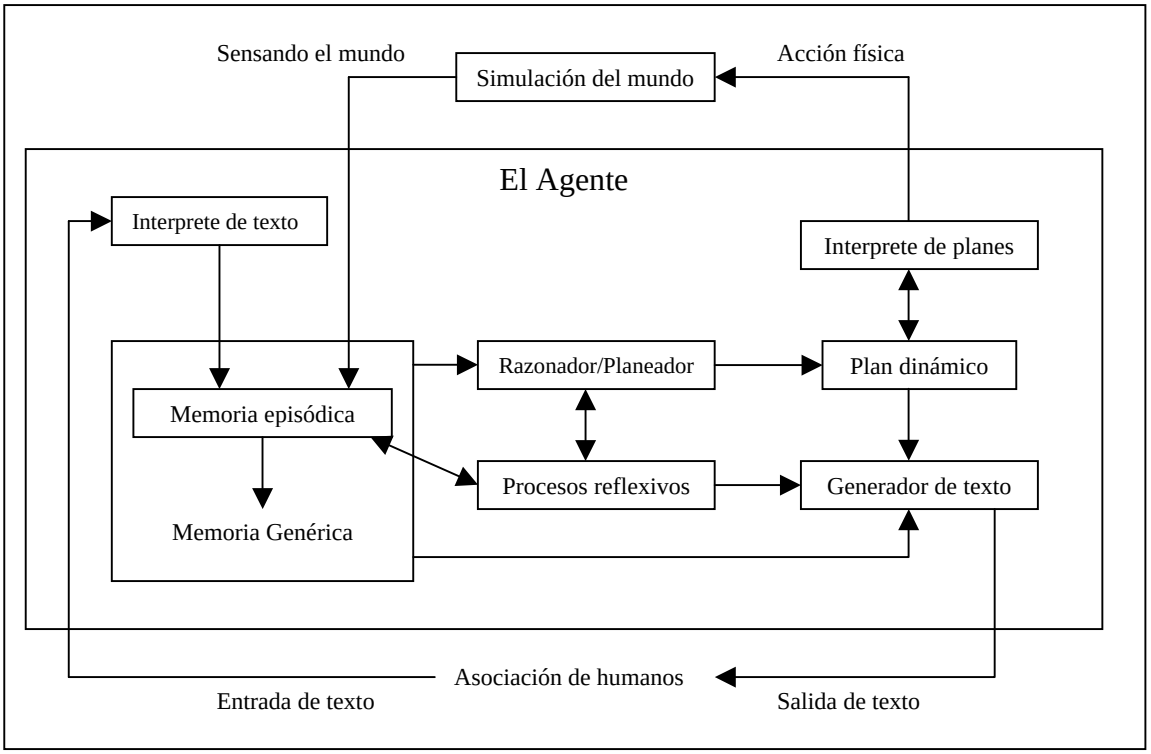

Fig. 1.4 Arquitectura HOMER

GRATE

GRATE es una arquitectura en capas (Fig. 1.5) en la que el comportamiento de un agente es guiado por actitudes mentales tales como creencias, deseos, intenciones individuales e intenciones colectivas [10]. Los agentes se dividen en dos partes distintas: un sistema de nivel de dominio y una capa de cooperación y control. El primero resuelve problemas para la organización, sea ésta en el dominio de controles industriales, de finanzas o transporte. El último es un controlador que opera en el nivel de dominio del sistema con la intención de asegurar que las actividades del agente a nivel del dominio sean coordinadas con aquellas otras dentro de la comunidad.

La capa de cooperación está compuesta de tres módulos genéricos: un módulo de control que es la interfaz con la capa de dominio del sistema, un módulo de fijación de situaciones y un módulo de cooperación que interactúan directa o indirectamente con los módulos de conocimiento, del módulo que abastece la información y del modelo del mismo agente, además posee un módulo que administra las comunicaciones.

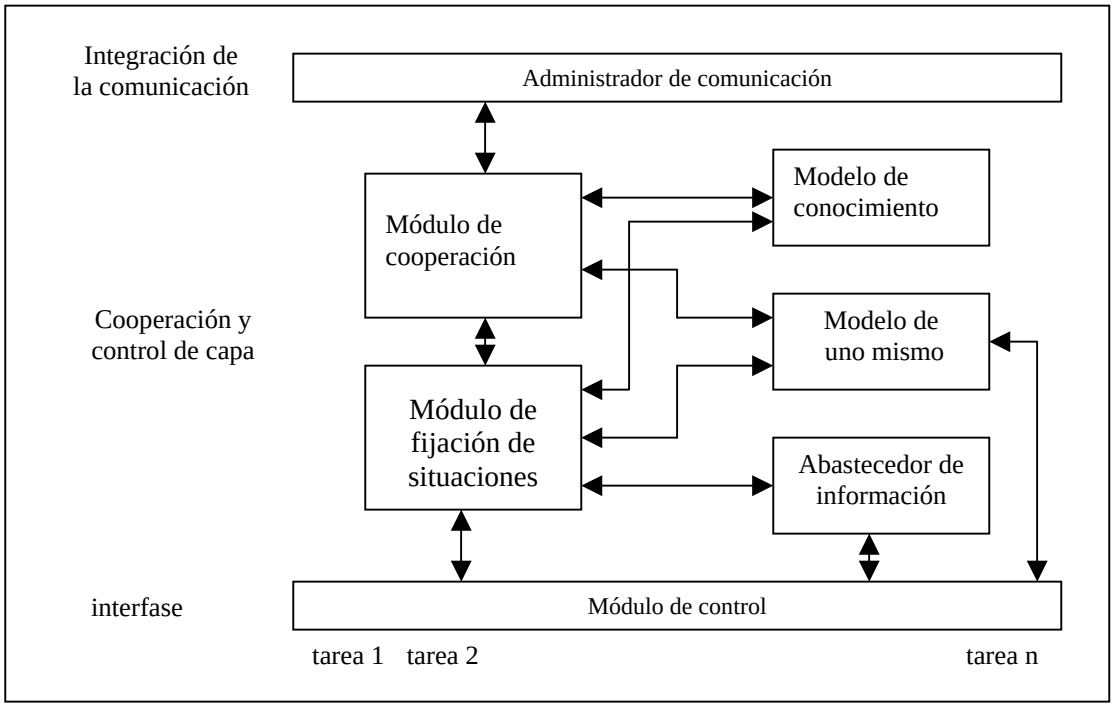

Fig. 1.5 Arquitectura GRATE

1.3.2.2 Arquitecturas para agentes Reactivos

Las arquitecturas para agentes reactivos cuestionan la viabilidad del paradigma simbólico y proponen una arquitectura que actúa siguiendo un enfoque conductista, con un modelo estímulo-respuesta. Estas arquitecturas siguen una serie de patrones que se activan bajo ciertas condiciones de los sensores y tienen un efecto directo en los actuadores, por lo cual pueden ser considerados como agentes no-inteligentes, pues no realizan ningún proceso de razonamiento abstracto ni de planeación, sin embargo, varios investigadores consideran que sus acciones reactivas son lo suficientemente buenas y rápidas para considerarlos como inteligentes.

Esta clase de agentes se caracterizan porque toman sus decisiones en tiempo de corrida, con información limitada y reglas simples de situación-acción, no planifican a futuro y tienen poco que recordar pues están situados en un espacio. Operan según las condiciones que se presentan en el medio ambiente, es decir, si el agente capta a través de sus sensores un cierto estímulo (datos), entonces genera rápidamente una respuesta (comportamiento) para adaptarse al nuevo estado del ambiente. La interacción que se da entre los agentes hace emerger patrones de comportamiento complejo que sólo se aprecia cuando se ve trabajar a todos los agentes en conjunto.

Los agentes reactivos están construidos por una serie de módulos que operan de manera autónoma y diseñados para llevar a cabo una tarea específica, no tienen un módulo central que controle a los demás y a la comunicación. Como se muestra en la figura 1.6 dependiendo de lo que perciben los sensores un módulo interactuará y si necesita de la colaboración de otro módulo existirá una colaboración para producir una acción.

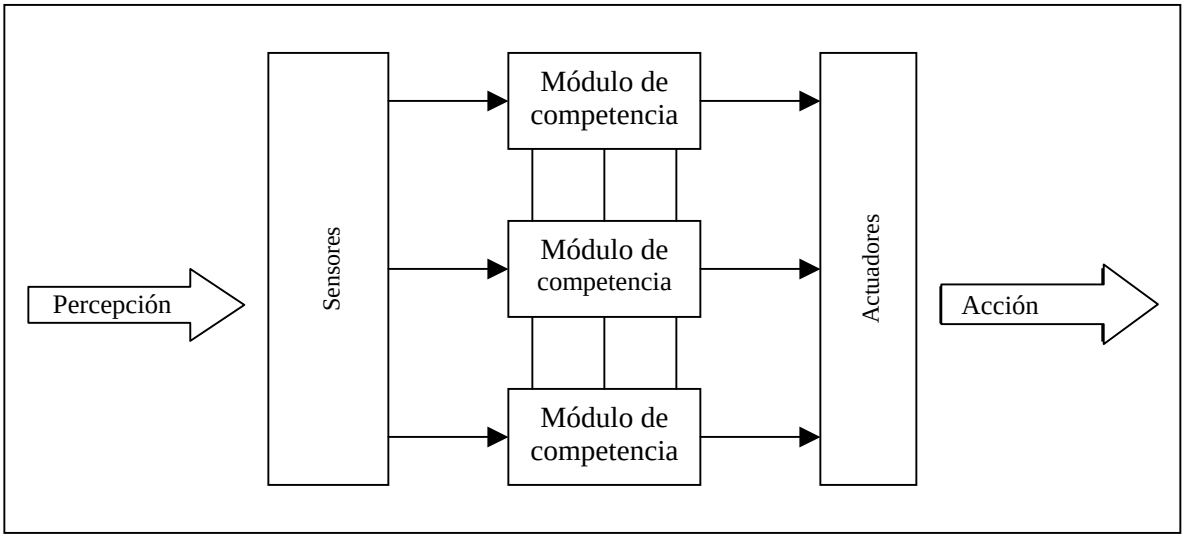

Fig. 1.6 Arquitectura para agentes Reactivos

Esta arquitectura fue pensada para la construcción de robots autónomos, los cuales pueden adaptarse a cambios en su entorno y moverse en él. La idea de tener esos agentes es poder analizar comportamientos sin representación simbólica, de la interacción con el mundo real.

SUBSUMPTION

La arquitectura de Subsumption (Fig.1.7) se construye en capas. Cada capa da un sistema de comportamientos prealambrados . La estructura jerárquica da niveles más altos sobre otros más bajos para crear comportamientos más complejos. El comportamiento del sistema en su totalidad es el resultado de muchos comportamientos simples que obran recíprocamente. Las capas funcionan asíncronamente [5].

Esta arquitectura tiene la particularidad de sensar su entorno que pasará a otros módulos esa información obtenida como es el caso del explorador, el de desvío o el de evitar obstáculo, que interactúan de manera directa para ejercer una acción, pasándose información para manejar mejor las situaciones encontradas.

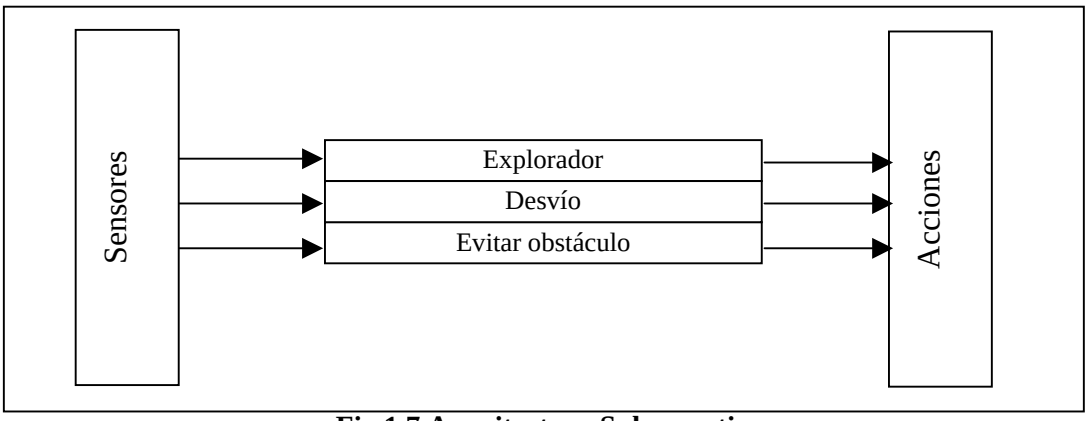

Fig 1.7 Arquitectura Subsumption

PENGI

Agre propuso una arquitectura de agente eficiente basada en la idea de 'running arguments', la idea es que, como la mayoría de las decisiones son rutinarias, pueden ser codificadas a través de una estructura de bajo nivel (tal como un circuito digital), que sólo necesita actualización periódica, para manejar nuevos tipos de problemas. Su enfoque fue ilustrado con el sistema PENGI. PENGI es un juego de computadora simulado, con un sistema central controlado.

El sistema comprende dos subsistemas:

- Uno central que reconoce y actúa sobre aspectos importantes de una situación.
- Otro periférico que consiste en un sistema visual que reconoce situaciones importantes y, de un sistema motor que sirve como interface con los efectores del agente [10].

AUTÓMATAS SITUADOS

En esta arquitectura propuesta por S. Rosensechein y L. Kaelbling, el agente se especifica en lógica modal de conocimiento (términos declarativos), utilizando semánticas de mundos posibles para interpretar mundos en términos de los estados del autómata. Un agente se especifica en términos de dos componentes: percepción y acción. Dos programas se utilizan para sintetizar a los agentes: RULER se emplea para especificar el componente referente a la percepción de un agente; mientras que GAPPS para el componente referente a la acción del agente [10].

1.3.2.3 Arquitecturas para agentes Híbridos

Las arquitecturas para agentes híbridos surgen de la combinación de aspectos de las arquitecturas anteriores, la idea es tomar lo mejor de la parte reactiva y deliberativa y evitar lo más posible las limitaciones y problemas que cada una presenta. Las limitaciones se han superado utilizando un enfoque de arquitecturas por capas (Fig. 1.8) [13], que permite estructurar funcionalidad y control, apoyando el diseño de capacidades como: reactividad, deliberación, cooperación y adaptación. Lo que se intenta es que los módulos reactivos se encarguen de procesar los estímulos que no necesitan deliberación, mientras que los módulos deliberativos determinen que acciones deben realizarse para satisfacer los objetivos locales y cooperativos de los agentes.

Las ventajas que pueden ofrecer estas plataformas son:

- Permiten modular un agente.
- Hacen el diseño de agentes más compacto y robusto.
- Hacen que las capacidades del agente se incrementen ,así como su reactividad.
- La cantidad de conocimiento que requiere cada capa se puede limitar para que el agente tome decisiones.

Este tipo de arquitecturas son utilizadas cuando por ejemplo se desea que un agente sea capaz de una abstracción para permitir respuestas muy rápidas en ciertas tareas y en otras que requieran encontrar una solución entre algunas alternativas donde el objetivo del agente marca las tareas a realizar.

INTERRAP

La arquitectura en capas InteRRaP, puede ser usada tanto para la construcción de un agente como para la de un robot autónomo. La arquitectura consta de un agente basado en conocimientos con su correspondiente unidad de control, que además controla las comunicaciones de bajo nivel. Existen tres niveles o capas de control en esta arquitectura (Fig.1.8): El BBL (nivel basado en comportamiento); el nivel de planeación local (LPL) y el nivel de planeación cooperativa (CPL). La parte reactiva que permite eficiencia, reactividad y robustez está implementada en BBL, y comprende una importante cantidad de patrones de comportamiento (PoBs) basados en reglas de situación-acción. Estas permiten al agente reconocer rápidamente situaciones de tiempo crítico y reaccionar adecuadamente. El nivel intermedio LPL implementa conductas dirigidas por objetivos, mientras que la superior, CPL, habilita al agente a planear/cooperar con otros agentes en pos de alcanzar planes multi-agentes, así como a resolver conflictos. Todos estos niveles trabajan con diferentes modelos en la base de

conocimientos del agente: BBL opera con el modelo "mundo", LPL lo hace con el modelo mental y CPL con el modelo social [13].

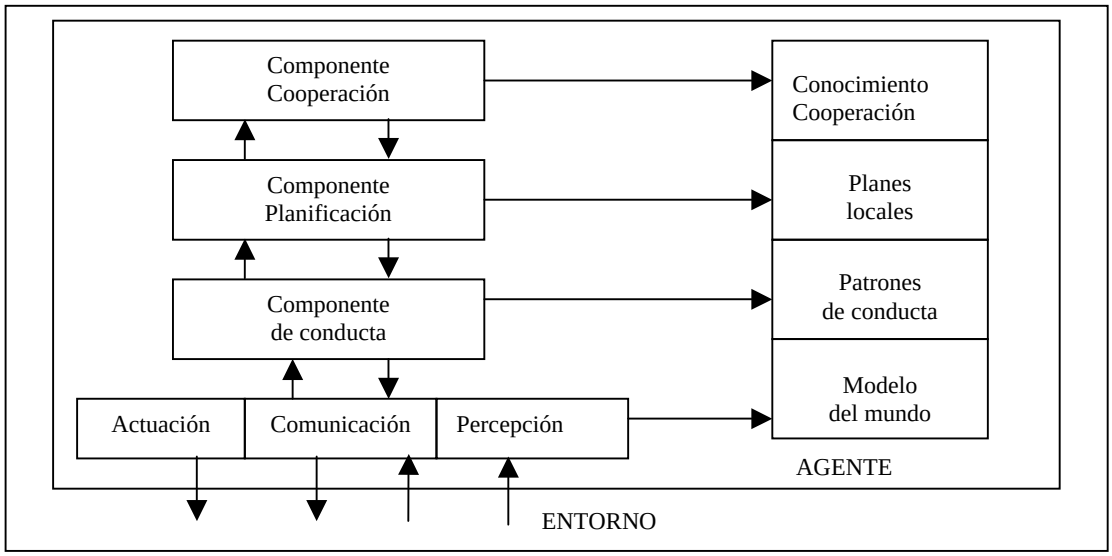

Fig.1.8 Arquitectura para agentes Híbridos InterRAP

MÁQUINA DE TURING

La arquitectura de la máquina de Turing propuesta por Ferguson [14] es también un buen ejemplo de un agente híbrido dinámico, racional y móvil. Esta arquitectura es similar a la "subsumption architecture" vista antes y consistente en tres capas de control: la reactiva, la de planeación y la de modelos (Fig.1.9). Todas ellas trabajan concurrentemente. La diferencia principal entre las arquitecturas de Turing y la de Subsumption con respecto a la InterRaP es que las primeras disponen las capas horizontalmente y la otra lo hace en forma vertical. En las primeras, todos los niveles tienen acceso a los datos que proceden de los sensores de datos y todos los niveles pueden contribuir a la acción. En InterRaP sólo el nivel inferior recibe y actúa sobre los datos perceptuales (Fig.1.8).

La arquitectura de la máquina de Turing cuenta con un subsistema que percibe el entorno y provee de información a las capas de modelado, planificación, reactiva y al subsistema de control, controlado por un pulso para ejercer una acción.

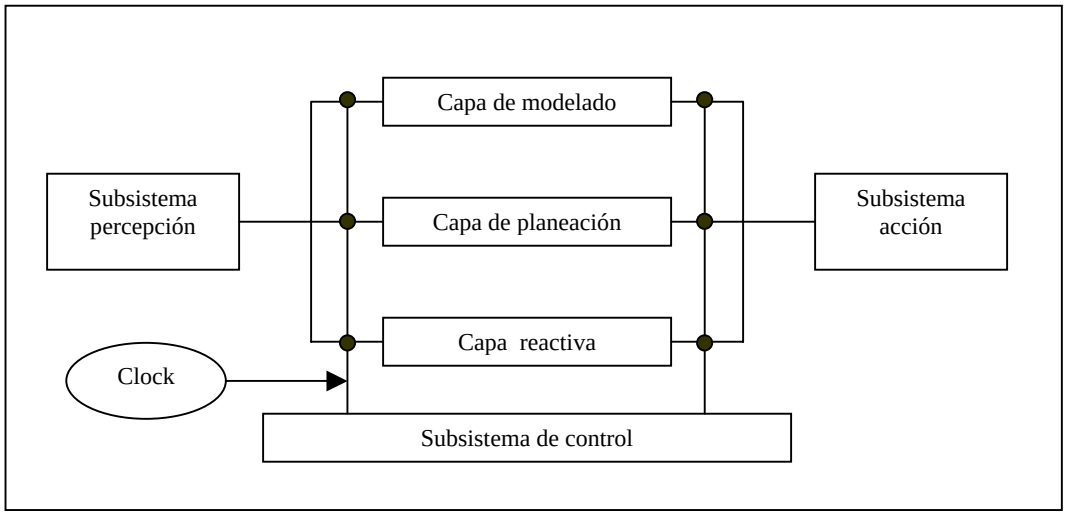

Fig. 1.9 Arquitectura de la Maquina de Turing

PRS (*Procedural Reasoning System*)

Es una arquitectura basada en creencias, deseos e intenciones, que incluye una librería de planes, así como una explícita representación simbólica de las creencias, deseos e intenciones. Las creencias son expresadas mediante la lógica clásica de primer orden. Los deseos son representados como comportamientos del sistema. Una librería de planes de un PRS contiene un conjunto de planes parcialmente elaborados, llamadas áreas de conocimiento (*knowledge areas Kas)* (Fig. 1.10),cada una de las cuales se asocia con una condición de invocación. Estas condiciones determinan cuando la KA debe ser activada. Las KAs pueden ser activadas de una manera dirigida por objetivos o dirigida por datos; las KAs pueden ser también reactivas, permitiendo que el PRS responda rápidamente a cambios en su ambiente. El conjunto de KAs actualmente activas en un sistema representan sus intenciones. Estas estructuras de datos son manipuladas por un sistema intérprete, que es responsable de la actualización de las creencias, invocando KAs, y ejecutando acciones [5].

Se cuenta con un módulo monitor que por medio de sensores capta los cambios en el ambiente. Los cambios son pasados al razonador que va a estructurar de acuerdo a las creencias, planes y deseos, interacciones para que un módulo generador de comandos realice acciones concretas.

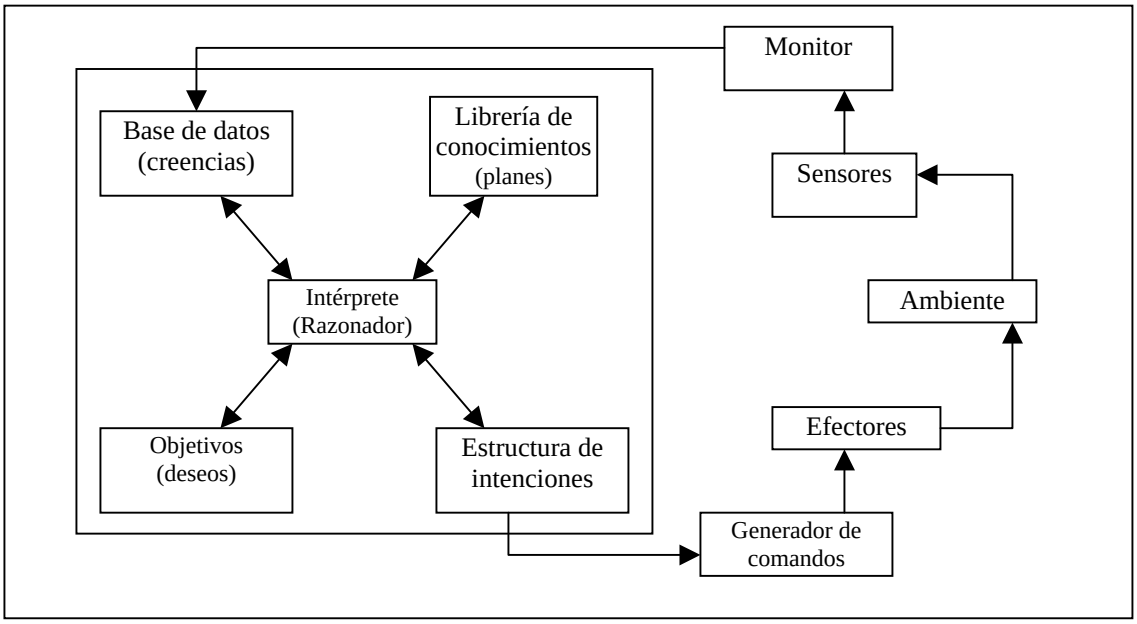

Fig. 1.10 Arquitectura PRS

1.3.3 Lenguajes de agentes

Los lenguajes de agentes son lenguajes que permiten programar agentes con los términos desarrollados por los teóricos de agentes.

Dentro de los lenguajes de agentes podemos hacer la distinción de:

Lenguajes de agentes de propósito general: lenguajes destinados a programar agentes genéricos utilizables en cualquier aplicación.

Lenguajes de agentes específicos: lenguajes para un tipo de agentes específicos, por ejemplo lenguajes para agentes móviles Telescript o Agent-Tcl.

En el momento de programar un agente existen niveles en la etapa de programación que deben de tenerse en cuenta, estos son:

Lenguajes de programación de la estructura del agente: permiten programar las funcionalidades básicas para definir a un agente: funciones de creación de procesos (creación del proceso agente y de los procesos concurrentes al agente) y funciones de comunicación entre agentes (nivel de transporte). Este nivel de programación normalmente sólo es utilizado por los desarrolladores de una plataforma de desarrollo de agentes, los lenguajes empleados suelen ser C, C++, Java, Lisp, Prolog, Prolog/C CUBL o CLOS/C++ entre otros.

Lenguajes de comunicación de agentes: definición del formato de los mensajes intercambiados, de las primitivas de comunicación y de los protocolos disponibles. De este tipo de lenguajes podemos distinguir:

- Procedimentales.- Se basan en el intercambio de directivas procedimentales, es decir, un agente recibe un mensaje que implica la ejecución de un procedimiento. Suelen emplear lenguaje de intérpretes de órdenes (scripts) como Perl, Tcl, etc..

- Declarativos.- se basan en el intercambio de actos comunicativos, es decir, un agente recibe un mensaje con un acto comunicativo que permite interpretar el contenido del mensaje. Entre los ejemplos más utilizados tenemos KQML, KIF y ACL-FIPA.

Lenguajes de programación del comportamiento del agente: permiten definir el conocimiento del agente: conocimiento inicial (modelo de entorno, creencias, deseos, objetivos), funciones de mantenimiento de dicho conocimiento (reglas, planes, etc.), funciones para alcanzar sus objetivos (planes, reglas, etc.) y funciones para desarrollar habilidades (programación de servicios).

Podemos distinguir diferentes tipos de lenguajes de descripción de agentes:

- *Lenguajes de descripción de agente:* los agentes se derivan de una clase de agente genérica, permitiendo la definición de los elementos básicos del modelo de agente tales como base de conocimiento, grupos de agentes, habilidades del agente, servicios ofrecidos, planes para alcanzar objetivos, etc.. La descripción se traduce a un lenguaje ya ejecutable. Ej. *MACE ADL , AgentSpeak* y *MAST*/*ADL*.

- *Lenguajes de programación orientados a agentes:* siguen el paradigma de Shoham de la programación orientada a agentes (AOP), se definen algunos lenguajes que toman en cuenta el estado mental del agente para programar las funciones de transición entre estados mentales, que consisten en creencias, capacidades y obligaciones. Ej. *AGENT0*, *PLACA* y *Agent-K*.

- *Lenguajes basados en reglas de producción:* la programación de la base de conocimiento de los agentes se realiza en algunos sistemas empleando reglas de producción. Ej. MAGSY (basado en *OPS5*), *DYNACLIPS* (basado en *CLIPS*) y *RTA*, que permite definir las conductas del agente con reglas.

- *Lenguajes de especificación:* emplean una especificación (normalmente lógica) del agente que se ejecuta directamente para generar su conducta, pudiendo verificar propiedades de la especificación. Ej. *METATEM* y DESIRE [15].

1.3.4 Tipología de agentes

La metáfora de los agentes ha superado el dominio de los laboratorios, interesando a varias empresas en el desarrollo de productos comerciales basados en agentes. La principal diferencia entre estos agentes radica en su especialización. Son agentes de acuerdo a la noción débil citados anteriormente (1.3.1). A continuación se describe brevemente los tipos de agentes más conocidos [16]:

- Agentes de interfaz
- Agentes móviles
- Agentes de Internet/información

Muchos investigadores por falta de una definición de agente han creado sinónimos añadiendo algo de confusión. Claro que hay buenas razones para tener tales sinónimos. Primero los agentes vienen en diferentes apariencias: por ejemplo los que habitan en un mundo físico, como una fábrica, son llamados robots, los que habitan en una vasta red de computadoras son referidos como softbots (software robots), aquéllos que realizan tareas específicas son llamados taskbots (Task based robots) y autonomous agents se refiere típicamente a agentes móviles o robots que operan en ambientes dinámicos e inciertos. Segundo, los agentes pueden jugar roles específicos, por lo tanto tenemos asistentes personales o Knowbots (Knowledge-based robots), que pueden tener el conocimiento de un experto en un dominio específico del saber humano [17].

1.4 Utilidad de los agentes

Las propiedades que poseen los agentes hacen que la utilidad de éstos sea diferente al “software” tradicional, por tanto la utilidad se toma en función de sus propiedades.

Por ejemplo uno de los objetivos en programar agentes es apoyar al usuario a realizar un conjunto de acciones repetitivas que pueden ser automatizadas, o por el contrario, emprender proactivamente acciones no contempladas y que deben modificarse con el tiempo por diferentes tipos de dinámicas, y que se realizarán basándose en ciertos módulos muy especializados de aprendizaje.

Los agentes son útiles cuando se tienen múltiples capacidades simples y complejas integradas en la misma aplicación, como por ejemplo, cuando son intermediarios en una transacción de comercio electrónico o financiera importante y, pueden además enviar un simple correo electrónico. La figura 1.11 muestra un esquema de utilidad de los agentes, que parte de una clasificación general de agentes, detallando los agentes de software que comprende a los sistemas de agentes y los agentes individuales. Dentro de los sistemas de agentes se observa

que existen distintos enfoques de sistemas de agentes según la utilidad que pretenda darse.

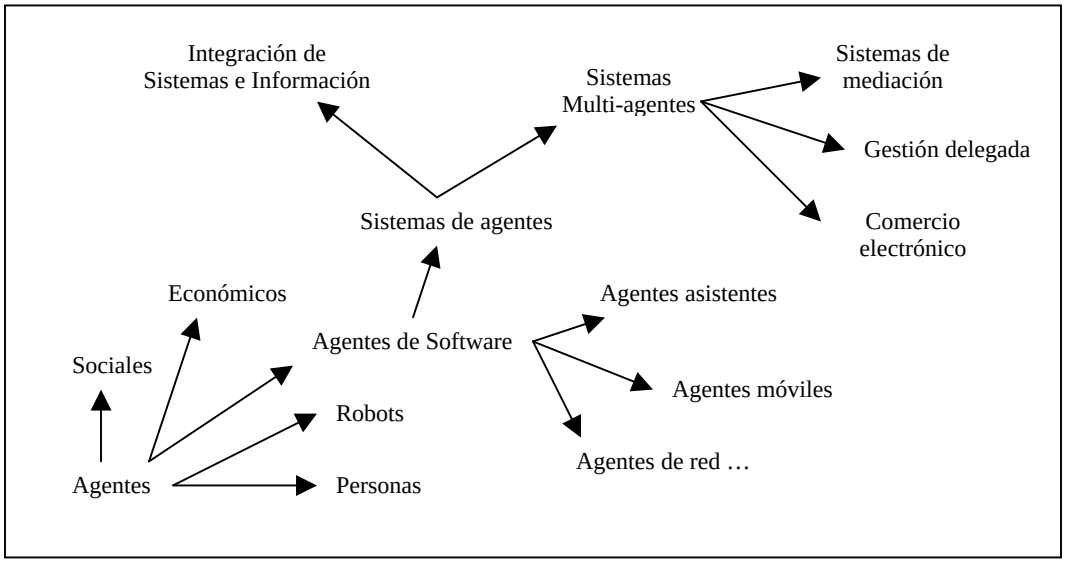

Fig. 1.11 Utilidad de los agentes

La utilidad también se da cuando los agentes tienen la posibilidad de comunicarse y cooperar entre ellos para realizar una tarea en común de una manera más fácil y eficiente, y si entran en conflicto entonces negocían para resolverlo.

Otra utilidad de los agentes diferente al del “software” tradicional es que éstos actúan continuamente, es decir, son procesos que corren temporal y continuamente en el tiempo activándose cuando se cumplen ciertas condiciones o sucede algún evento particular que puede ser detectado.

Son útiles para aplicaciones que deben ser flexibles y para aquellas aplicaciones que requieren la capacidad de movilidad del agente de una computadora a otra en la red [18].

CAPÍTULO 2
Estudio de Herramientas para el Desarrollo de Agentes

2.1 Introducción

En el presente capítulo se muestra un esbozo de las metodologías para desarrollar agentes, así como las herramientas para la creación de éstos. Se inicia con la definición de metodología (sección 2.2), a continuación se presentan los enfoques de las metodologías orientadas a agentes (sección 2.3) y por último se despliega información de las herramientas más conocidas para el desarrollo de la tecnología de agentes (sección 2.4). Al final de las dos secciones anteriores se dan los aspectos que se tomaron en cuenta para la elección tanto de la metodología como de la herramienta para la construcción del MAS respectivamente.

2.2 Metodologías

Una vez que ha comenzado a establecerse la tecnología de agentes, y se han desarrollado diversas plataformas y lenguajes para emplear MAS en variadas aplicaciones, han comenzado a surgir metodologías que tratan de asistir en el ciclo de vida de la construcción de los MAS para obtener las ventajas de reciclaje de los sistemas y mantenimiento, entre otras.

Como punto de partida se define qué es una metodología. Una metodología en un sentido amplio, es un conjunto de métodos y técnicas que se siguen para ayudar en el desarrollo de un producto de software.

Rumbaugh señala:

"Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convenciones en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso… Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo" [19].

La tecnología de agentes ha recibido gran atención en los últimos años y, como resultado, la industria está comenzando a interesarse en esta tecnología para desarrollar sus propios productos. Empresas como IBM, Microsoft, Mitsubishi cuentan con laboratorios de agentes inteligentes y han sacado sus productos al mercado, principalmente sobre agentes de usuario e Internet. Sin embargo, para que el paradigma de la programación orientada a agentes se extienda para desarrollar aplicaciones genéricas, es necesario que se desarrollen técnicas de análisis y diseño con este paradigma [20].

2.3 Metodologías Orientadas a Agentes

Después de una revisión de metodologías orientadas a agentes, se ha encontrado que no han surgido metodologías orientadas a agentes totalmente nuevas debido a la relación de este paradigma con el paradigma orientado a objetos y con los sistemas basados en conocimiento. Por tal razón se han planteado extensiones de las metodologías existentes, pudiéndose distinguir dos enfoques principales [5]:

- Extensión de metodologías orientadas a objetos.
- Extensión de metodologías de ingeniería del conocimiento.

2.3.1 Extensión de Metodologías Orientadas a Objetos

Parten de las metodologías orientadas a objetos añadiendo las peculiaridades de los agentes: creencias, objetivos, planes, cómo identificar agentes, relaciones e interacciones entre agentes, etc. La razón del por qué se puede partir de este tipo de metodologías se encuentra justificada en los siguientes puntos citados por Iglesias et al [20].

1. Existen similitudes entre el paradigma Orientado a Objetos y el paradigma de Agentes, tal como lo señala Shoham [21] "los agentes pueden ser considerados como objetos activos, con estados mentales".

2. Ambos paradigmas usan el paso de mensajes para la comunicación.
3. La popularidad y la experiencia que se tienen de las metodologías orientadas a objetos.
4. El uso de lenguajes orientados a objetos para la implementación de agentes.

Las metodologías que presentan un enfoque orientado a objetos se describen en las secciones siguientes.

2.3.1.1 Análisis y Diseño Orientado a Agentes, de Burmeister (ADOAB)

Burmeister define tres modelos para analizar un sistema agente: el modelo de agente, que se limita al agente y a su estructura interna (creencias, planes, objetivos); el modelo organizacional, que describe la relación entre agentes (herencia y roles en la organización); y el modelo de cooperación, que describe la interacción entre los agentes [20].

Los pasos del proceso de desarrollo de cada modelo son:

- *Modelo de Agentes*: se realiza la identificación de los agentes y sus ambientes utilizando una extensión de las tarjetas CRC (Clases-Responsabilidades-Colaboraciones) para incluir creencias, motivaciones, planes y atributos de cooperación.
- *Modelo Organizacional*: propone la identificación de los roles de cada agente y la elaboración de diagramas utilizando la notación OMT (Técnica de Modelado de Objetos) para la jerarquía de herencia y las relaciones entre los agentes.
- *Modelo de Cooperación*: se identifican las cooperaciones y los modelos de cooperación, los tipos de mensajes que se intercambian entre agentes, además se analizan los protocolos utilizados.

2.3.1.2 Técnicas de Modelado de Agentes para Sistemas de agentes BDI (BDI)

Este método define dos niveles principales para el modelado de agentes BDI [20].

El nivel externo consiste en descomponer el sistema en agentes y la definición de sus interacciones. Esto se lleva a cabo a través de dos modelos:

- Modelo de agentes: describe la jerarquía de relaciones entre agentes y las relaciones entre agentes concretos.
- Modelo de interacción: define las responsabilidades, servicios e interacciones entre agentes y sistemas externos.

El nivel interno lleva a cabo el modelado de cada clase de agente BDI a través de tres modelos:

- Modelo de creencias: describe las creencias acerca del ambiente.
- Modelo de objetivos: detalla los objetivos y eventos que un agente puede adoptar o responder.
- Modelo de planes: puntualiza los planes que un agente debe utilizar para lograr sus objetivos.

El proceso de desarrollo del nivel externo, se inicia con la identificación de los roles (funcionales, organizacionales, etc.) del dominio de la aplicación, a fin de identificar los agentes y organizarlos en una jerarquía de clases de agentes descrito utilizando una notación parecida a la OMT. Luego las responsabilidades asociadas a cada rol son identificadas, junto con los servicios proveídos y utilizados para cumplir con las responsabilidades. El siguiente paso es la identificación de las interacciones necesarias para cada servicio. Finalmente, se colecta la información en un modelo de instancia de agente.

El desarrollo del nivel interno, se inicia con el análisis de las diferentes formas (planes) de lograr un objetivo, los planes para responder a un evento. Finalmente, las creencias del agente acerca de objetos del ambiente se modelan y representan utilizando la notación OMT.

2.3.1.3 Método para Multi-Agentes basado en Escenarios (MASB)

Este método está destinado a ser aplicado a MAS en el campo del trabajo cooperativo [20]. La fase de análisis consiste de las siguientes actividades:

- Descripción de Escenarios: identificación de los principales roles de los usuarios, agentes, los objetos del ambiente y los escenarios típicos en lenguaje natural.
- Descripción funcional de los roles: descripción de los roles de los agentes utilizando diagramas de comportamiento, que describen los procesos, la información relevante y la interacción entre los agentes.
- Modelado conceptual del dato y el mundo: modelado del dato y el conocimiento utilizado por el agente empleando diagramas entidad-relación y diagramas de ciclo de vida de entidades.
- Modelado de la interacción sistema-usuario: simulación y definición de diferentes interfaces para la interacción humano-máquina en todos los escenarios.

La fase de diseño consiste de las siguientes actividades:

- Descripción de la arquitectura y el escenario del sistema MAS: selección de los escenarios a ser implementados y los roles de los agentes en dichos escenarios.

- Modelado de Objetos: refinamiento del modelado del mundo definido en el análisis, definición de jerarquías, atributos y procedimientos.
- Modelado de Agentes: especificación como estructuras de creencias, de los elementos definidos en el modelado conceptual del análisis.

2.3.1.4 Metodología Orientada a agentes para el Modelado de Empresas

Esta metodología propone la combinación de metodologías orientadas a objetos OOSE (Ingeniería de Software Orientada a Objetos) y metodologías de modelado de empresas IDEF (Integration Definition for Function modelling) [20]. Los modelos identificados son:

- Modelo Funcional: describe las funciones (entradas, salidas, mecanismos y control) utilizando diagramas IDEF que incluyen la selección de los métodos posibles dependiendo de las entradas y el control.
- Modelo de Casos de Uso: describe a los actores envueltos en cada función, utilizando la notación de casos de uso OOSE.
- Modelo Dinámico: este modelo está destinado a analizar las interacciones entre objetos. Los casos de uso se representan en diagramas de transición de eventos.
- El sistema Orientado a Agentes: es un compuesto de:
    - Identificación de agentes: los actores de los casos de uso son identificados como agentes.
    - Protocolos de coordinación o scripts: se describen en diagramas de estados.
    - Invocación de planes: diagramas de secuencia extienden a los diagramas de transición de estados a fin de incluir condiciones para indicar cuando se invoca a un plan.
    - Creencias, Sensores y Efectores: las entradas de las funciones deben ser modeladas como creencias u obtenidos de los objetos vía sensores, y los objetivos satisfechos deben ser modelados como cambios a las creencias vía efectores.

2.3.1.5 Ingeniería de Sistemas Multi-Agentes (MaSE)

Esta metodología sigue los siguientes pasos: Diseño a nivel del Dominio, Diseño a nivel del Agente, Diseño de Componente y Diseño del Sistema [22].

Los pasos en el diseño a nivel del Dominio son:

1. Identificar los tipos de agentes.
2. Identificar las posibles interacciones entre los tipos de agentes.
3. Definir protocolos de coordinación para cada tipo de interacción.

Los pasos específicos de diseño a nivel de agentes son:

1. Mapear las acciones identificadas en la conversación de los agentes a componentes internos.
2. Definir estructuras de datos identificadas en la conversación de agentes. Esas estructuras de datos representan entradas o salidas de los agentes.
3. Definir estructuras de datos adicionales, internas al agente. Estas estructuras de datos representan los flujos de datos entre los componentes en la arquitectura.

Diseño de componentes: Una vez que la arquitectura del agente ha sido definida, los componentes especificados deben ser diseñados.

Los pasos específicos del diseño del sistema incluyen:

1. Seleccionar los tipos de agentes que son necesarios.
2. Determinar el número de agentes requeridos para cada tipo y definir:

    a. La ubicación física o dirección de los agentes.
    b. Los tipos de conversaciones que los agentes serán capaces de mantener.
    c. Cualquier otro parámetro definido en el dominio.

2.3.1.6 La Metodología GAIA para el Análisis y Diseño Orientado a Agentes

Esta metodología de Wooldridge, Jennigs y Kinny para el análisis y diseño orientado a agentes, maneja el macro nivel (Sociedad de agentes y estructura organizacional) y el micro nivel (estructura de agentes) de desarrollo de agentes [23].

Esta metodología está compuesta por dos fases: de análisis y diseño. El análisis comprende los siguientes módulos:

- *Modelo de Roles*: Identifica los roles claves en el sistema. Un rol puede ser visto como una descripción abstracta de las funciones esperadas de una entidad, o en otros términos, un rol puede ser visto más o menos como la noción de oficio. Este modelo contiene cuatro atributos:

    - *Responsabilidades*: La función de un rol está definido por su responsabilidad.
    - *Permisos*: Representan lo que los roles permiten hacer, en particular a que información permite acceder.
    - *Actividades*: Son tareas que un rol desempeña sin interactuar con otros roles.
    - *Protocolos*: Son los modelos específicos de interacción.

- *Modelo de Interacción:* Trata las dependencias y relaciones entre los roles en una organización de multi-agente. Define los siguientes protocolos:

  - *Propósito*: Breve descripción del objetivo.
  - *Iniciador*: Los roles responsables de comenzar la interacción.
  - *Receptor*: Los roles con quien interactúa.
  - *Entradas*: Información utilizada por Iniciador.
  - *Salidas*: Información entregada.
  - *Procesamiento*: Breve descripción del proceso.

La fase de diseño contiene los módulos que a continuación se muestran:

- *Modelo de Agentes:* Mediante este modelo se busca documentar los diferentes tipos de agentes que utilizará el sistema en su desarrollo y documentar las instancias que simularán los agentes en ejecución.

- *Modelo de Servicios:* Este modelo se utiliza para identificar los servicios asociados con cada uno de los roles de los agentes y la especificación de las principales propiedades de estos servicios.

- *Modelo de Comunicaciones:* Define los enlaces de comunicación entre los tipos de agentes.

2.3.2 Extensión de Metodologías de Ingeniería del Conocimiento

Las metodologías de ingeniería del conocimiento pueden proporcionar una buena base para modelar MAS ya que estas metodologías tratan el desarrollo de sistemas basados en conocimiento. Dado que los agentes tienen características cognitivas, estas metodologías pueden proporcionar las técnicas de modelado de la base de conocimiento de los agentes.

La extensión de metodologías de conocimiento pueden aprovechar la experiencia adquirida en dichas metodologías. Además se pueden emplear las bibliotecas de métodos de resolución de problemas y ontologías así como las herramientas desarrolladas en estas metodologías.

Se han propuesto varias extensiones para desarrollar sistemas Multi-agentes basándose en la metodología KADS o en su sucesora CommonKADS, las cuales se describen en las siguientes secciones [24].

2.3.2.1 Metodología CoMoMAS

CoMoMAS (Contribución a la adquisición y modelado del conocimiento en un entorno multi-agente) propone una extensión de CommonKADS para modelar MAS a través de desarrollo de los siguientes modelos [20]:

- *Modelo de agente:* Es el modelo central de la metodología. Define la arquitectura del agente y el conocimiento del mismo, que se clasifica en social, cooperativo, cognitivo, reactivo y de control.
- *Modelo de experiencia:* Describe las competencias cognitivas y reactivas del agente. Distingue entre conocimiento de tareas, de resolución de problemas y reactivo. *El conocimiento de las tareas* contiene la descomposición de las tareas, descrita en el modelo de tareas. El *conocimiento de resolución de problemas* describe los métodos de resolución de problemas (PSM) y las estrategias para seleccionarlos. El *conocimiento reactivo* describe los procedimientos para responder a un estímulo.
- *Modelo de tareas:* Describe la descomposición de las tareas indicando si son solucionadas por el usuario o por un agente.
- *Modelo de cooperación: D*escribe la cooperación entre varios agentes. Se descompone en métodos de resolución de conflictos (métodos de negociación y cooperación) y conocimiento de cooperación (primitivas, protocolos y terminología de interacción).
- *Modelo del sistema*: Representa la organización de los agentes y describe la arquitectura del MAS y de sus agentes.
- *Modelo de diseño*: define cómo pasar de los modelos previos a código ejecutable. Recoge los requisitos funcionales y no funcionales de la aplicación, así como las guías de diseño (plataforma y lenguaje de implementación) y la historia del diseño.

2.3.2.2 Metodología MAS-CommonKADS

Esta metodología extiende el modelo definido en CommonKADS, agregando técnicas de metodologías orientadas a objetos [25].

La metodología inicia con una fase de conceptualización que es una fase informal destinada a colectar los requerimientos del usuario y obtener una primera descripción del sistema desde el punto de vista del usuario. Para este propósito se utilizan técnicas de OOSE.

La metodología incluye los siguientes modelos:

- Modelo de Agentes: Describe las características principales de los agentes, incluyendo las capacidades de razonamiento, habilidades (sensores/efectores), servicios, objetivos, etc.

- Modelo de Tarea: Describe las tareas llevadas a cabo por el agente, y la descomposición de tareas.
- Modelo de Coordinación: Describe las conversaciones entre los agentes, esto es, sus interacciones, protocolos y capacidades requeridas.
- Modelo de Organización: Describe la organización en la cual el MAS será introducido y la organización de la sociedad de agentes.
- Modelo de Comunicación: Detalla las interacciones humano-agente, y los factores humanos para desarrollar estas interfaces.
- Modelo de Diseño: Reúne los modelos previos y se subdivide en tres submodelos:
    - diseño de la aplicación: composición o descomposición de los agentes del análisis.
    - diseño de la arquitectura: diseño de los aspectos relevantes de la red de agentes.
    - diseño de la plataforma: selección de la plataforma de desarrollo de los agentes para cada arquitectura de agente.

2.3.3 Selección de la metodología para el diseño de agentes

De las metodologías citadas se pueden encontrar varias similitudes, dado que todas ellas pretenden identificar: los agentes que requiere el sistema, las tareas que van a desempeñar cada uno, las interacciones que existen con otros agentes y la forma en la cual van a obtener la información de su entorno para actuar. Las diferencias que existen entre ellas son el énfasis o profundidad con que se tratan algunos puntos o bien la ausencia de ellos en otras metodologías, lo cual hacen más o menos atractivas las metodologías a los diseñadores de sistemas de agentes.

A continuación se mencionan características de las metodologías en las cuales puse más interés para el desarrollo del prototipo.

- **ADOBA**, esta metodología de algún modo no profundiza por medio de sus modelos en la descripción de tareas y servicios que pueda proporcionar el agente. En el modelo de organización no se alcanzan a describir propiamente cómo se encuentran organizados los agentes.
- **BDI**, deja de lado algunos otros aspectos importantes para el diseño de MAS como la coordinación que debe existir entre algunos agentes y la organización que deben guardar éstos. Aunque se profundiza en aspectos para llevar a cabo sus cometidos.
- **MASB**, la propuesta planteada es interesante ya que presenta aspectos sobre todo el entorno en el que va a trabajar el MAS, como lo es el usuario, así como la descripción de la organización sobre la cual se piensa trabajar.
- **Metodología Orientada a Empresas**, presenta un buen detalle de los pasos para obtener una descripción detallada de los estados internos del agente y la coordinación con otros.

- **MaSE**, cuenta con buenos recursos para identificar las tareas y el entorno que envuelve al agente, las comunicaciones y su estructura**.**
- **GAIA**, parte de la identificación de aspectos generales y los va detallando a través de cada uno de los módulos para que posteriormente en el módulo de diseño se integra la información. Para esta metodología no se encuentra una sección que plasme la estructura organizacional de la sociedad de agentes.
- **CoMoMAS**, es interesante ya que dentro de su modelo de experiencia se plantea de acuerdo a qué criterios o conocimientos se soluciona un problema.
- **MAS-CommonKADS**, lo interesante en esta metodología es que se parte de una etapa donde se recolecta la información de los requerimientos del usuario y durante otra etapa contempla la relación entre los agentes y los usuarios.

Después de revisar las características mencionadas de cada una de las metodologías para la construcción de MAS y presentándose una descripción de las metodologías más conocidas es el momento de dar los motivos por los cuales me inclino por la metodología MAS-CommondKADS. Los criterios para elegir la metodología han sido:

- Cuenta con una etapa inicial que permite identificar los componentes principales del sistema de forma general.
- Abarca de manera profunda la especificación de las características de inteligencia de los agentes, ya que se utilizan técnicas de Ingeniería del Conocimiento.
- Cada uno de los modelos pertenecientes a la metodología poseen puntos específicos que permiten la construcción del sistema.
- Metodología con gran aceptación por parte de los desarrolladores de MAS.
- Es una metodología de la cual se encontró una amplia documentación.

2.4 Herramientas para la creación de agentes

En la presente sección se procede a realizar una descripción de las herramientas existentes para la creación de agentes. Aunque estas no son todas las herramientas que existen en el mercado para la construcción, sí son las de mayor uso, ya sea en instituciones de investigación o en empresas. De las herramientas encontradas a mi criterio realice una clasificación de acuerdo al tipo de especificación de su uso, éstas son:

- Herramientas para desarrollar MAS estándar.
- Herramientas para desarrollar MAS especializados en la movilidad.

2.4.1 Herramientas para desarrollar MAS estándar

Las herramientas diseñadas para la construcción de MAS estándar cuentan con mecanismos que permiten elaborar agentes en una amplia gama de aplicaciones, es decir, no son herramientas que son especializadas en la construcción de un tipo de agentes. Entre estas herramientas encontradas se tienen las que se describen en las siguientes secciones.

2.4.1.1 AgentBuilder

AgentBuilder es un ambiente integrado para la construcción de agentes [26]. Consiste de dos componentes principales: el *Toolkit* y el *Sistema de Ejecución*. El Toolkit incluye herramientas que permiten administrar los procesos de desarrollo de software basado en agentes, el análisis del dominio de las operaciones de agentes, el diseño e implementación de redes de agentes que se comunican, la definición de los comportamientos individuales de los agentes, así como también la prueba y depuración de dichos sistemas. El Sistema de Ejecución incluye un agente motor que proporciona el ambiente para la ejecución de software de agentes.

Su modelo de agentes está basado en la arquitectura BDI que mantiene una base de datos que representan las creencias del agente y un conjunto de reglas de comportamiento. Está escrito en *Java* y cada agente creado con esta herramienta es un programa *Java* que puede ser ejecutado en cualquier plataforma que soporte *Java*. Cada agente construido desde *AgentBuilder* se comunica usando KQML. Soporta la integración y el uso de librerías de software existentes (Java, C y C++), CORBA y protocolos IIOP, sockets TCP/IP y HP E-Speak.

AgentBuilder no permite la movilidad de agentes en tiempo de ejecución, la localización de agentes y recursos se puede realizar empleando un directorio facilitador, no soporta el trabajo con base de datos.

Las herramientas de AgentBuilder permiten al desarrollador diseñar agentes en un ambiente de construcción sencillo y rápido con código Java. Las siguientes son las herramientas con las que cuenta:

- Herramientas de control de proyectos.- provee ayuda al desarrollador para administrar todos los procesos de desarrollo del agente. Incluye el manejador del proyecto, las herramientas del diccionario del proyecto y un manejador del contenedor del proyecto.
- Administrador de Ontologías.- asiste al desarrollador en el análisis del dominio del problema, en el uso del agente, identifica y define los conceptos de operación del agente dentro de su dominio.

- Administrador de la plataforma.- ayuda al desarrollador a construir agentes que se comunican y cooperan con la tarea que cada uno desempeña.
- Administrador del agente.- provee herramientas para definir un modelo mental de un agente individual y su comportamiento. Las herramientas de definición del agente incluyen editores gráficos para definir las diferentes construcciones mentales que hacen al agente. En suma, el administrador del agente soporta herramientas para añadir planeación y aprendizaje a las capacidades del agente.
- Depurador del agente.- provee de herramientas en tiempo de ejecución para la comunicación con un agente que se encuentra en ejecución. El depurador cuenta con software de desarrollo con salidas gráficas que describen el modelo mental del agente, entradas y salidas de mensajes.

Las capacidades de estos agentes son una asociación entre una acción y las precondiciones necesarias para ejecutar la acción. Ésto representa una limitación debido a que no cuenta con estructuras para deliberación, razonamiento y/o aprendizaje.

Ventajas

Una característica destacable de AgentBuilder es la posibilidad de construir agentes casi sin programar, ya que provee un ambiente de desarrollo para editar las capacidades y características de los agentes, y depurarlos durante su ejecución. Soporta una gran variedad de características Web incluidas formas, cookies, etc.. Permite la distribución de la herramienta en diversos sistemas operativos.

Desventajas

La arquitectura general de los agentes no puede ser redefinida ni adaptada, debido a que ésto forma parte del lenguaje RADL (Reticular Agent Definition Language). La mayor limitación del *toolkit* consiste en que no posee ningún componente para *planning* o aprendizaje. Como consecuencia de ésto, los agentes construidos siguen patrones de comportamiento fijos.

Requisitos del sistema.

La distribución AgentBuilder Pro está distribuido con el JRE (Java Runtime Environment) para el soporte de la plataforma, se encuentra disponible para las siguientes plataformas: Solaris, Windows 98/NT/2000/XP y Linux. Los requerimientos mínimos son una máquina con procesador Pentium 266 o equivalente con 128 MB en RAM.

2.4.1.2 JATLite

El framework de la Universidad de Stanford para la construcción de MAS llamado JATLite [27] está implementado en Java y provee un conjunto de paquetes Java con:

- Herramientas para la comunicación, a través del intercambio de mensajes KQML usando TCP/IP.
- Servidor de nombres de agentes, con la información de todos los nombres y direcciones de los agentes existentes.
- Funcionalidad para que los Applets Java intercambien mensajes con cualquier agente registrado en Internet.

Provee en general la funcionalidad básica para construir aplicaciones de MAS. Tanto el servidor de nombres como la funcionalidad para el intercambio de mensajes son centralizados. Posee un interfaz para el desarrollo de agentes y otro para el control de su ejecución, los agentes pueden crearse local o remotamente. La programación puede ser en Java o bien en C++.

Utiliza el AMR (Agent Message Router) que permite a los agentes registrarse en la plataforma utilizando un “login” y un “password”, conectarse o desconectarse a Internet automáticamente, o enviar y recibir mensajes por FTP, proporcionando una gran robustez a los desarrolladores. La movilidad es controlada también por el AMR.

JATLite está compuesto por capas las cuales se describen brevemente a continuación:

- *Capa Abstracta* . Proporciona una colección de clases abstractas necesarias para la implementación de JATLite.
- *Capa Base*. Proporciona la comunicación básica sobre los protocolos en TCP/IP.
- *Capa KQML*. Permite el almacenamiento y análisis de los mensajes KQML.
- *Capa Router*. Proporciona nombres a los agentes, y almacenamiento y direccionamiento para los mensajes. Cuando un agente se desconecta ya sea intencionadamente o por una caída en el sistema, el *Router* almacena los mensajes entrantes dirigidos a él hasta que el agente se reconecta.
- *Capa de Protocolo*. Soporta los diversos servicios de Internet como SMTP, FTP, POP3 o HTTP.

Ventajas

Permite crear rápidamente nuevo software de agentes con una robusta comunicación sobre Internet, además los Applets Java intercambian mensajes con cualquier agente registrado en Internet, permite una fácil construcción de agentes, particularmente los que se comunican por medio del envío y recepción

de mensajes, también puede ser implementado en diferentes máquinas que cuenten con diferentes sistemas operativos.

Desventajas

Una limitación de esta herramienta es que sólo apoya la coordinación centralizada de los agentes a través de ruteadores, cuando los MAS deben caracterizarse por su descentralización y un bajo grado de control global para poder explotar todo el potencial de los MAS, ya que de lo contrario pueden producir cuello de botella en el sistema cuando se trate de una aplicación real con varios agentes ejecutándose simultáneamente. No facilita el uso de herramientas de razonamiento ni establece ninguna estructura para autonomía, percepción, etc. En general JATLite permite desarrollar unidades que intercambian mensajes a fin de desarrollar una tarea (agentes reactivos).

Requisitos del sistema.

El único requisito necesario para ejecutar JATLite es que la plataforma soporte el JDK 1.1 o superior, como son *Sun Microsystems Inc*, Windows 95/98/NT, Solaris o Mac OS.

2.4.1.3 JADE

JADE (Java Agent Development Framework) un software de ambiente de trabajo implementado totalmente en lenguaje Java. Simplifica la implementación de MAS a través de un middle-ware[1] que demanda para cumplir con las especificaciones de FIPA[2] y a través de un juego de herramientas que soportan la depuración y despliegue de fase [27]. Esta plataforma de agentes puede ser distribuida a través de máquinas y la configuración puede ser controlada vía remota mediante la Interfaz Gráfica de Usuario (GUI) así como para la administración remota, monitoreo y control del status de los agentes, permitiendo, por ejemplo parar y reiniciar agentes. El GUI permite también crear e iniciar la ejecución de un agente en un host[3] remoto, con tal de que el contenedor de agentes esté ya corriendo, controlar otras plataformas remotas que sean FIPA compatibles. La configuración puede ser incluso cambiada en tiempo de ejecución por un agente móvil de una máquina a otra, como y cuando se requiera. Emplea el lenguaje de comunicaciones estándar de FIPA (ACL, Agent Communication Language), además de permitir la movilidad de agentes. JADE no presenta una interfaz para el desarrollo de sistemas de agentes.

[1] Middle-ware. Software que se coloca entre una computadora cliente y un servidor. Éste garantiza que se transmita la información aunque corran sobre plataformas distintas.
[2] FIPA. Foundation for Inelligent Physical Agents, organización formada en 1996 para producir estándares sobre software de agentes heterogéneos y sistemas de agentes.
[3] Host. Máquina anfitriona.

El objetivo de JADE es el desarrollo de MAS simples mientras se asegura que los estándares se cumplan a través de un comprensivo juego de servicios de sistemas y agentes siguiendo las especificaciones de la FIPA:

- Servicio de nombres y servicio de página amarilla.
- Transporte de mensajes y paso de servicios.
- Librerías de FIPA listas para ser usadas.

La plataforma de agentes JADE cumple con las especificaciones de la FIPA e incluye todos los componentes que obliga para la administración de la plataforma, que son :

- El ACC (Canal de Comunicación de Agentes).
- El AMS (Servicio Administrador de Agentes).
- El DF (Directorio Facilitador).

En su parte medular JADE tiene varias herramientas (agentes) gráficas que han sido complementadas para soportar la fase de depuración, normalmente basados en un sistema de distribución compleja. Estas herramientas se enlistan a continuación:

- El *Remote Management Agent (RMA)* actúa como consola gráfica para la gestión y control de la plataforma. Es necesaria para iniciar el resto de herramientas.
- El *Dummy Agent* es una herramienta de depuración y visualización, en la que es posible componer mensajes ACL y enviarlos, visualizar la lista de todos los mensajes ACL enviados o recibidos, así como la información contenida en dichos mensajes.
- El *Sniffer* es un agente capaz de interceptar mensajes ACL y mostrarlos gráficamente utilizando una notación similar a los diagramas de secuencia de UML. Es bastante útil para depurar las sociedades de agentes mediante la observación de cómo intercambian mensajes ACL.
- El *SocketProxyAgent* es un agente simple que actúa como enlace bidireccional entre una plataforma JADE y una conexión TCP/IP, y se utiliza para canalizar los mensajes entrantes y salientes de una plataforma JADE.
- El *DF GUI* es interfaz gráfico utilizado junto con el *Directory Facilitator* de JADE para crear complejas redes de dominios y subdominios. Esta interfaz permite controlar el conocimiento de los DFs y registrar, modificar o buscar dentro de ellos.

Ventajas

Soporta la implementación de Sistemas Multi-agentes FIPA compatibles, y permite la distribución de la plataforma en diferentes ambientes para la ejecución de los agentes, así como controlar las demás plataformas FIPA compatibles, permite monitorear casi cualquier aspecto del agente, (comunicación, mensajes, controlarlos, migrarlos, clonarlos, etc.), Simplifica la implementación de MAS a

través de un middle-ware, la distribución puede ser distribuida en diferentes máquinas con diferentes sistemas operativos. Es un software abierto lo que permite realizar modificaciones para ser adaptado a un entorno o con las características requeridas.

Desventajas

La documentación existente de JADE es un poco confusa, y no explica en detalle todos los aspectos que maneja. El no contar con una interfaz para la desarrollo de agentes involucra más tiempo en llevar a cabo los MAS.

Requerimientos del sistema.

JADE requiere al menos contar con JDK 1.2 Run Time o posteriores.

2.4.1.4 ZEUS

Los laboratorios de British Telecom desarrollaron el marco de trabajo para MAS Zeus [28]. La plataforma está escrita en Java, utiliza tecnología estándar y escalable (JFC/Swing[4], TCP/IP, KQML, FIPA) y es una fuente abierta, no permite la movilidad de agentes en tiempo de ejecución. El entorno de desarrollo de MAS está constituido por los siguientes grupos funcionales:

- Librería de componentes de agentes.
  Es una colección de clases que forman el bloque de construcción de agentes individuales, además implementan las aplicaciones funcionales independientes requeridas a nivel de agente para agentes colaborativos, permite la construcción de agentes genéricos ZEUS independientes de las aplicaciones. Incluye librerías para:
  - Comunicación
  - Ontologías
  - Coordinación.

- Herramienta para la construcción de agentes.
  Para facilitar el rápido desarrollo, el juego de herramientas de ZEUS provee un alto nivel de desarrollo que oculta la complejidad de la librería de componentes de agentes para el desarrollador de agentes. Contiene una suite de editores que soportan el diseño de agentes ZEUS, los cuales han sido diseñados para permitir al usuario la especificación visual de los atributos en la creación de los agentes. La suite de editores contiene:
  - Editor de Ontologías, para definir los elementos de la ontología en el dominio. Los conceptos son definidos en términos de sus atributos y de

[4] JFC/Swing. Fundación de clases de Java para crear componentes para la interfaz gráfica del usuario.

rangos de valores válidos para cada atributo. Los valores de los atributos pueden ser del tipo primitivo, listas, expresiones constantes.

- Editor de Hechos, para describir las especificaciones por ejemplo de hechos y variables, usando las plantillas creadas por el editor de ontologías.
- Editor de definición de agentes, para describir la lógica de los agentes. Esto involucra la especificación de las tareas de agentes, su estado inicial y las dimensiones de los planes diarios.
- Editor de descripción de tareas, para definir la relación organizacional entre agentes y creencias de los agentes sobre las habilidades de otros agentes.
- Editor de coordinación, para elegir el tipo de protocolos de coordinación con el cual cada agente será equipado.

Así, para generar el código específico de la aplicación, la herramienta generadora de código lo hereda de la librería de componentes de agentes, y lo integra con los datos de los diversos editores visuales. Los programas resultantes pueden ser compilados y ejecutados normalmente.

- Conjunto de utilerías de agentes:
  - Servidor de nombres.
  - Agente facilitador que proporciona el descubrimiento de información de un agente.
  - Agente visualizador, para visualizar, analizar o depurar la sociedad de agentes ZEUS. Este visualizador contiene herramientas para analizar desde diferentes perspectivas a los agentes, estas herramientas son:

    - Visualizador de sociedad, muestra toda la sociedad de agentes y su inter-relación organizacional.
    - Herramienta de Reportes, muestra la descomposición/distribución de las actividades, tareas de toda la sociedad y los estados de ejecución de tareas.
    - Visualizador de agente, permite observar y monitorear los estados internos de los agentes.
    - Herramientas de control, usadas para la inspección remota o modificación de los estados internos de los agentes individualmente.
    - Herramientas estáticas, despliega en varios formatos información estática de los agentes en forma individual o de la sociedad de agentes.

Ventajas

El razonamiento de los agentes sigue un modelo comercial típico, en el cual existen recursos, productores y consumidores que lo simplifican en el momento de crear sistemas comerciales. Posee editores para cualquier componente de un

agente los cuales facilitan el desarrollo de MAS. Permite la utilización de KQML o ACL para la comunicación de agentes, la distribución de la herramienta es permitida en diferentes sistemas operativos.

Desventajas

Las clases del *framework* no están diseñadas para ser redefinidas por el programador, además de que no es posible personalizar o adaptar los componentes del agente utilizando otras clases sin realizar modificaciones considerables al conjunto de clases básicas. En general, no permite definir métodos de las capacidades básicas de un agente en el *toolkit*, ni tampoco posee un formalismo de alto nivel para especificar el comportamiento del agente, tales como RADL.

Requerimientos del sistema.

ZEUS se ejecuta en cualquier plataforma que cuente con JDK 1.2 ó posteriores, cada máquina deberá ser compatible con comunicaciones TCP/IP, pero no es necesario tener instalado ningún servicio middle-ware. ZEUS ha sido comprobado exitosamente en Windows 95/98/NT4 y plataformas Solaris.

2.4.1.5 FIPA-OS

FIPA-OS (FIPA Open Source) es una plataforma de agentes abierta creada por Nortel Networks en los laboratorios del Reino Unido [29]. FIPA-OS está implementado en Java. La plataforma soporta la comunicación entre múltiples agentes usando un lenguaje de comunicación de agentes el cual obedece a los estándares de agentes FIPA. Un enfoque dominante de la plataforma es que apoya la claridad. Ésto es apoyado naturalmente por el paradigma de agentes mismo y las innovaciones para apoyar la comunicación de agentes que pueden ocurrir en varias áreas claves. FIPA-OS está siendo desarrollado en varios dominios, incluidas las redes virtuales privadas. Se ha estado definiendo para la interoperatividad con otras plataformas FIPA compatibles.

El modelo de referencia FIPA habla de una fácil definición de los componentes principales de la distribución de FIPA-OS los cuales son:

- El directorio Facilitador (DF)
- El sistema de administración de agentes (AMS)
- El canal de comunicación de agentes (ACC)
- El transporte interno de mensajes de la plataforma (IPMT)

Actualmente no cuenta con un mecanismo formal o claro para determinar los cumplimientos de la implementación de la arquitectura de FIPA. Además hay diversas versiones de las arquitecturas disponibles, que no son compatibles totalmente.

Además de los componentes obligatorios del modelo de referencia FIPA, la distribución FIPA-OS incluye el soporte para:

- Diferentes tipos de Agentes Shells para producir agentes los cuales pueden comunicarse con otros facilitadores usando FIPA-OS.
- Utiliza múltiples capas para soportar la comunicación de agentes.
- Administración de conversaciones y mensajes.
- La configuración dinámica de la plataforma soporta múltiples IPTMs, múltiples tipos de persistencias y múltiples codificaciones.
- Una interfaz abstracta y un modelo de diseño de software.
- Herramientas de diagnóstico y visualización.

La arquitectura de FIPA-OS se puede considerar como modelo no estricto de capas. El desarrollador es capaz de ampliar la arquitectura no sólamente añadiendo capas de valor añadido tales como agentes de servicio especialistas o agentes facilitatores en capas superiores, además, en capas medias o inferiores pueden ser substituidas, modificadas o suprimidas.

Ventajas

Una de las principales ventajas del uso de los estándares de la FIPA es que reduce considerablemente la cantidad de trabajo dedicado a garantizar una interoperatividad de las aplicaciones, también es una herramienta abierta, lo cual permite hacer modificaciones para adaptarla al dominio deseado si es requerido realizar unos ajustes, es FIPA-compatible, lo cual permite interactuar con otras plataformas que también sean FIPA-compatibles, facilita los desarrollos rápidos de agentes FIPA-compatibles, puede ser implementado en diversos sistemas operativos.

Desventajas

No cuenta con una interfaz gráfica para la construcción de los agentes y las diversas versiones de esta plataforma no son totalmente compatibles entre ellas.

Requerimientos del sistema.

Versión probada principalmente con Windows 2000 y también ha tenido un desempeño satisfactorio en ambientes Linux y Solaris. El hardware recomendado como mínimo para una PC es:

Procesador Pentium 166Mhz, 60 MB de memoria y 35 MB de espacio en disco, así como JDK 1.1 o superior.

2.4.2 Herramientas para desarrollar MAS especializados en la movilidad

Las herramientas diseñadas para desarrollar MAS especializados en la movilidad cuentan con mecanismos y con modelos que permiten la elaboración de agentes móviles más fácil de implementar, ya que reducen los problemas de migración de los agentes a simples instrucciones. De las herramientas encontradas se tienen las siguientes que se describen en las secciones siguientes.

2.4.2.1 Aglets

Los aglets son agentes autónomos basados en Java, desarrollados por IBM. Proveen las capacidades básicas requeridas para la movilidad [30]. Un aglet refleja el modelo de applet en Java pero brindándole la propiedad de movilidad. Es un agente móvil ya que soporta las ideas de ejecuciones autónomas y ruteo dinámico sobre sus itinerarios (ruta que el aglet seguirá cuando es instanciado).

La estructura de Aglets consiste en dos partes, el núcleo del aglet y el proxy. El núcleo es el corazón del aglet y contiene todas las variables internas del aglet y los métodos que proporcionan interfaces a través de las cuales el aglet puede comunicarse con su medio ambiente. El núcleo del aglet es encapsulado por el proxy que actúa como un protector contra el acceso directo a cualquier método y variables privadas, y puede ocultar la posición real del aglet de aglets dañinos.

Aglet tiene las siguientes características:

- Un esquema global único para agentes (Modelo de navegación/seguridad).
- Un itinerario de viaje, para la especificación de patrones complejos de viajes con múltiples destinos y manejos de fallas automáticos (Modelo de navegación).
- Un mecanismo *white-board* (pizarra de trabajo) permitiendo que múltiples agentes colaboren y compartan información asíncronamente (Modelo de comunicación).
- Un esquema de transmisión de mensajes que soporta una unión asíncrona desahogada tan bien como una comunicación síncrona entre agentes (Modelo de comunicación).
- Una carga de clases dinámicamente que permite que el código Java de los agentes y la información de su estado viajen a través de la red (Modelo de navegación).
- Un contexto de ejecución que proporciona un ambiente independiente del sistema actual sobre el cual se están ejecutando (Modelo computacional).

El Aglets Workbench (AWB) es el conjunto de clases y herramientas necesarias para la construcción de agentes móviles. El AWB está integrado por:

- Aglet FrameWork.- Es el componente principal y provee un mecanismo de seguridad extensible.
- Protocolo de Transferencia de Agentes.- Es un protocolo que hace posible la transferencia de agentes. Está basado en los URL's (Universal Resource Locator), se enfoca principalmente a Internet y ofrece servicios independientes de la plataforma y de lenguaje.
- Tazza.- Es el compilador con un ambiente de desarrollo visual para aglets.
- Tahiti.- Es el administrador visual de agentes que trabaja conjuntamente con el Aglet FrameWork. Permite monitorear y controlar la ejecución de aglets. Es el componente esencial para poder utilizar los aglets, ya que abre un puerto de comunicaciones, como el que utilizan los demonios http, esto es necesario para que el ATP (Agents Transport Protocol) pueda conectar otros sitios que funcionen con este protocolo.
- Fiji.- Es capaz de crear applets los cuales corren en contextos de aglets y puede crear, despachar o retractar aglets desde páginas Web.

Los aglets utilizan el Protocolo de Transferencia de Agentes (ATP) que es un protocolo a nivel de aplicación para sistemas distribuidos basados en agentes, puede ser usado para transferir agentes móviles entre redes de computadoras, y es independiente de la plataforma para transferir agentes entre redes de computadoras.

Ventajas

Permiten enviar mensajes a otros agentes sin que exista la rigurosa necesidad de tener que conocerlos previamente, permite que actúen autónomamente colaborando con otros agentes. Al contar con similitudes con los applets hace que el aprendizaje de la plataforma sea rápido para quienes trabajan con applets. Está compuesto por una serie de elementos que facilitan la implementación y lo vuelve amigable al usuario. Los mecanismos de seguridad son fácilmente implementados por el usuario. Diseña una arquitectura armoniosa con la existencia de tecnología Web/JAVA. Al igual que algunas otras herramientas permite que la herramienta pueda ser distribuida en diferentes sistemas operativos.

Desventajas

No permite migrar un agente con su estado (registros), los mecanismos de seguridad son limitados en la práctica ya que no existe una estructura en los sitios que proteja de los Aglets no seguros, así como permitir el acceso a los Aglets seguros.

Requerimientos del sistema.

Se solicita contar con el JDK 1.1 o posterior con sistema operativo SPARC/Solaris 2.5, x86/Solaris(1.1 beta 4), Windows 95/NT, AIX 4.1.4 o Macintosh (MRJ SDK 2.0.1).

2.4.2.2 D'Agents

Agent Tcl, ahora llamado D'Agents [27] es una plataforma simple independiente del sistema de agente móvil, diseñada para que sea independiente de la máquina virtual y su respectivo lenguaje. Actualmente, además de Tcl (*Tool Command Language*) soporta Java y C/C++.

D'Agents posee dos componentes principales:

- El intérprete Tcl consiste de una máquina virtual Tcl extendida con los siguientes cuatro componentes:

  1. Un módulo para capturar el estado del agente que provee checkpoint/restart transparentes en las operaciones de migración.
  2. Un modelo de seguridad, para inhibir a los agentes de la ejecución de operaciones impropias.
  3. Una interfaz (API) para que el servidor de agentes implemente, por ejemplo, primitivas de comunicación y migración.
  4. El intérprete Tcl encargado de ejecutar los agentes creados.

- El servidor de agentes consiste de dos procesos cooperativos:

  1. Un proceso de sockets que es responsable de la creación, manejo y transporte de agentes.
  2. Un proceso asíncrono (*tabla de agentes*) que es responsable del manejo de identidades y la comunicación entre agentes. Cada agente es un proceso autónomo, creado por el servidor de agentes, que más allá de su código específico, tiene el código del intérprete respectivo.

El modelo de navegación está basado en un simple comando *agente_salta,* este comando puede aparecer en un agente y provocar que su estado y contexto de ejecución sea congelado y transportado a un nodo específico; esta habilidad es más sofisticada que la de los Aglets. El Modelo de Comunicación tiene tres comandos: *agente_enviar, agente_recibir* y *agente_reunir.*

D'Agents tiene importantes características, como:

- Arquitectura Simple.
- Seguridad.
- Transparencia en la movilidad (TCP/IP).

El Lenguaje D'Agent es una extensión de Tcl/Tk que soporta la programación distribuida en la forma de agentes transportables. Tcl es realmente dos cosas: un lenguaje script y un intérprete para este lenguaje, que es diseñado para ser fácilmente incorporado dentro de las aplicaciones.

Ventajas

Emplea un lenguaje de codificación principal (Tcl) simple, el sistema de seguridad es aceptable garantizando una confiable utilización de los recursos protegiendo contra ataques de agentes maliciosos, permite la implementación de los agentes tanto en lenguajes de programación como JAVA o C++, lo cual permite a los programadores emplear las características de lenguajes, también incluye un servidor de alto desempeño para procesos multi-hilo.

Desventajas

Es una herramienta que limita drásticamente los sistemas sobre los cuales permite operar ya que sólo trabaja en equipos UNIX. No cuenta con un entorno visual de depuración, el uso de intérprete hace que sea lento en comparación con otros lenguajes de codificación.

Requerimientos del sistema.

D'Agent es una herramienta destinada a UNIX y puede ser compilada en las siguientes plataformas, Linux 2.0, 3.0, FreeBsd 2.1, AIX 3.2.5, IRIS 6.2, Digital OSF1 V4.0 y Solaris 2.5.1.

2.4.2.3 Concordia

Concordia es una herramienta de trabajo desarrollado por Mitsubishi Electronic para desarrollar y manejar aplicaciones de agentes móviles eficientemente y para accesar información en cualquier tiempo, en cualquier lugar y sobre cualquier dispositivo que soporte Java [30].

Las aplicaciones:

- Procesan datos sobre los datos fuente.
- Procesan datos aún cuando el usuario está desconectado de la red.
- Accesan y entregan la información a través de múltiples redes.
- Utilizan comunicación inalámbrica.
- Soportan múltiples dispositivos clientes, tales como computadoras de escritorio, portátiles, PDAs y teléfonos inteligentes.

Concordia oculta las complejidades de programar una aplicación móvil a los programadores, desarrollando una aplicación agente-habilitado similar a un programa estacionario o no móvil. Los agentes mantienen su estado interno mientras viajan en la red, así que ellos pueden reanudar su ejecución al llegar a una nueva posición.

Un agente Concordia viaja en la red definido por su *Itinerario*. El Itinerario especifica a dónde viajará el agente y qué tareas deberá desarrollar cuando

llegue. Los itinerarios de Concordia son especificados en tiempo de ejecución, los agentes pueden cambiar su propio itinerario basados en la información y eventos descubiertos a medida que los agentes viajan.

Servidor Concordia.- El servidor de Concordia es el nombre del componente completo de Concordia instalado y que funciona en una máquina en la red de Concordia. Está formado por componentes de administración como:

- Administrador de agentes.- Provee una infraestructura para la comunicación que permite a los agentes transmitir y recibir por medio de nodos de red. Administra el ciclo de vida del agente. Provee un mecanismo para crear agentes y destruirlos, además de brindar la ejecución de un agente.
- Administrador.- La administración de la red Concordia esta provista por el manejador de administración, en cooperación con los servicios de Concordia corriendo sobre los diversos nodos bajo administración. El manejador de Administración maneja todos los servicios provistos por Concordia incluyendo el Administrador de Agentes, Administrador de Seguridad, Administrador de eventos, etc..
- Administrador de seguridad.- Responsable de identificar a los usuarios, autenticar sus agentes, proteger los recursos del servidor y garantizar la seguridad e integridad de los agentes.
- Administrador de persistencia.- Mantiene el estado del agentes en transmisión alrededor de la red. Permite el chequeo y reinicio de los agentes ante un evento en el cual falle el sistema.
- Administrador de eventos.- Manipula el registro de anuncios y notificaciones de eventos de los agentes. Puede pasar la notificación de eventos a los agentes en algún nodo de la red de Concordia. Una importante función de este componente es el soporte de colaboración de agentes.
- Administrador de Cola.- Es responsable de la programación y de revisar el movimiento de agentes entre sistemas Concordia. Este administrador provee el mecanismo para prioritización y manejo de la ejecución de agentes en la entrada para nodos Concordia.
- Administrador de directorio.- Provee el servicio de nombres en la red de Concordia.
- Servicio Bridge.- provee la interfaz de agentes Concordia para los servicios disponibles a varias máquinas en la red Concordia.
- Librería de agentes.- provee todas las clases necesarias para desarrollar agentes móviles Concordia.

Concordia proporciona una interfaz para el desarrollo y también para el control de los agentes en tiempo de ejecución, el lenguaje de comunicación que emplea está basada en protocolos estándar de Java y la comunicación existente se realiza a través de TCP/IP.

Ventajas

Prácticamente se puede emplear en cualquier tipo de dispositivo de clientes múltiples (PC, PDA, teléfono), la movilidad y el acceso a datos eficiente es una de sus características. Permite que las aplicaciones accedan a información distribuida en cualquier momento, también procesa información sin la necesidad de encontrarse el usuario conectado a la red y el acceso a los datos es eficiente. Ofrece seguridad completa, ya que protege al agente contra agentes maliciosos, el acceso a recursos del sistema servidor por los agentes autorizados, así como protección durante la transmisión.

Desventajas

Los agentes tienen reflejado en su estado un itinerario el cual es creado por el usuario y envía al agente indicando eventualmente su itinerario por tanto los agentes no resultan ser tan autónomos.

Requerimientos del sistema.

Se ejecuta en cualquier plataforma que cuente con JDK (Windows 95/98, Windows NT, Solaris, etc), Para equipos Windows 95/98 se requiere como mínimo una PC a 100 Mhz con 12Mb en RAM para desarrollo y 8Mb mínimo para ejecución. Con Windows NT demanda un PC 486 a 100 Mhz con 16Mb mínimo de RAM para desarrollo y 12Mb para ejecución.

2.4.3 Comparación de las herramientas para la construcción de agentes

Después de presentar un panorama de las herramientas más destacadas en el mercado, se presentan en las tablas siguientes (tabla 2.1 y 2.2), una comparación que muestra las características más importantes de cada una de ellas, para poder así comprobar y comparar la utilidad de las mismas de una manera más sencilla y ágil. Los aspectos que están en comparación son la ***organización de investigación*** que la desarrolla, el ***lenguaje*** de programación utilizado para el desarrollo de los agentes, el ***lenguaje de comunicación*** empleado para las interacciones entre agentes, sí emplean una interfaz gráfica de usuario (***GUI***) ***para desarrollo*** de los sistemas, donde es posible llevar a cabo la ***creación de agentes*** ya sea local o si permite crearlo remotamente, si la herramienta soporta la ***movilidad*** de los agentes dentro de sus características, si proporciona un medio para el conocimiento de los servicios que proporciona cada agente empleando un ***servicio de directorios***, si proporciona un ***GUI para test y control de ejecución*** de los agentes para conocer sus estados internos y acciones realizadas, que ***protocolos de comunicación*** son empleados para llevar a cabo la comunicación, el tipo de ***distribución*** que hace referencia a que es un producto libre o comercial y una ***descripción*** breve de la herramienta.

Producto / Caract	AgentBuilder	JATLite	JADE	Zeus
Organización de investigación	Reticular Systems, Inc.	Universidad de Stanford	Universidad de Parma	Laboratorios de British Telecomunicaciones
Lenguaje	Java	Java	Java	Java
Lenguaje de comunicación	KQML	Socket TCP/IP KQML	ACL-FIPA	KQML, ACL
GUI para desarrollo	Si	Si	No	Si
Creación de agentes	Local y Remota	Local	Local y Remota	Local y Remota
Movilidad	No	Si	Si	No
Servicio de directorios	Si	Si	Si	Si
GUI para test y control de ejecución	Si	Si	Si	Si
Sistema Operativo	Todos	Todos	Todos	Windows y Solaris
Distribución	Comercial	Freeware	Freeware	Freeware
Protocolos de comunicación	TCP/IP	TCP/IP, FTP, SMTP	TCP/IP	TCP/IP
Descripción	Ambiente de desarrollo de agencia y agente integrado	Paquete de Java para Multi-agentes	Framework de Multi-agentes	Ambiente para construcción de agentes

Tabla 2.1 Tabla comparativa de herramientas

Producto / Caract	FIPA-OS	Aglets	D'Agents	Concordia
Organización de investigación	Laboratorios de Nortel Networks	IBM Japón	Universidad Dartmouth	Mitsubishi Electric, Centro de información tecnológica americana
Lenguaje	Java	Java	Tcl / C++ y Java	Java
Lenguaje de comunicación	ACL-FIPA	Directivas	Tcl	Directivas
GUI para desarrollo	No	Si	Si	Si
Creación de agentes	Local y Remota	-	Local y Remota	Local
Movilidad	Si	Si	Si	Si
Servicio de directorios	Si		Si	Si
GUI para test y control de ejecución	Si	Si	Si	Si
Sistema Operativo	Todos	Todos	Unix	Todos
Distribución	Freeware	Comercial	Freeware	Comercial
Protocolos de comunicación	TCP/IP	-	TCP/IP	TCP/IP
Descripción	Componentes basados en un juego de herramientas	Agentes Móviles	Recuperación, organización y presentación	Agentes Móviles

Tabla 2.2 Tabla comparativa de herramientas

2.4.4 Selección de la herramienta para la programación de agentes

Del grupo de herramientas que se han descrito en la sección 2.4, se ha seleccionado una para llevar a cabo el desarrollo del prototipo que pretende solucionar las problemáticas detectadas (sección: *Descripción de la problemática encontrada*) en el área de Soporte Informático de la Gerencia de Tecnología Informática. Los criterios para la selección de la herramienta son los siguientes:

- Tipo de distribución de la herramienta, esto es, si es un producto comercial o freeware, y según este criterio se descartan para realizar el desarrollo las herramientas de carácter comercial, al no estar disponibles en el marco de la tesis. Por lo tanto se descartan las herramientas AgentBuilder, Concordia y Aglets.
- La herramienta debe permitir la ejecución en equipos que cuenten con sistema operativo Windows, por lo cual, se tiene que descartar D'Agents ya que esta herramienta sólo puede ejecutarse en equipos que cuenten con sistemas Unix.
- La herramienta tiene que permitir la ejecución remota de los agentes, por lo cual ahora se descarta JATLite ya que sólo permite la creación de agentes localmente, y como el sistema está diseñado para que los agentes además de ejecutarse localmente, se ejecuten en algunas situaciones especiales, de manera remota, por tanto ha sido desechada esta herramienta.
- La Herramienta debe basarse en estándares internacionales dictados por FIPA, por lo cual se tiene que descartar ZEUS ya que es una herramienta que no se basa en estos estándares. Las herramientas FIPA-OS y JADE son las que tratan de cumplir totalmente con los estándares marcados por FIPA.
- Herramienta que más se utiliza en el desarrollo de aplicaciones comerciales, por lo cual en el desarrollo del prototipo de esta tesis me inclino por la elección de la plataforma para construcción de agentes JADE ya que de entre las dos herramientas se ha utilizado en un mayor número de sistemas de agentes, no sólo para proyectos académicos, sino también en proyectos comerciales,
- Compatibilidad entre diversas versiones de la herramienta, por ejemplo en el caso de las diferentes versiones de FIPA-OS no son totalmente compatibles entre ellas y esto en un futuro podría causar problemas ya que se tiene en mente la construcción de otros sistemas dentro del Instituto Mexicano del Petróleo y si se desea que los demás proyectos cooperen entre ellos, ésto podría involucrar retrasos en los proyectos al tratar de hacer que las plataformas de agentes sean compatibles. Otra cosa que perjudicaría que las plataformas de agentes no sean totalmente compatibles es la administración remota de las plataformas y por ende de los agentes.

Por los puntos anteriormente citados se cree que la mejor opción para la construcción del prototipo de agentes es emplear la herramienta JADE.

CAPÍTULO 3
Modelo conceptual del Sistema

3.1 Introducción

En este capítulo se muestra el diseño realizado sobre el Sistema Basado en Agentes para la Instalación Remota de Software empleando la metodología MAS-CommonKADS (sección 2.3.2.2) [25]. Inicia con la fase de conceptuar el sistema desde el punto de vista del usuario y posteriormente una serie de modelos que permiten vislumbrar requerimientos y reflejar como interactúa el sistema con sus componentes. Para evitar una extensión excesiva, no se presenta el desarrollo completo, sino los casos que se han considerado más representativos del sistema.

3.2 Conceptuación

La fase de conceptuación consiste en concebir el problema que se va a resolver y elaborar un esbozo del sistema que puede resolverlo. Propone el uso de uno de los enfoques más extendidos de la metodología orientadas a objetos: el análisis centrado en el usuario, que es el proceso de captura de requisitos desde el punto de vista del usuario, cuyo objetivo es comprender los requisitos de los usuarios para construir el sistema que se ajusta a sus necesidades. Los siguientes puntos resumen los requisitos del usuario.

3.2.1 Identificación de los actores

Tomando a un actor como un ente que interpreta un papel demandante externo del sistema y que interactúa con él, se encuentran los siguientes actores:

- Técnico
- Secretaria
- Supervisor
- Usuario

3.2.2 Descripción de los actores

De los actores encontrados en el punto anterior, a continuación se describen empleando plantillas textuales.

Actor *Técnico*
descripción
persona que realiza instalaciones y configuraciones de software y elabora reportes de servicios.

Actor *Secretaria*
descripción
persona que toma reportes de solución de servicios y elabora reportes globales de servicios.

Actor *Supervisor*
descripción
persona que elabora estadísticas de las problemáticas que presentan los usuarios con sus equipos y confirma la solución de los problemas reportados.

Actor *Usuario*
descripción
persona que reporta una problemática en su equipo.

3.2.3 Identificación de los casos de uso

En la siguiente sección se muestran los posibles usos que da cada actor al sistema los cuales se presentan a continuación. Los usos se encuentran separados dependiendo de cómo cada actor los utiliza.

Casos de uso para el actor Técnico.

Crear Características Del Software Nuevo, Editar Las Características Del Software, Eliminar Las Características Del Software, Leer Reportes De Solicitud De Servicios, Leer Reportes De Servicios, Editar Reporte De Servicios, Crear Reporte De Servicios.

Casos de uso para el actor Secretaria.

Crear Reporte De Solicitud De Servicios, Leer Reporte De Solicitud De Servicios, Crear Reporte Global Automático, Leer Reporte Global, Editar Reporte Global, Leer Reporte de Servicios.

Casos de uso para el actor Supervisor.

Leer Reportes Globales, Leer Reportes De Servicios, Crear Estadística De Servicios, Editar Estadística De Servicios, Leer Confirmación Del Servicio, Crear Reporte Solicitud De Servicio, Leer Reporte De Solicitud De Servicio, Confirmar Servicio.

Casos de uso para el actor Usuario.

Crear Reporte De Solicitud De Servicio, Confirmación De Servicio.

3.2.4 Descripción de los casos de uso

Para la descripción de los casos de uso se utilizan dos tipos de notaciones habitualmente: textuales y gráficas.

3.2.4.1 Descripción textual de los casos de uso

Las siguientes plantillas muestran la notación propuesta por Rumbaugh [19], las cuales muestran una descripción más detallada de los casos de uso, cómo interactuarían los actores con ellos, qué actores podrán trabajar con cada uno de los casos, las postcondiciones, etc..

Caso de uso *Crear Características de Software Nuevo*

resumen

El Técnico introduce las características del software en un registro de una base de datos, permitiendo editar posteriormente las características.

actores

Técnico.

precondiciones

Ser usuario.

descripción

El usuario selecciona en el menú la opción *Software* y dentro de éste la opción *Nuevo*, en la vista de edición aparece el esqueleto de la especificación, a continuación el usuario puede editar las características del software, puede guardar o puede cerrar la aplicación.

excepciones

Ninguna.

postcondición

Tras este caso de uso el técnico puede iniciar los casos de uso: Editar las Características del Software, Eliminar las Características del Software.

Caso de uso *Crear Reportes de Solicitud de Servicios*

resumen

El usuario crea un reporte de solicitud de servicios con alguna petición.

actores

Usuario, Secretaria y Supervisor.

precondiciones

Ser usuario.

descripción

El usuario selecciona en el menú la opción *Reportes de Solicitud* y dentro de este la opción *Nuevo*, enseguida aparecerá una plantilla la cual contendrá campos con las opciones necesarias, el usuario puede ejecutar comandos de edición (ingresar texto, seleccionar texto, copiar texto, pegar texto, eliminar texto).

excepciones

Ninguna.

postcondición

Tras el caso de uso el usuario puede iniciar los casos de uso: Leer Reporte de Solicitud de Servicio.

Caso de uso *Crear Reporte de Servicio*

resumen

El usuario crea un reporte de las actividades realizadas en la máquina del usuario.

actores

Técnico.

precondiciones

Ser usuario.

descripción

El usuario selecciona en el menú la opción *Reportes de Servicio* y dentro de éste la opción *Nuevo*, en la vista aparece una pantalla en la cual contiene campos con las opciones necesarias, el usuario puede ejecutar comandos de edición.

excepciones

Ninguna.

postcondición

Tras este caso de uso el usuario puede iniciar los casos de uso: Editar Reportes de Servicio y Leer Reporte de Servicio.

Caso de uso *Crear Reporte Global Automático*

resumen

El usuario elige la opción de generar reporte global y el sistema lo generará automáticamente .

actores

Secretaria.

precondiciones

Ser usuario Autorizado, tener al menos un reporte de servicio.

descripción

El usuario selecciona en el menú la opción *Reporte Global* y dentro de este la opción *Nuevo*, aparece una vista indicando las fechas dentro de las cuales hay que generar uno, al mismo tiempo se puede ejecutar el caso de uso: Leer Reporte Global.

excepciones

Ninguna.

postcondición

Tras este caso de uso se puede ejecutar los casos de uso: Leer Reporte Global y Editar Reporte Global.

Caso de uso *Leer Reporte Global*

resumen

El usuario lee el reporte global y no lo puede modificar.

actores

Secretaria y Supervisor.

precondiciones

Ser usuario, tener al menos un reporte global.

descripción

El usuario selecciona en el menú la opción *Reporte Global* y dentro de este la opción *Abrir*, se despliega una vista con los reportes existentes, se elige uno y es desplegado en una vista al mismo tiempo se puede ejecutar el caso de uso Editar Reporte Global.

excepciones

El caso de uso de editar sólo está disponible para el usuario Secretaria.

postcondición

El reporte no sufre modificaciones.

Caso de uso *Crear Estadísticas de Servicio*

resumen

El usuario crea las estadísticas de servicio.

actores

Supervisor.

precondiciones

Tener reporte global.

descripción

El usuario selecciona en el menú la opción *Estadísticas de Servicio* y dentro de este la opción *Crear*, debe elegirse el reporte global del cual se pretende generar las estadísticas y debe visualizar las estadísticas.

excepciones

Ninguna

postcondición

Tras este caso de uso se puede ejecutar los casos de uso: Editar Estadísticas de Servicio y Visualizar Estadísticas de Servicio.

Caso de uso *Visualizar Estadísticas de Servicio*

resumen

El usuario visualiza las estadísticas de servicio sin poder editarlo.

actores

Supervisor.

precondiciones

Tener al menos una estadística de servicio.

descripción

El usuario selecciona en el menú la opción *Estadística de Servicio* y dentro de éste la opción *Visualizar,* se mostrará una lista para elegir la estadística de servicio, una vez elegida se debe mostrar el documento de servicio. Se puede ejecutar al mismo tiempo el caso de uso Editar Estadísticas de Servicio.

excepciones

Ninguna.

postcondición

Tras este caso de uso no debe de modificarse el documento activo.

Caso de uso *Confirmar Servicio*

resumen

Crear un mensaje de confirmación de servicio.

actores

Supervisor.

precondiciones

Tener un reporte de solicitud de servicio.

descripción

El usuario selecciona en el menú la opción *Confirmar Servicio* y dentro de éste la opción *Nuevo,* se muestra una vista con los datos del usuario y el problema.

excepciones

Ninguna.

postcondición

Después de ejecutar este caso puede ejecutarse el caso de uso Confirmación de Servicio.

Caso de uso *Leer Confirmación de Servicio*

resumen

Lista las respuestas de los usuarios a un servicio solicitado.

actores

Supervisor.

precondiciones

Tener una confirmación de servicio.

descripción

El usuario selecciona la opción mostrada en el cuadro de mensajes que se adecue a su respuesta.

excepciones

Ninguna.

postcondición

Ninguna.

3.2.4.2 Descripción gráfica de los casos de uso

La notación gráfica que sigue la metodología es la propuesta de Regnell [25] para indicar la relación entre un caso de uso y la técnica de descripción formal MSC (Diagrama de Secuencia de Mensajes, Message Sequense Chart), pero para esta tesis se adoptó el uso del lenguaje Agent-UML (Ver Anexo A), ya que describe de mejor forma el comportamiento de los agentes.

Para ilustrar este capítulo solo se han colocado los diagramas más importantes del sistema. Los demás han sido colocados en el Anexo A.

La figura 3.1 muestra el caso de uso del usuario en tanto que la figura 3.2 el caso de uso del técnico. Los casos de usos se encuentran descritos en la sección 3.2.4.1.

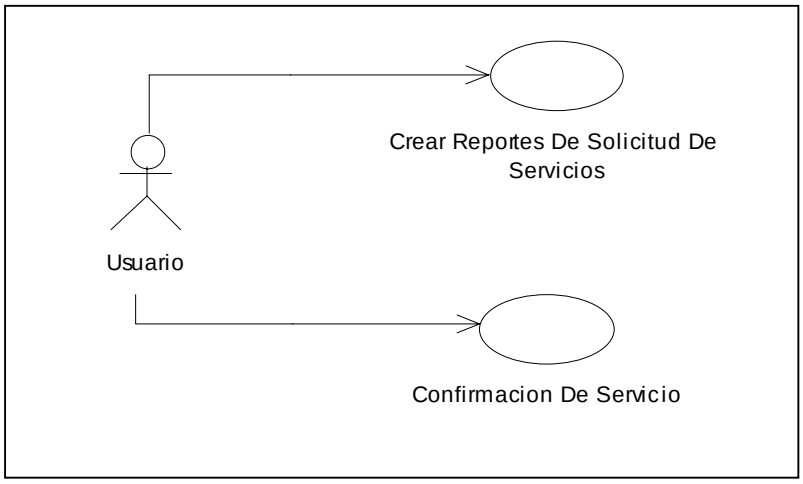

Fig. 3.1 Casos de uso del usuario

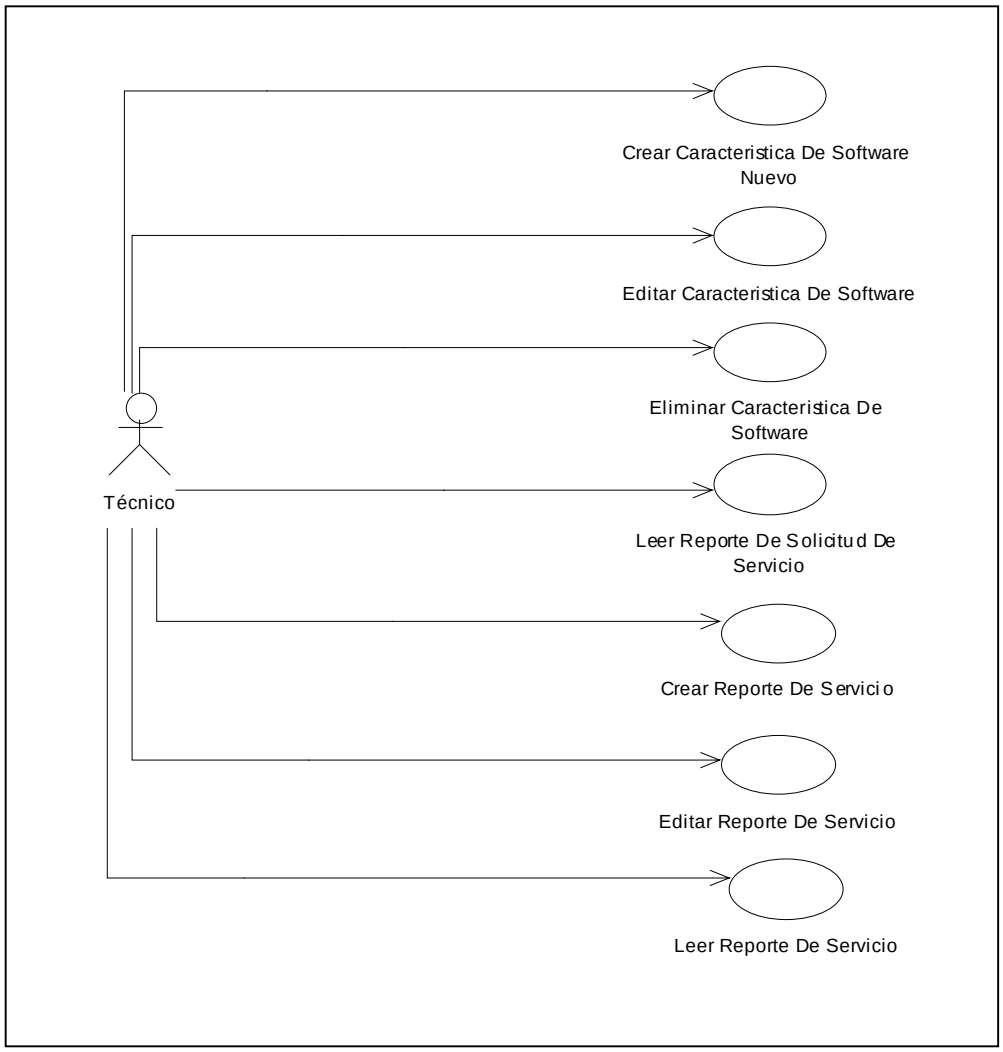

Fig.3.2 Casos de uso del Técnico

3.2.5 Descripción de las interacciones de los casos de uso

En este punto de la metodología se describe las interacciones de los casos de uso empleando gráficas MSC's, pero se han tomado los diagramas de secuencias del A-UML ya que, en base a un estudio previo es un lenguaje de modelado que está diseñado específicamente para modelar el comportamiento de los agentes y no así el MSC que es para modelar objetos, además de que A-UML es más fácilmente comprensible para usuarios que no están familiarizados con MSC, sobre todo en relación con la forma en que interactúa la secuencia de mensajes.

Por medio de los siguientes diagramas se muestra cómo se realiza la interacción de los casos de uso con los actores y con el sistema. La figura 3.3 ilustra la secuencia de mensajes que existen para agregar un nuevo software a la base de datos.

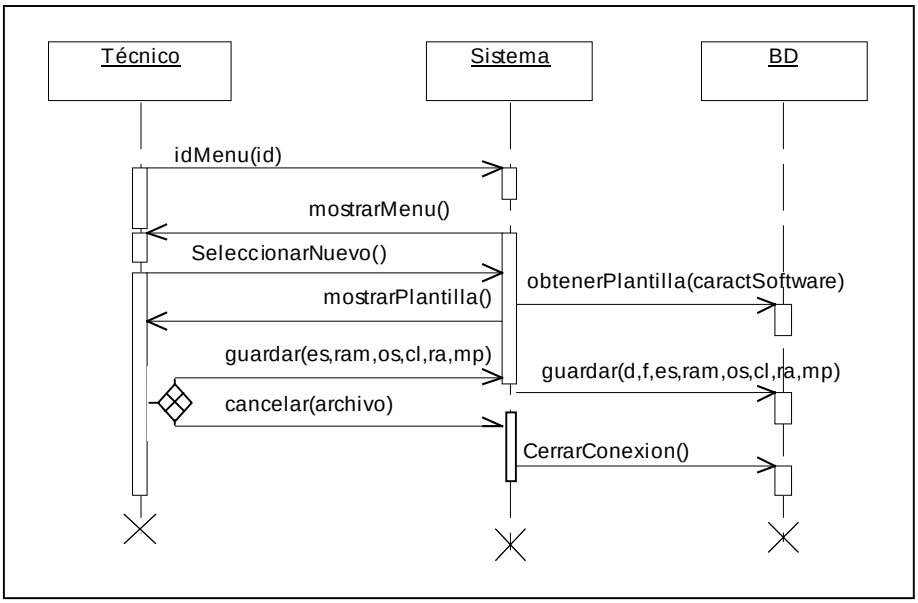

Fig. 3.3 Diagrama de secuencia
"Crear características de software"

La figura 3.4 muestra la secuencia de mensajes que existen para la creación de un reporte de solicitud de servicio por parte de un usuario en el momento en que se le presente un problema del software.

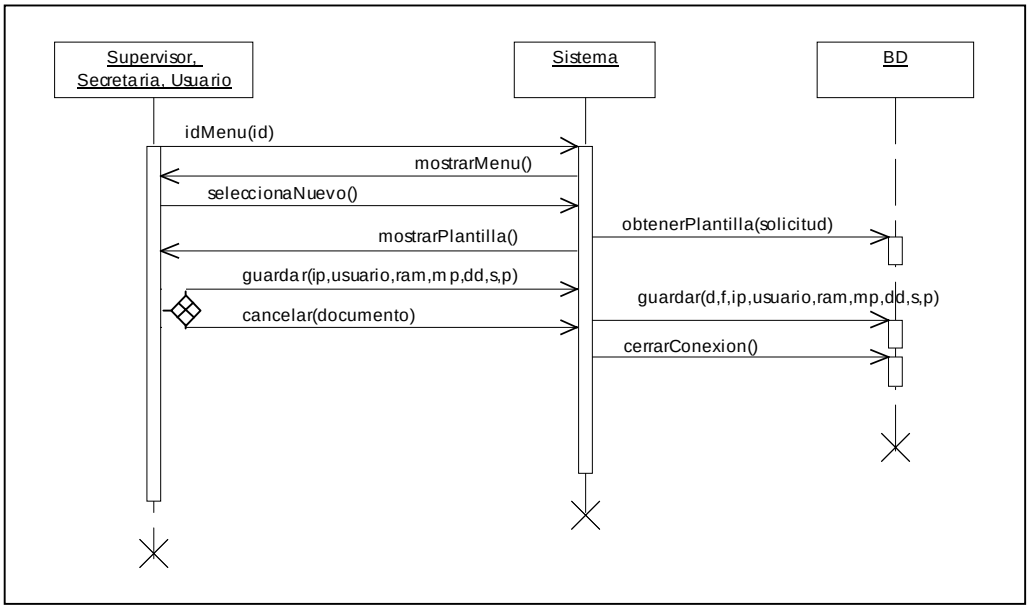

Fig. 3.4 Diagrama de secuencia
"Crear Reporte de Solicitud de Servicio"

La figura 3.5 describe el paso de mensajes entre el técnico y el sistema cuando éste desea crear un reporte de servicio manualmente.

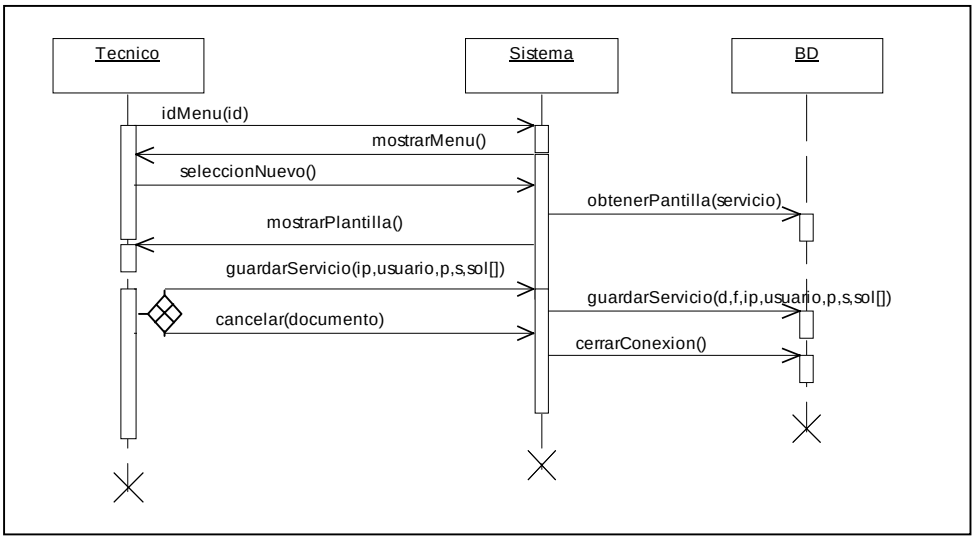

Fig.3.5 Diagrama de secuencia
"Crear Reporte de Servicio"

La figura 3.6 muestra la secuencia de mensajes que existen cuando la secretaria pide que se cree un reporte global de los servicios que se han realizado en un lapso de tiempo.

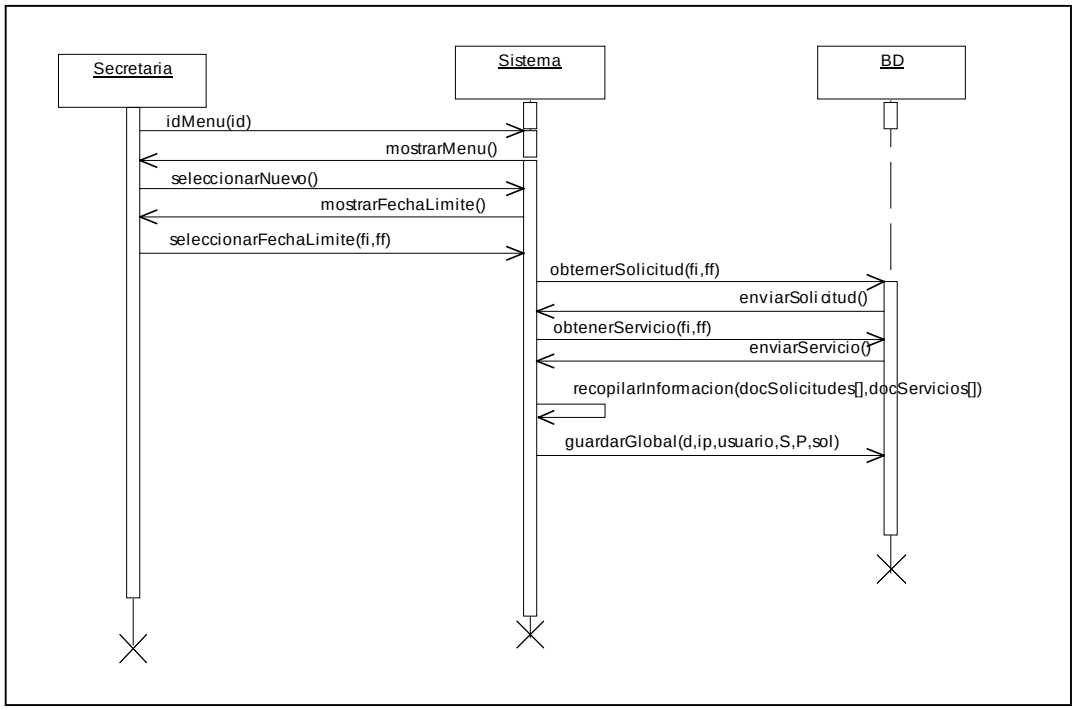

Fig. 3.6 Diagrama de secuencia
"Crear Reporte Global Automático"

3.3 Modelo de Tareas

El modelo de tareas permite describir las actividades relacionadas para alcanzar un objetivo. El objetivo del modelo de tareas es documentar la situación actual y futura del sistema, facilitar la administración de los cambios, y ayudar a estudiar el alcance y viabilidad del sistema inteligente, siendo más específico, este modelo describe las tareas que los agentes pueden realizar: los objetivos de cada tarea, su descomposición, los ingredientes y los métodos de solución de problemas para resolver cada objetivo.

3.3.1 Plantillas de Tareas y Capacidades

Partiendo de la idea de que una tarea es un conjunto de actividades que se realizan para conseguir un objetivo en un dominio dado, a continuación se presenta el detalle de cada tarea identificada en la descomposición del problema identificado (instalación de software vía remota).

Tarea *ObtenerDatosDelEquipo*

objetivo

Recopilar las características de la computadora del usuario a fin de instalar la versión de software apropiada.

descripción

En esta tarea se obtienen las características de la computadora (ip, nombrePC, OS, ram, DD, etc.) por medio del acceso a los archivos que contienen dichas características.

entrada

Archivos de contenido de características de PC.

salida

Documento de características de PC.

precondición

Contar con los archivos del sistema operativo que posee las características de la computadora.

frecuencia

Por cada solicitud de servicio.

Tarea *ObtenerDatosDelUsuario*

objetivo

Capturar los datos del usuario por medio de una interfaz gráfica.

descripción

en esta tarea se presenta una interfaz gráfica la primera vez que inicia el agente en la máquina pidiendo ingresar los datos del usuario (nombre, claveEmpleado, extensión, etc.), o cuando el documento que contiene estos datos no se encuentre o está dañado. La información obtenida se almacenará en un documento específico sobre datos del usuario.

entrada

Datos del usuario.

salida

Documento con datos del usuario.

precondición

Ser usuario del equipo utilizado.

frecuencia

Ser la primera vez que se levanta el agente o bien que el documento que contiene los datos presente daños o se pretenda modificar los datos.

Tarea *ObtenerArchivosDeConfiguraciónDelEquipo*

objetivo

Obtener los archivos del registro de Windows.

descripción

En esta tarea se obtienen los archivos del registro de Windows que contiene la información sobre la configuración de los programas que

puede manipular el sistema de agentes, una vez obtenidos se realizará el respaldo de los mismos.

entrada

Archivos de configuración.

salida

Documento de información de la configuración de programas instalados.

precondición

Tener funcionando el OS.

frecuencia

Semanalmente o bien cuando se realice una solicitud de servicio.

Tarea *ElaborarCadenaSolicitudDeServicio*

objetivo

Conformar una cadena de petición de servicio.

descripción

Crear una cadena que realice una petición de un servicio por medio de un interfaz con el usuario.

entrada

Datos proporcionados por el usuario.

salida

Cadena de solicitud de servicio.

precondición

Tener alguna petición de servicio de instalación, configuración de software.

frecuencia

De acuerdo a las necesidades del usuario.

Tarea *CoordinarAcciónDeAgentes*

objetivo

Activar el o los agentes necesarios para realizar un servicio.

descripción

De acuerdo al servicio que se solicite por parte del usuario se activan los agentes necesarios para cumplir con el servicio solicitado, pasándole los datos necesarios a él o los agentes que realicen el servicio.

entrada

Cadena de Solicitud de Servicio.

salida

Activación de un agente.

precondición

Tener una solicitud de servicio.

frecuencia

Cada ocasión que se necesite un servicio disponible.

Tarea *ElaborarCadenaDeServicio*

objetivo

Conformar una cadena, la cual contenga información de los actos realizados por parte del especialista en el ordenador del usuario.

descripción

Crear una cadena que contenga información de las actividades realizadas por parte del técnico en la máquina del usuario tras una solicitud de servicio.

entrada

Datos del usuario, del PC y actividades realizadas en el PC.

salida

Cadena del reporte de servicio.

precondición

Tener una solicitud de servicio.

frecuencia

Cada vez que se realice o se proporcione un servicio al usuario.

Tarea *GuardarArchivoDeReporte*

objetivo

Almacenar un documento.

descripción

Almacenar un reporte en una carpeta que corresponda al tipo de reporte del cual se trate y de la fecha de creación del mismo.

entrada

Reporte.

salida

Documento.

precondición

Tener un reporte elaborado.

frecuencia

Cada ocasión que llegue una cadena de un reporte.

Tarea *ObtenerInformaciónDelReporte*

objetivo

Obtener información de un reporte.

descripción

Extraer la información vital de un reporte para realizar actividades de otras tareas.

entrada

Documento (reporte).

salida

Una tarea precedida a la obtención del reporte solicitado.

precondición

Tener que dar seguimiento a un reporte.

frecuencia

Cada ocasión que se pretenda procesar la información de un reporte.

Tarea *InstalarSoftware*

objetivo

Instalar un paquete de software solicitado por el usuario.

descripción

Ejecutar la instalación de software desde el ordenador en el cual se realiza la solicitud de instalación hacia la máquina que se desempeña como administradora del software.

entrada

Solicitud de instalación.

salida

Instalación de software.

precondición

Una solicitud de instalación, que sea el software que se encuentre en la lista de disponibles.

frecuencia

Cada vez que se necesite instalar un paquete de software.

Tarea *IntroducirParámetrosDeInstalación*

objetivo

Proporcionar los datos en una instalación de software.

descripción

Proporcionar los datos que se piden en el momento de realizar la instalación de software, obteniéndolos de una base de datos.

entrada

Parámetros de instalación.

salida

Introducción de parámetros de instalación.

precondición

Ejecutar la instalación de software.

frecuencia

Cada vez que se encuentre en proceso una instalación de software.

Tarea *ObtenerDatosDeDB*

objetivo

Obtener datos de la base de datos.

descripción

Realizar la conexión a la base de datos para obtener datos específicos y cerrar la conexión una vez obtenidos los datos.

entrada

Datos a obtener.

salida

Datos obtenidos.

precondición

Necesitar la obtención por parte de parámetros para la instalación de software o bien para configurar la instalación de software.

frecuencia
Cada ocasión que se necesiten parámetros de la base de datos.

Tarea *IngresarDatosADB*
objetivo
Ingresar datos en la base de datos.
descripción
Realizar la conexión a la base de datos para ingresar datos específicos en los campos correspondientes y cerrar la conexión una vez ingresados los datos.
entrada
Campos a ingresar.
salida
Ingreso de datos.
precondición
Solicitar dar de alta una nueva aplicación en el sistema.
frecuencia
Cada ocasión que se pretenda dar de alta una aplicación en el sistema.

Tarea *EliminarDatoDeDB*
objetivo
Eliminar datos de la base de datos.
descripción
Realizar la conexión a la base de datos para eliminar datos específicos de un paquete de software y cerrar la conexión una vez eliminados los datos.
entrada
Campos a eliminar.
salida
Eliminación de datos.
precondición
Dar de baja una aplicación de software o paquete.
frecuencia
Cuando el paquete sea caduco.

Tarea *Configurar*
objetivo
Realizar la configuración del software.
descripción
Introducir parámetros del software que requiere especificaciones de red como son el I.E. y Outlook.
entrada
Parámetros.
salida
Ninguna.

precondición
Solicitar la personalización de los programas Outlook e I.E.
frecuencia
Después de la instalación de software que requiera configuración o bien cuando se requiera una re-configuración.

Tarea *EjecutarAplicación*
objetivo
Ejecutar una aplicación.
descripción
Ejecutar una aplicación que contenga el ordenador.
entrada
Nombre de la aplicación.
salida
Cadena de actividades realizadas.
precondición
Presentar problemas con el software instalado en el equipo del usuario.
frecuencia
Cuado el software presente problemas.

Tarea *ElaborarEstadísticas*
objetivo
Elaborar las estadísticas de los servicios prestados en un periodo establecido.
descripción
Crear una juego de gráficas que proporcione las estadísticas de los servicios prestados ya sea por usuarios con más problemas, tipo de servicio más solicitado, que software causa más conflictos, etc.
entrada
Reporte global.
salida
Gráficas.
precondición
Tener el reporte global que proporcionará la información necesaria para crear las gráficas.
frecuencia
Cada ocasión que sea solicitada por el usuario.

Tarea *CrearAccesosDirectos*
objetivo
Restaurar los accesos directo de aplicaciones en el escritorio.
descripción
Buscar la aplicación a la cual desea acceder el usuario y crear su acceso directo en el escritorio.
entrada
Nombre de la aplicación.

salida
Acceso directo en el escritorio.
precondición
Encontrar la aplicación.
frecuencia
Cuando el usuario borre por accidente el acceso a una aplicación determinada del escritorio de Windows y sea reportada al sistema.

Tarea *RestaurarArchivosDeConfiguracionDelEquipo*
objetivo
Remplazar los archivos de configuración del equipo.
descripción
Tomar los últimos archivos de configuración sin problemas del equipo en conflicto y reemplazar los archivos de configuración que presentan conflictos.
entrada
Archivos de configuración de respaldo.
salida
Archivos de configuración restaurados.
precondición
Tener respaldos de los archivos de configuración del sistema.
frecuencia
En ocasiones cuando el equipo presente problemas con alguna aplicación y sea reportado al sistema.

Tarea *EvaluarDatosPC*
objetivo
Realizar comparación entre los datos de un equipo con los datos que requiere el software.
descripción
Obtener de la BD los requerimientos del software que desea instalarse en un equipo y obtener la información del equipo y discernir si el equipo cumple con las recomendaciones mínimas de funcionamiento.
entrada
Datos del equipo y requerimientos del software.
salida
Respuesta aprobatoria o no aprobatoria del software.
precondición
Realizar una solicitud de instalación de software, tener el software registrado en el sistema, obtención de la información del equipo en el cual pretende instalarse.
frecuencia
Después de una solicitud de instalar un software.

3.4 Modelo de Agente

El modelo de agente sirve como enlace entre los demás modelos del análisis. El propósito del modelo de agentes es describir los agentes que participan (especificando sus capacidades de razonamiento, habilidades, servicios, sensores, efectores, grupos de agentes a los que pertenece y clase de agente) en la resolución del problema y la repercusión del sistema en los agentes humanos. Este modelo está orientado a servicios.

3.4.1 Identificación de los agentes

Tras el análisis de tareas realizado se puede identificar algunos agentes del sistema multi-agente.

Siguiendo el criterio de asignar agentes de interfaz a los agentes humanos del sistema, definimos un agente *Asistente* para el usuario, un agente *Secretario* para las labores de la
secretaria, un agente *Supervisor* para el supervisor y un agente *Administrador* para el técnico.

Si atendemos a la distribución del problema, definimos un agente *Instalador*, un agente *Configurador*, un agente *Ejecutor* y uno *Restaurador*. A continuación presentamos las plantillas de los agentes identificados en el dominio de aplicación.

3.4.2 Descripción de los agentes

Una vez identificados los agentes que empleará el sistema es el turno de describirlos, como por ejemplo el tipo de agente del que se tiene en mente, el papel que desempeñará dentro del sistema, etc., por tanto se muestra enseguida las plantillas que muestran la descripción de los agentes.

Agente *Asistente*

tipo
Agente software estacionario.

papel
Este agente desempeña el papel de suministrar una solicitud de servicio a otro agente y activar al agente restaurador.

descripción
Este agente será la interfaz con el usuario que captura los datos de los problemas que pueda presentar el usuario con su máquina, y si se trata de un problema con una aplicación ya instalada previamente iniciará al agente restaurador, para solucionar el problema.

Agente *Secretario*

tipo

Agente software estacionario.

papel

Desempeña el papel de administrador de documentos.

descripción

Este agente será la interfaz con la secretaria y se encarga de elaborar reportes globales así como tener comunicación con otros agentes y permitir administrar los reportes.

Agente *Supervisor*

tipo

Agente software estacionario.

papel

Desempeña el papel de monitor de servicios solicitados.

descripción

Es una interfaz en donde el supervisor permite revisar los reportes, elaborar las estadísticas de servicio, enviar mensajes de confirmación de servicios prestados y visualiza las respuestas de confirmación de servicios.

Agente *Administrador*

tipo

Agente software estacionario.

papel

Desempeña el papel de administrador de software y decide que agente es el que tiene que inicializar para dar un servicio.

descripción

Será el encargado de proporcionar una interfaz que le servirá al técnico para administrar el software que puede manipular el sistema, además de que permite visualizar los reportes tanto de solicitud de servicio como los de servicio que haya realizado el sistema. Servirá como motor de comunicación y coordinación con otros agentes que realizan propiamente las tareas del sistema.

Agente *Instalador*

tipo

Agente software móvil.

papel

Desempeña la acción de instalar software.

descripción

Es el encargado de realizar la instalación del software en un equipo lo cual involucra desde trasladarse a la máquina del solicitante hasta enviar un reporte de las acciones realizadas y los resultados obtenidos.

Agente *Configurador*

tipo
Agente software móvil.

papel
Desempeña la acción de configurar el software del equipo.

descripción
Está encargado de efectuar la configuración del software después de que haya sido instalado en un equipo trasladándose e introduciendo parámetros necesarios para personalizar las aplicaciones.

Agente *Ejecutor*

tipo
Agente software móvil.

papel
Realiza la ejecución de aplicaciones.

descripción
Encargado de ejecutar las aplicaciones que sirven para dar mantenimiento al sistema operativo.

Agente *Restaurador*

tipo
Agente software estacionario.

papel
Realiza la acción de manipular los archivos de registros del software de un equipo.

descripción
Encomendado a efectuar la restauración de los archivos de configuración de aplicaciones instaladas en el sistema operativo.

3.4.3 Distribución tareas-agentes

A continuación se muestra la tabla 3.1 en la cual se denota cómo se encuentran distribuidas las tareas que se han identificado en el punto 3.3.1 con respecto a los agentes que se enlistaron en la sección 3.4.1.

tareas / agentes	ObtenerDatos DelEquipo	ObtenerDatos DelUsuario	ObtenerArchivosDe ConfiguracionDelEquipo	ElaborarCadena DeServicio
Asistente	**X**	**X**		**X**
Secretario				
Supervisor				
Administrador				
Instalador				**X**
Configurador				**X**
Ejecutor				**X**
Restaurador			**X**	**X**

Tabla 3.1 tareas Vs Agentes

tareas / agentes	ObtenerInformación DelReporte	GuardarArchivo DeReporte	ElaborarCadena SolicitudDeServicio	Elaborar Estadisticas
Asistente				
Secretario	**X**	**X**		
Supervisor			**X**	**X**
Administrador				
Instalador				
Configurador				
Ejecutor				
Restaurador				

Tabla 3.1 tareas Vs agentes (continuación)

tareas / agentes	CoordinarAccion DeAgentes	Instalar Software	IntroducirParametros DeInstalacion	EvaluarDatosDePC
Asistente	**X**			
Secretario				
Supervisor				
Administrador	**X**			
Instalador		**X**	**X**	**X**
Configurador				
Ejecutor				
Restaurador				

Tabla 3.1 tareas Vs Agentes (continuación)

tareas / agentes	Configurar	EjecutarAplicación	CrearAccesosDirectos	RestaurarArchivosDe ConfiguracionDelEquipo
Asistente				
Secretario				
Supervisor				
Administrador				
Instalador				
Configurador	**X**			
Ejecutor		**X**		
Restaurador			**X**	**X**

Tabla 3.1 tareas Vs Agentes (continuación)

tareas / agentes	IngresarDatosDeDB	ObtenerDatosDeDB
Asistente		
Secretario		
Supervisor		
Administrador	**X**	**X**
Instalador		**X**
Configurador		**X**
Ejecutor		
Restaurador		**X**

Tabla 3.1 tareas Vs Agentes (continuación)

3.4.4 Identificación de objetivos

Se han identificado los siguientes objetivos a partir de la asignación de tareas a los agentes.

- Objetivos del Agente *Asistente*: Obtener las necesidades del usuario y determinar la activación del agente Restaurador.
- Objetivos del Agente *Secretario*: Administrar los reportes.
- Objetivos del Agente *Supervisor*: Monitorear el desempeño del sistema y diseñar las estadísticas de servicio.
- Objetivos del Agente *Administrador*: Determinar la activación de agentes que realizan los servicios de instalación, configuración y ejecución de aplicaciones y Administrar el conocimiento sobre el software.
- Objetivo del Agente *Instalador*: Instalación de software y trasladarse a la PC.
- Objetivo del *Configurador*: Configurar el software y trasladarse a la PC.
- Objetivo del Agente *Ejecutor*: Activar aplicaciones y trasladarse a la PC.
- Objetivo del *Restaurador*: Recuperar registros.

3.4.5 Definición de los objetivos a través de plantillas de objetivos

A continuación se muestra una serie de plantillas que revelan las responsabilidades asignadas o adoptadas por los agentes. La ejecución de estas responsabilidades pueden realizarse mediante la ejecución de una determinada tarea o mediante un mecanismo de planificación.

Objetivo Obtener las necesidades del usuario
tipo
Objetivo reactivo.
parámetros-entrada
Datos del Usuario y datos del ordenador y problemas que presenta.
parámetros-salida
Tipo de servicio y software sobre el cual se dará el servicio.
condición-activación
Inicio del agente.
condición-finalización
Fin del agente.
lenguaje rep. conocimiento
Lenguaje natural.

Ontología
De servicios.
descripción
Este objetivo proporciona los datos para la entrada al sistema. Captura:
- las necesidades del usuario (problemas).
- datos del equipo.
- datos del usuario.

Objetivo Administrar reportes
tipo
Objetivo reactivo.
parámetros-entrada
Cadenas de lenguaje natural.
parámetros-salida
Archivos de reporte de solicitud de servicio, reporte de servicio y reporte global.
condición-activación
Entrada de parámetros, detección de eventos del usuario.
condición-finalización
Cerrar agente.
lenguaje rep. conocimiento
Lenguaje natural.
Ontología
De servicios.
descripción
Controlar el flujo de los parámetros para la creación, obtención almacenamiento de los reportes.

Objetivo Monitorear el desempeño del sistema
tipo
Objetivo reactivo.
parámetros-entrada
Archivos de reportes de solicitud de servicio.
parámetros-salida
Cadena de lenguaje natural.
condición-activación
Iniciar del agente.
condición-finalización
Cerrar aplicación.
lenguaje rep. conocimiento
Lenguaje natural.
Ontología
De servicios.
descripción
Envío de mensajes para confirmar el buen servicio prestado.

Objetivo Diseñar estadísticas de servicio

tipo
Objetivo reactivo.
parámetros-entrada
Archivos de reportes globales.
parámetros-salida
Archivo de graficación.
condición-activación
Iniciar servicio.
condición-finalización
Cerrar aplicación.
lenguaje rep. conocimiento
Lenguaje natural.
descripción
Elaboración de una gráfica que muestre porcentajes de los servicios más realizados, que usuarios realizan más solicitudes de servicios.

Objetivo Determinar activación de agentes

tipo
Objetivo reactivo.
parámetros-entrada
Cadena de reportes de solicitud del servicio.
parámetros-salida
Nombre del software, datos de activación de agente, datos de máquina.
condición-activación
Llegada de una solicitud.
condición-finalización
Ninguna.
lenguaje rep. conocimiento
Lenguaje natural.
Ontología
De servicios.
descripción
Activar al agente que se encargue de atender el tipo de servicio del usuario.

Objetivo Instalar software

tipo
Objetivo reactivo.
parámetros-entrada
Nombre del software, datos de máquina, datos para realizar la instalación.
parámetros-salida
Lista de actividades y resultados obtenidos.
condición-activación
Inicialización del agente.

condición-finalización
Ninguna.
lenguaje rep. conocimiento
Lenguaje natural.
Ontología
De servicios.
descripción
Realizar la actividad de instalar el software que ha solicitado el usuario en su máquina, para lo cual tendrá que trasladarse.

Objetivo Configurar software
tipo
Objetivo reactivo.
parámetros-entrada
Nombre del software, datos de máquina, datos para realizar la configuración.
parámetros-salida
Lista de actividades y resultados obtenidos.
condición-activación
Inicialización del agente.
condición-finalización
Ninguna.
lenguaje rep. conocimiento
Lenguaje natural.
Ontología
De servicios.
descripción
Reaccionar a la indicación del agente Administrador y trasladarse a la máquina que necesite configurar el software instalado.

Objetivo Ejecutar aplicación
tipo
Objetivo reactivo.
parámetros-entrada
Nombre del software, datos de máquina.
parámetros-salida
Lista de actividades y resultados obtenidos.
condición-activación
Inicialización del agente.
condición-finalización
Ninguna.
lenguaje rep. conocimiento
Lenguaje natural.
Ontología
De servicios.

descripción
Activa aplicaciones con las que cuenta el sistema operativo por ejemplo: la corrida del anti-virus, iniciar el escáner del sistema de archivo, el escáner del registro de Windows.

Objetivo Recuperar registros
tipo
Objetivo reactivo.
parámetros-entrada
Nombre del software.
parámetros-salida
Lista de actividades y resultados obtenidos.
condición-activación
Inicialización del agente e inicialización del sistema en el equipo del usuario.
condición-finalización
Ninguna.
lenguaje rep. conocimiento
Lenguaje natural.
Ontología
De servicios.
descripción
Obtener los archivos de configuración que el sistema posee en sus registros, ya que si se presenta un problema con una aplicación instalada con anterioridad el agente entrará en acción para restablecer los registros creados por el sistema operativo.

Objetivo Administrar software
tipo
Objetivo reactivo.
parámetros-entrada
Características del software.
parámetros-salida
Ninguno.
condición-activación
Iniciar servicio.
condición-finalización
Ninguna.
lenguaje rep. conocimiento
Lenguaje natural.
Ontología
De servicios.
descripción
Almacenar los datos del software con el que trabaja el sistema basado en agentes.

Objetivo Trasladarse a PC

tipo
Objetivo reactivo.
parámetros-entrada
Dirección.
parámetros-salida
Ninguna.
condición-activación
Inicialización del agente.
condición-finalización
Ninguna.
lenguaje rep. conocimiento
Lenguaje natural.
Ontología
De servicios.
descripción
Tomar los datos y parámetros necesarios para dar el servicio, trasladándose con ellos a la máquina del usuario demandante.

3.4.6 Identificación de los objetivos empleando tarjetas CRC

La técnica de modelado de CRC proporciona un método para organizar las clases relevantes para modelar un sistema. Las tarjetas muestran la clase, los objetivos asignados al agente, los planes de cómo se realizan los objetivos y el conocimiento que se necesita para llevarlos a cabo.

Agente: Asistente		**Clase**: Agente_Interfaz		
Objetivos	**Planes**	**Conocimiento**	**Colaborador**	**Servicio**
Obtener necesidades del usuario	Obtener datos del equipo.	Clave y valor del registro.		
	Obtener datos del usuario.	Clave y valor del registro.		
		Interfaz gráfica.		
	Elaborar cadena de solicitud de servicio.		Secretario	Crear reporte
Determinar activación de agentes.	Coordinar acción del agente.	Tipo de problema (mal funcionamiento)	Restaurador.	Restaurar software.
		Otro problema	Administrador	Instalar
				Configurar
				Ejecutar aplicación

Agente: Supervisor		**Clase**: Agente_Interfaz		
Objetivos	**Planes**	**Conocimiento**	**Colaborador**	**Servicio**
Examinar el proceso de servicio.	Obtener información del reporte.	Reporte aleatorio.		
Diseñar estadísticas de servicio.	Elaborar estadísticas.	Reportes globales	Secretaria.	Obtener datos de reportes

Agente: Secretario		**Clase**: Agente_Interfaz		
Objetivos	**Planes**	**Conocimiento**	**Colaborador**	**Servicio**
Administrar Reportes.	Guardar archivo de reporte.	Tipo de cadena		
	Obtener información del reporte	Tipo de reporte.		
		Fecha de reporte.		
		No. de reporte.		

Agente: Administrador		**Clase**: Agente_Interfaz		
Objetivos	**Planes**	**Conocimiento**	**Colaborador**	**Servicio**
Determinar activación del agente.	Coordinar acción del agente.	Tipo de problema.	Instalador.	Instalar.
			Configurador.	Configurar.
			Ejecutor.	Ejecutar aplicación.
Administrar software	Ingresar datos DB.			
	Eliminar datos DB.			
	Obtener datos DB.			

Agente: Instalador		**Clase**: Agente móvil		
Objetivos	**Planes**	**Conocimiento**	**Colaborador**	**Servicio**
Instalar software.	Obtener datos de DB.	Software		
	Evaluar datos de PC.	Requerimientos de software y datos de PC.		
	Introducir parámetros.			
	Elaborar cadena de servicio	Se presenta problema	Administrador	Instalar
				Configurar
				Ejecutar aplicación
			Secretario	Crear reporte
Trasladarse a PC.	Moverse a la PC.	Dirección		

Agente: Ejecutor		**Clase**: Agente móvil		
Objetivos	**Planes**	**Conocimiento**	**Colaborador**	**Servicio**
Activar aplicación.	Ejecutar aplicación.	Aplicación		
	Elaborar cadena de servicio.	Se presenta problema	Administrador	Instalar
				Configurar
				Ejecutar aplicación
				Restaurar
			Secretario	Crear reporte
Trasladarse a PC.	Moverse a la PC.	Dirección		

Objetivos	**Planes**	**Conocimiento**	**Colaborador**	**Servicio**
Configurar Software.	Obtener datos de DB.	Software a configurar		
	Configurar.	Datos del software		
	Elaborar cadena de servicio.	Se presenta problema	Administrador	Instalar
				Configurar
				Ejecutar aplicación
			Secretario	Crear reporte
Trasladarse a PC.	Moverse a la PC.	Dirección		

Agente: Restaurador		**Clase**: Agente		
Objetivos	**Planes**	**Conocimiento**	**Colaborador**	**Servicio**
Recuperar Registros.	Obtener archivos de configuración del equipo.	Registro de Windows.		
	Restaurar archivos de configuración del equipo.			
	Obtener datos de DB.	Dirección		
	Crear accesos directos.	Aplicación.		
	Elaborar cadena de servicio.	Se presenta problema	Administrador	Instalar
				Configurar
				Ejecutar aplicación
			Secretario	Crear reporte

Las tarjetas CRC de los agentes ilustran de una forma íntegra cuales son los objetivos, como se llevan a cabo los procesos internos del agente y dependiendo de los casos que se presenten, como se llevan a cabo la colaboración con otros agentes y a través de que servicios.

3.4.7 Identificación de los servicios

Los servicios son tareas que un agente ofrece a otro agente, derivándose de las conversaciones, por tanto de las tareas que se identificaron de cada agente se han identificado los siguientes servicios que pueden ser demandados por los agentes:

- Restaurar aplicaciones.
- Crear reportes.
- Obtener datos de reportes.
- Instalar.
- Configurar.
- Ejecutar aplicaciones.

3.4.8 Descripción de los servicios

Ya que se han identificado los servicios en el punto anterior, a continuación se presenta la descripción de cada uno de los servicios empleando la utilización de las plantillas de servicios que permiten reflejar el objetivo del servicio, que parámetros va a necesitar para llevarse a cabo y la ontología que se empleará para el entendimiento entre los agentes.

Servicio *Restaurar aplicaciones*
tipo
Una actividad.
objetivo
Restaurar los archivos de registros que contiene la configuración de aplicaciones con anomalías.
parámetros-entrada
Activación, aplicación.
ontología
Servicios.

Servicio *Crear reportes*
tipo
Documentación.
objetivo
Conformar unas cadenas en un documento con formato específico del tipo de cadena entrante.
parámetros-entrada
Sentencia del servicio.
ontología
Servicios.

Servicio *Obtener datos de reportes*

tipo
Documentación.
objetivo
Extraer de un documento con formato específico, los datos demandados.
parámetros-entrada
Tipo de documento, datos requeridos.
ontología
Servicios.

Servicio *Instalar*

tipo
Una actividad.
objetivo
Trasladarse a la máquina que necesite el software e instalarlo.
parámetros-entrada
Activación y sentencias indicativas del software.
ontología
Servicios.

Servicio *Configurar*

tipo
Una actividad.
objetivo
Obtener datos necesarios de la DB y trasladarse a la máquina del usuario e introducir los parámetros necesarios.
parámetros-entrada
Activación, nombre de la máquina y aplicación.
ontología
Servicios.

Servicio *Ejecutar aplicaciones*

tipo
Una actividad.
objetivo
Trasladarse a la máquina y ejecutar una aplicación específica.
parámetros-entrada
Activación y sentencias indicativas del software a ejecutar.
ontología
Servicios.

3.5 Modelo de Coordinación

El modelo de coordinación está propuesto para desarrollar y describir las interacciones entre los agentes de un sistema multi-agente, involucrados en la resolución de un problema, en donde la definición de conversación se establece como un conjunto de interacciones iniciadas con el fin de alcanzar un objetivo.

3.5.1 Identificación de las conversaciones

En el modelo de agente se han identificado algunas conversaciones existentes entre los agentes del sistema. Estas conversaciones son: Comunicar problemática, activar agente, enviar datos de servicio, intercambio de reporte, activación del agente adecuado. El diagrama de casos de uso (Fig. 3.7) ilustra entre qué agentes se llevan a cabo las conversaciones.

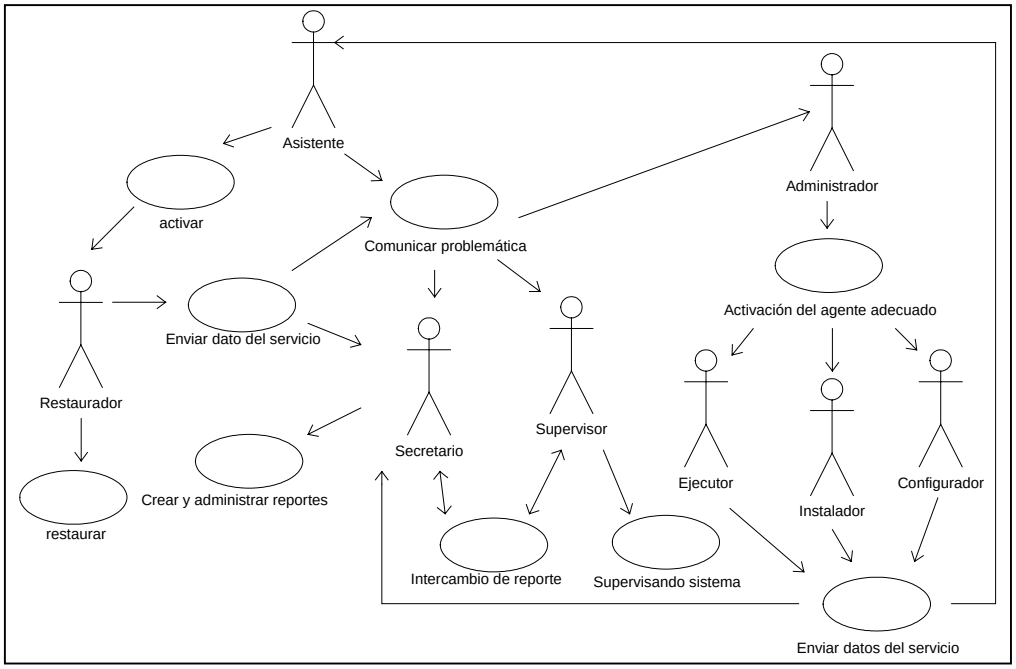

Fig. 3.7 Diagrama de Casos de Uso internos

3.5.2 Descripción de las conversaciones

De las conversaciones identificadas, a continuación se presenta una descripción textual de las mismas por medio de las plantillas de conversaciones, las cuales analizan cuál es el objetivo de la conversación, su pre y post condiciones, y qué participantes existen.

Conversación *Comunicar problemática*

tipo
Comunicar-información.

objetivo
Comunicar a los agentes Secretario, Administrador y Supervisor, los requerimientos por parte del usuario.

agentes
Asistente, Secretario, Administrador y Supervisor.

iniciador
Asistente.

servicio
Crear reporte, instalar, configurar, ejecutar aplicación.

descripción
El agente Asistente tiene por objetivo comunicar a los agentes involucrados en proporcionar los servicios del sistema, los requerimientos del usuario.

precondición
Activación del agente.

postcondición
Envío de datos.

Conversación *Activar*

tipo
Comunicar-información.

objetivo
Activar al agente Restaurador cuando reciba una señal de restaurar archivos de una determinada aplicación.

agentes
Asistente y Restaurador.

iniciador
Asistente.

servicio
Restaurar aplicaciones.

descripción
El agente Asistente pasará el nombre del software que tiene problemas para que el agente Restaurador se ejecute y restaure los registros o accesos directos.

precondición
Presente problemas una aplicación.

postcondición
Envío de datos.

Conversación *Enviar datos del Servicio*

tipo
Comunicar-información.
objetivo
Comunicar al Secretario actividades realizadas.
agentes
Reconfigurador, Ejecutor, Instalador, Configurador y Secretario.
iniciador
Reconfigurador, Ejecutor, Instalador y Configurador.
servicio
Crear reporte.
descripción
Los agentes Reconfigurador, Ejecutor, Instalador o Configurador una vez realizadas sus labores en el equipo del usuario, tienen que informar de ellas al Secretario para que cree el reporte correspondiente del servicio prestado al usuario.
precondición
Problema en el equipo del usuario.
postcondición
Informe comunicado.

Conversación *Intercambio de Reporte.*

tipo
Comunicar-información.
objetivo
Transferir información de un reporte.
agentes
Secretario y Supervisor.
iniciador
Supervisor.
servicio
Obtener datos de reporte.
descripción
El agente supervisor realiza una petición de la información contenida en los reportes para poder procesar los datos o simplemente visualizarlos.
precondición
Una petición del Supervisor.
postcondición
Información transferida.

Conversación *Activación del agente adecuado.*

tipo
Comunicar-información.
objetivo
Activar agente y transmitir información para llevar a cabo su servicio.
agentes
Administrador, Instalador, Ejecutor y Configurador.
iniciador
Administrador.
servicio
Instalar, configurar, ejecutar aplicación.
descripción
El agente Administrador es el encargado de activar a los agentes que pueden dar el servicio de las solicitudes que han llegado a él y de pasar los datos que éstos necesitan para proceder con su tarea.
precondición
La entrada de un servicio.
postcondición
Activación de un agente.

3.5.3 Descripción de las intervenciones

La descripción de las intervenciones de una conversación tiene por objetivo determinar los mensajes intercambiados entre los agentes, para lo cual se hace uso de los diagramas de secuencia del A-UML, permitiendo describir los mensajes intercambiados entre los agentes.

El diagrama de secuencia de la figura. 3.8 muestra la activación del agente adecuado, refleja cómo es activado el agente que se pretende iniciar y luego el paso de los mensajes que contienen la información para poder llevar a cabo el servicio.

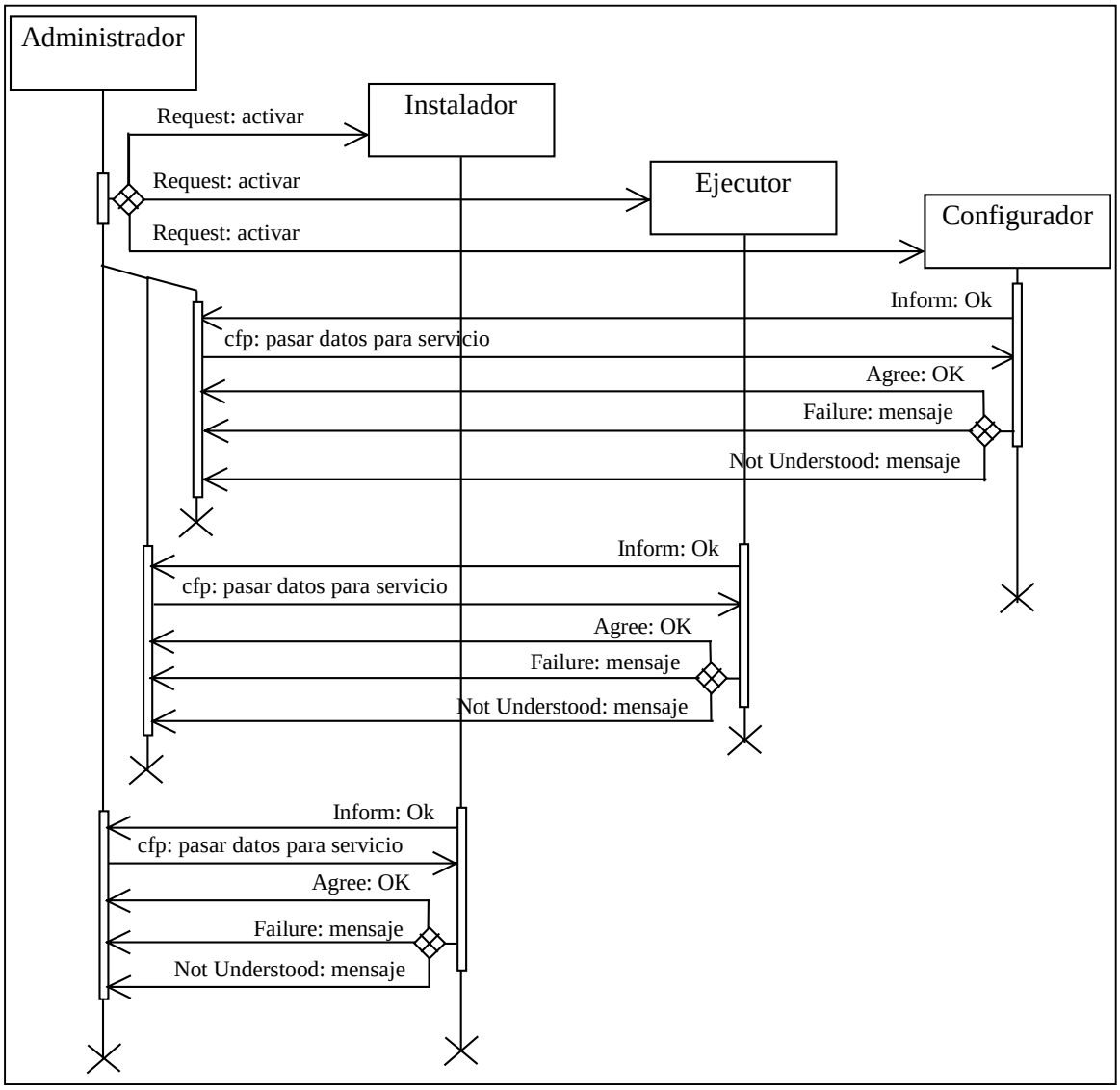

Fig. 3.8 Diagrama de secuencia de la "Activación del agente adecuado"

El diagrama de secuencias que a continuación se presenta (Fig. 3.9) muestra el intercambio de mensajes existentes entre los agentes. Iniciando la comunicación el agente Asistente transmite un mensaje que indica que existe un problema y espera que exista una respuesta de los demás agentes para transmitir las características del problema presentado.

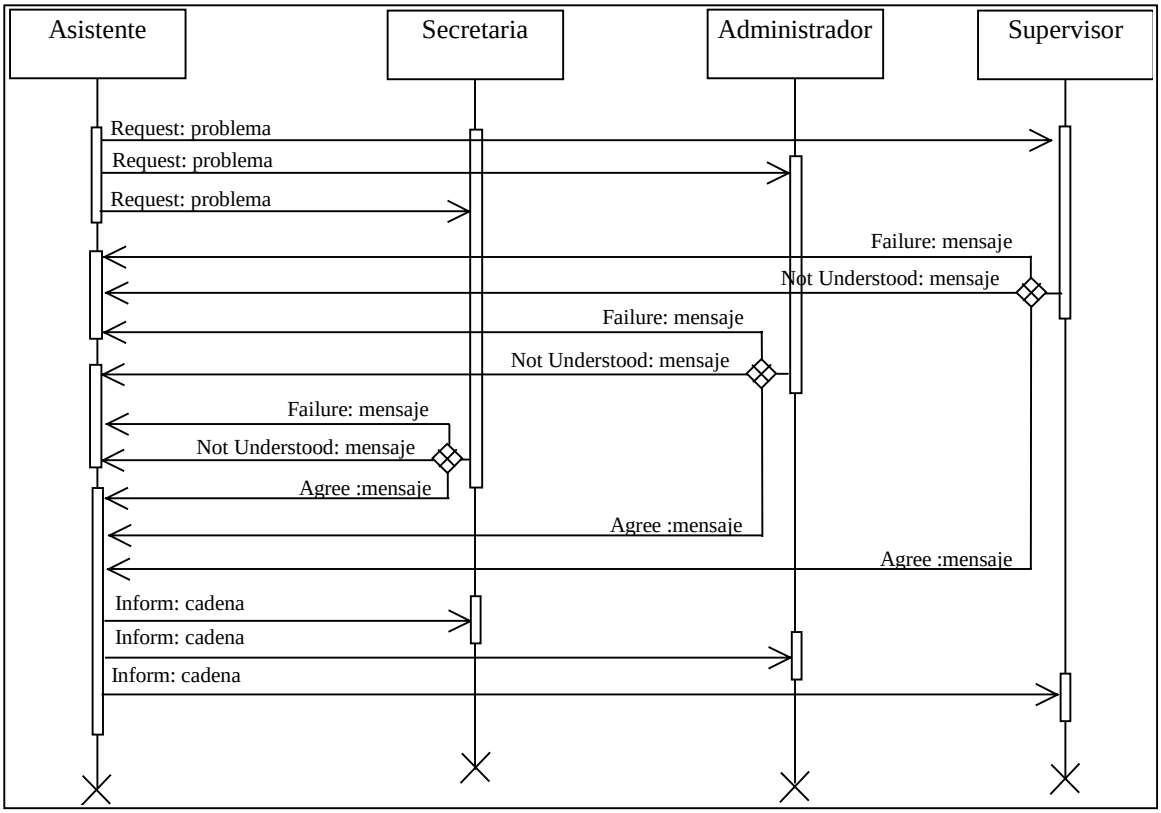

Fig. 3.9 Diagrama de secuencia "Comunicar problemática"

3.6 Modelo de Experiencia

El desarrollo del modelo de experiencia en un sistema multi-agente puede subdividirse en el conocimiento de la aplicación, el propio agente, el resto de agentes y el entorno. En este modelo se desarrollan las tareas que requieren conocimiento para ser llevadas a cabo y que permitirán caracterizar al agente como un sistema basado en conocimiento.

3.6.1 Identificación de las tareas genéricas

En el dominio de aplicación se identifican las tareas genéricas que se adaptan al problema que desea resolverse, por tanto la siguiente lista muestra las tareas genéricas:

- Elaborar Cadena De Servicio
- Elaborar Solicitud De Servicio
- Obtener Datos De BD
- Moverse A La Máquina
- Introducir Parámetros

3.6.2 Identificación y descripción del esquema del modelo

En este apartado se identifican, organizan y describen las ontologías o esquemas del modelo, que representan las estructuras de los conceptos del dominio y sus relaciones empleando un lenguaje de representación de conocimiento.

Concepto *Usuario*
descripción
El usuario tiene asociado una clave de empleado, área, extensión y gerencia a la que pertenece.
propiedades
identificador: cadena
clave: entero
nombre: cadena
extensión: cadena
gerencia: cadena

Concepto *Software*
descripción
Las características mínimas demandantes del software.
propiedades
identificador: cadena
procesador: cadena
OS: cadena
ram: entero
DD: entero
Video: entero

Concepto *Máquina*
descripción
Características que presenta el equipo del usuario tanto para ser identificada, como para conocer sus capacidades.
propiedades
identificador: cadena
ip: cadena
procesador: cadena
OS: cadena
ram: entero
DD: entero
Video: entero

Concepto *Servidor*
descripción
Máquina que contiene la base de datos, software.
propiedades
identificador: cadena

ip: cadena
BaseDeDatos: cadena //nombre de la base de datos

Concepto *Problema*
descripción
Anomalía que presenta el software de un equipo y sobre el cual se debe proporcionar el servicio.
propiedades
identificador: cadena
aplicación: cadena //nombre del programa que presenta anomalías
accesoDirecto: booleano
instalación: booleano
configuración: booleano

Concepto Reporte global
descripción
Contiene información sobre los problemas que presentan los usuarios y las soluciones dadas.
propiedades
identificador: cadena
número: entero
software: cadena
problema: cadena
solución: cadena
día: entero {1, 2.., 31}
mes: entero {1, 2,...,12}
año: {00,01,...,99}

Concepto *Reporte de servicio*
descripción
Contiene información sobre los procedimientos llevados a cabo por el agente para solucionar el problema.
propiedades
identificador: cadena
técnico: cadena
número: entero
software: cadena
solución: cadena
día: entero {1, 2.., 31}
mes: entero {1, 2,...,12}
año: {00,01,...,99}

Concepto *Reporte de solicitud*

descripción

Información referente al problema que presenta la máquina del usuario.

propiedades

identificador: cadena
usuario: cadena
número: entero
software: cadena
problema: cadena
día: entero {1, 2.., 31}
mes: entero {1, 2,...,12}
año: {00,01,...,99}

Concepto *Aplicación*

descripción

Software específico.

propiedades

identificador: cadena
ubicación: cadena
nombre: cadena
parámetro: cadena

Concepto *Instalar*

descripción

Procedimiento que se realiza con la finalidad de que una máquina cuente con un software.

propiedades

identificador: cadena
software: cadena
ubicación: cadena
aplicación: cadena

Concepto Reinstalar

descripción

Procedimiento que se realiza cuando se pretende reparar una aplicación instalada con anterioridad y presenta fallas.

propiedades

identificador: cadena
software: cadena
ubicación: cadena
aplicación: cadena

Concepto *Configurar*

descripción

Procedimiento que se realiza cuando una aplicación necesita de parámetros para su funcionamiento.

propiedades

identificador: cadena
software: cadena
ubicación: cadena //de la máquina
usuario: cadena
proxy: cadena //dirección proxy
puerto: entero //puerto del proxy
dirección de inicio: cadena // inicio del explorador
e-mail: cadena //para el outlook
tipo de servidor entrante: cadena
servidor de correo entrante: cadena
servidor de correo saliente: cadena

Concepto *Acceso Directo*

descripción

Características necesarias para crear una liga a un programa.

propiedades

identificador: cadena
ubicación: cadena
aplicación: cadena
dirección: cadena
ubicación del acceso: cadena

Concepto *Base de Datos*

descripción

Contiene información de los parámetros necesarios para llevar a cabo las tareas del sistema.

propiedades

identificador: cadena
ubicación: cadena
parámetro: cadena

Concepto *Estadística de servicio*

descripción

Información necesaria que conforma la estadística de servicio.

propiedades

identificador: cadena
problema: cadena
solución: cadena
agente: cadena

Concepto Guardar Archivo
descripción
Información de la ubicación y nombre con el que se guardarán los reportes.
propiedades
identificador: cadena
tipoArchivo: cadena
carpeta: cadena
nombreArchivo: cadena
día: entero //los archivos se guardan en carpetas de acuerdo a la
mes: entero //fecha y al tipo de archivo
año: entero

Concepto *Parámetros del software*
descripción
Información indispensable para instalar una aplicación.
propiedades
identificación: cadena
númeroSerie: cadena
usuario: cadena
compañía: cadena
tipoInstalación: cadena
aplicaciones: cadena // aplicaciones a ser instaladas

3.6.3 Correspondencia entre el esquema del modelo y las tareas genéricas

En esta sección se plasman las relaciones que guardan los esquemas del modelo respecto a las tareas genéricas, identificándose las siguientes relaciones:

- ElaborarCadenaDeServicio **USA**: Usuario, Máquina, Servidor, Aplicación.
- ObtenerDatosDeBD **USA**: Software, Aplicación, Configurar, BaseDeConocimiento, ParámetrosDelSoftware.
- MoverseAMáquina **USA**: Máquina.
- IntroducirParámetros **USA**: ParámetrosDelSoftware, Configuración, Aplicación.

3.7 Modelo de Comunicación

El objetivo del modelo de comunicación es modelar las interacciones hombre-máquina. Para modelar estas interacciones, se debe descomponer el diálogo entre el programa y el usuario, indicando los intercambios elementales de información que se llevan a cabo. Se determina qué diálogos hombre-máquina son necesarios, quién toma la iniciativa en cada diálogo, qué explicaciones son apropiadas en cada situación, qué medio es más adecuado.

3.7.1 Identificación de la comunicación entre hombre-agente

Los actos de comunicación que se han identificado entre el hombre y la máquina (agentes) son los mismos que se encuentran enunciados por medio de los casos de uso que se identificaron en la sección de conceptualización (sección 3.2.3), por lo tanto no se enlistarán nuevamente.

3.7.2 Descripción de la comunicación entre hombre-agente

A continuación se describen por medio de plantillas las interacciones que se usan en la parte principal del sistema.

Comunicación *Crear Reporte de Solicitud de Servicio*

descripción

El usuario inicia al agente Asistente que muestra la interfaz con los campos para que el usuario introduzca sus datos personales. Donde fueron introducidos previamente se desplegarán junto con las opciones a seleccionar para solicitar un servicio. Si se solicita el servicio y faltan datos importantes el agente debe solicitarlos, si no existe una comunicación, los demás agentes deben enviar un mensaje mostrando la extensión donde deben reportar el problema. Si se contactó al agente debe desplegar un mensaje del proceso de la solicitud.

Comunicación *Confirmación de Servicio*

descripción

El agente supervisor muestra un mensaje en la pantalla del usuario que solicitó un servicio donde pregunta si el servicio se completo satisfactoriamente. El usuario debe responder con un “si” o “no”, si la respuesta es “no” debe preguntar si se reportó el fallo del servicio a la extensión indicada.

Comunicación *Crear Característica de Software Nuevo*

descripción

El usuario activa la interfaz del Administrador la cual despliega la pantalla principal con menús, el usuario debe seleccionar la opción para “crear característica de software nuevo” enseguida el agente muestra los campos para introducir las características del software. El usuario debe completar los campos, si faltan datos vitales debe solicitarlos el agente. Si se guardan los datos debe mostrar los campos vacíos para introducir un nuevo software, y si es cancelado el proceso debe mostrar la pantalla principal del Administrador.

Comunicación *Editar Características de Software*

descripción

El usuario activa la interfaz del Administrador la cual despliega la pantalla principal con menús, el usuario debe seleccionar la opción para “editar características de software” se muestra una lista con las características de software registradas, el usuario debe seleccionar una. Ya seleccionada deben aparecer los campos con los datos del software seleccionado. A continuación el usuario debe hacer las modificaciones pertinentes y una vez terminada debe guardar los datos. Si se guardan los datos debe mostrar un mensaje que indica que se guardaron y posteriormente al igual que si se cancela la edición debe mostrar la pantalla principal del Administrador.

Comunicación *Eliminar Características de Software*

descripción

El usuario activa la interfaz del Administrador la cual despliega la pantalla principal con menús, el usuario debe seleccionar la opción para “eliminar características de software” se muestra una lista con las características de software registradas, el usuario debe seleccionar una. Ya seleccionada deben aparecer los campos con los datos del software seleccionado. Ahora si el usuario selecciona cancelar se muestra la pantalla principal, en tanto que si se selecciona eliminar el agente muestra una ventana con la confirmación de que realmente desea eliminar las características del software, el usuario debe decir si lo desea o “no”. Si se selecciona “si”, debe mostrarse la pantalla principal, si es “no” debe regresarse a la pantalla de características.

Comunicación *Leer Reporte*

descripción

El usuario activa la interfaz del agente (Supervisor, Administrador o Secretario) la cual despliega la pantalla principal con menús donde debe seleccionarse la opción “leer reporte”, enseguida se desplegará una ventana con una lista de los reportes existentes o bien a los que puede tener acceso cada uno de los diferentes usuarios, debe seleccionar uno el usuario, ya seleccionado el agente mostrará la información del reporte

específico. Si en el momento de elegir la opción de leer existe un fallo debe mostrarse un mensaje que indique que no existe comunicación entre el sistema. Cuando termine el usuario de visualizar el reporte debe indicar al agente que terminó por medio de un botón y por tanto se presentará la pantalla principal de la interfaz de agente.

3.8 Modelo de Organización

Con este modelo se describe tanto la organización humana en que el sistema multi-agente va a ser introducido, como la organización interna del sistema multi-agente, especificando las relaciones "estáticas" o "estructurales" entre los agentes del sistema, como por ejemplo, las relaciones de herencia, agregación, autoridad, utilización, etc..

La descripción de la organización del Instituto Mexicano del Petróleo se ha descrito en la Introducción de la tesis, por lo cual a continuación se describe la organización interna del sistema MAS.

3.8.1 Identificación de las relaciones de herencia

La relación de herencia entre agentes permite agrupar clases de agentes que comparten algunas capacidades parecidas (servicios, sensores, actuadores, objetos, creencias, etc.). Se han identificado las relaciones de herencia entre los agentes tomando los agentes descritos en el modelo de agente.

Se identificaron las siguientes clases de agentes:

- Clase de los agentes de Interfaz (AgInterfaz), agrupa a los agentes que tienen que mostrar una interfaz con el usuario.
- Clase de los que proporcionan el servicio (AgServicio), agrupa a los agentes que tienen la característica de ofrecer servicios a otros agentes.
- Clase de agentes móviles agente (AgServicioMovil), se desprende de la clase de agentes AgServicio, pero que además tienen la característica de ser móviles.

Para ilustrar de una forma más simple y fácil de entender la herencia de los agentes se presenta el diagrama de clases de agentes (Fig. 3.10) con sus respectivas herencias.

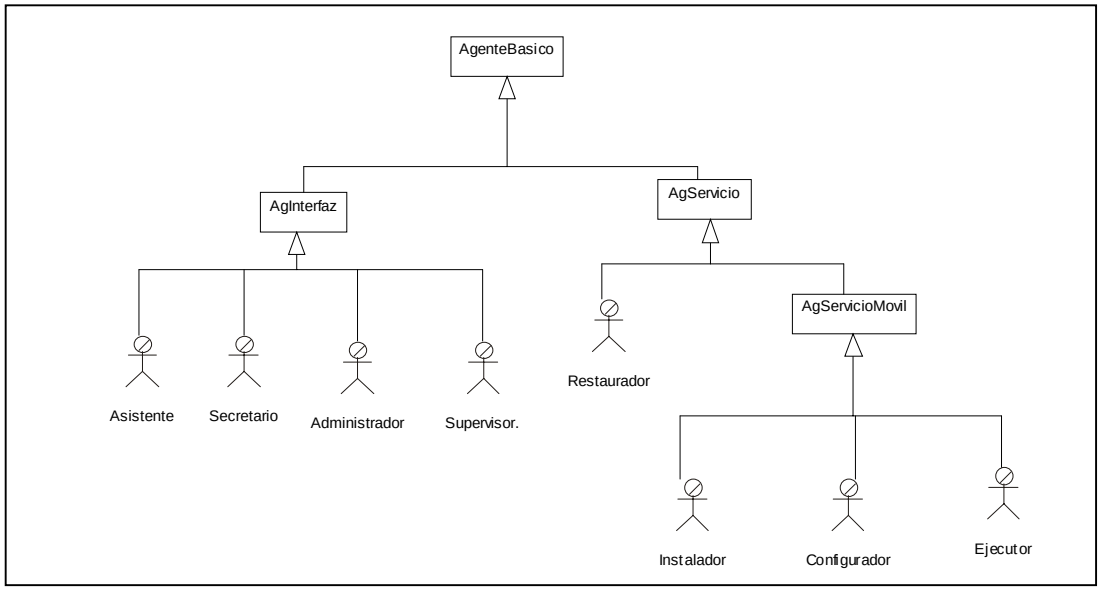

Fig. 3.10 Jerarquía de agentes

3.8.2 Identificación de los objetos del entorno

Se necesita identificar los objetos del entorno con los que los agentes interactúan a través de sensores y actuadores. El modelado de los objetos del entorno permite identificar las características y métodos relevantes para el problema.

Los objetos del entorno identificados son:

- Cadena de solicitud de servicio
- Cadena de reporte global
- Cadena de reporte de servicio
- Cadena de activación de agente
- Base de Datos
- Estadísticas
- Usuario
- Secretaria
- Supervisor
- Técnico

En las figuras 3.11 y 3.12 se muestra la estructura lógica de la organización de agentes y de la relación que existe entre la organización de agentes con los objetos que se identifican del entorno respectivamente.

3.9 Modelo de Diseño

El modelo de diseño posee la particularidad de documentar todas las decisiones de diseño. Para cumplir con estas decisiones el modelo distingue tres clases de decisiones de diseño: *diseño de la red*, que consiste en diseñar el modelo de red, el cual determina las funciones de los agentes de red encargados de mantener la sociedad multi-agente (facilitadores, gestores de base de datos, etc.); *diseño de los agentes*, descompone cada agente en subsistemas, que permiten documentar la arquitectura seleccionada para cada agente y las funciones de los módulos que deben implementarse en dicha arquitectura; y *diseño de la plataforma*, que recoge las decisiones de bajo nivel sobre el lenguaje de implementación seleccionando, software y hardware empleado y los usuarios finales del sistema.

3.9.1 Diseño del modelo de red

En esta actividad se presentan las facilidades que debe ofrecer el modelo de red. En la plantilla siguiente se muestra las facilidades funcionales básicas, que proporcionan los mecanismos mínimos para obtener una dirección y entrar/salir de la red. Estos agentes son proporcionados por la plataforma de agentes que se pretende emplear, cuya descripción se realiza posteriormente en la sección 3.9.3.

Red *Red-Instalación-Software*
tipo
tcp/ip.
servicios-de-red
Registrarse, registrar-salida, activar-servicio, desactivar-servicio, consulta-páginas-amarillas, coordinar-agentes.
Agentes-de-red
Agente DF, Agente AMS, Agente Administrador.

3.9.2 Diseño de los agentes

En el diseño de los agentes se selecciona una arquitectura para cada agente determinando qué arquitectura es la más adecuada para el desempeño de cada agente.

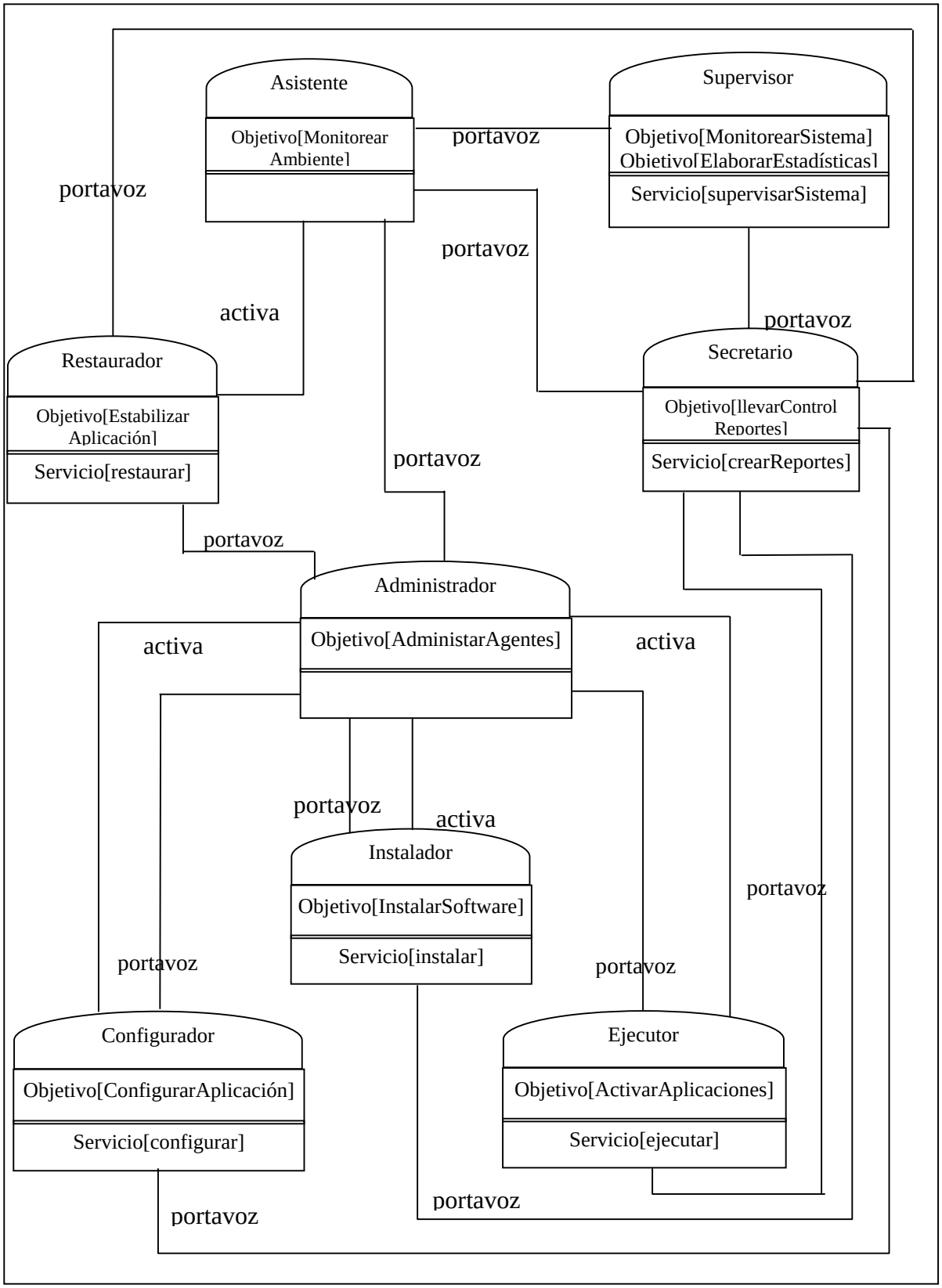

Fig. 3.11 Estructura organizativa de agentes.

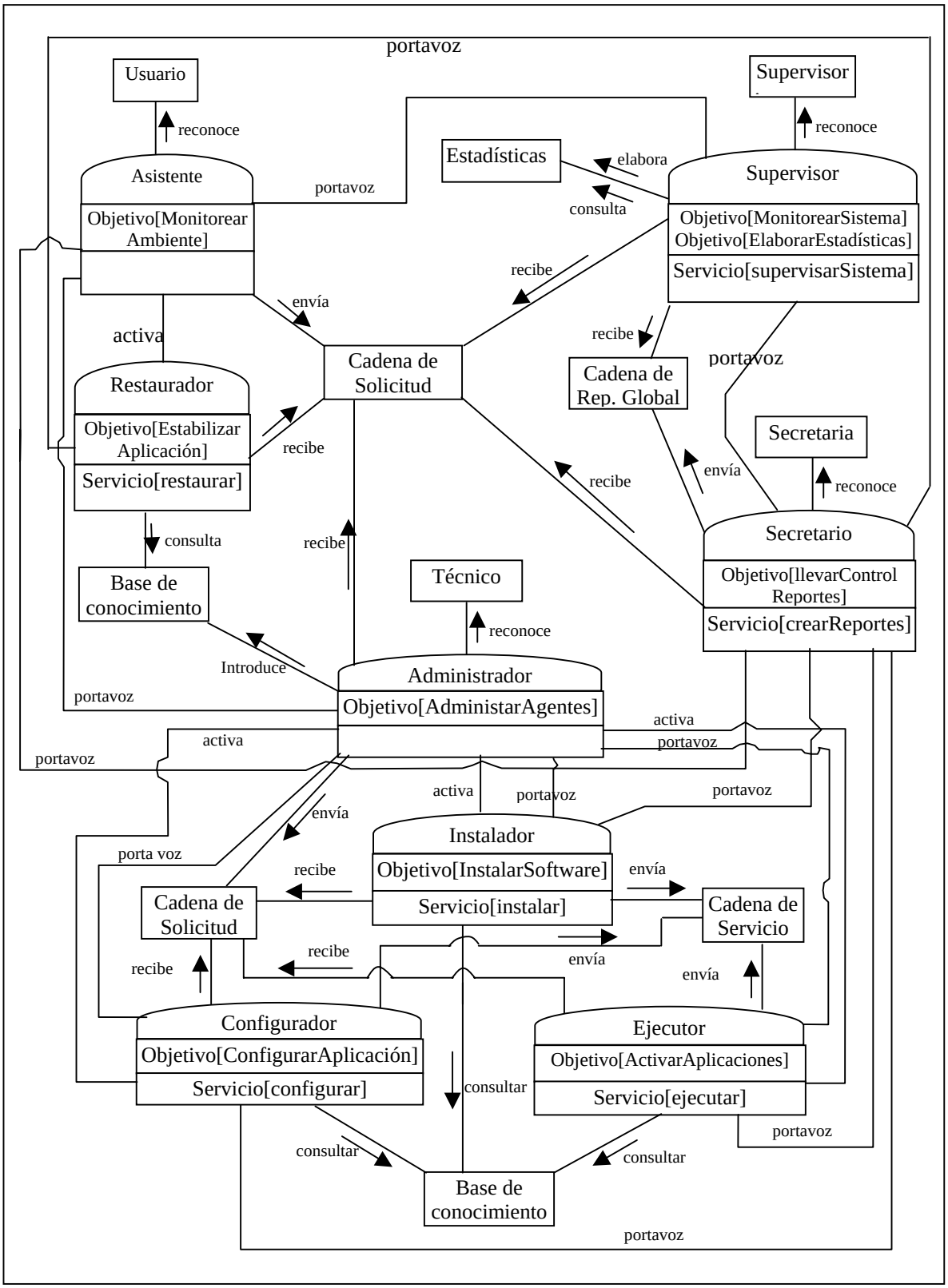

Fig. 3.12 Relación de la estructura organizativa de agentes con los objetos del entorno.

La elección del tipo de arquitectura de los diferentes agentes está basada en cómo se pretenden llevar a cabo las tareas de cada uno de los agentes. Por tanto las arquitecturas reactivas han sido elegidas por pretenderse que el agente reaccione a los estímulos que se encuentren en su entorno. En tanto que la elección de las arquitecturas híbridas se encuentran fundamentadas en que los agentes no siempre se van a topar con condiciones óptimas y tendrán que evaluar el entorno de trabajo para lograr sus objetivos.

Sistema-Agente *Asistente*
arquitectura
Arquitectura reactiva.
implementa
Obtener datos del equipo, obtener datos del usuario, elaborar cadena de servicio, coordinar acción de agentes.

Sistema-Agente *Secretario*
arquitectura
Arquitectura reactiva.
implementa
Obtener información del reporte, guardar archivo de reporte.

Sistema-Agente *Supervisor*
arquitectura
Arquitectura reactiva.
implementa
Elaborar cadena de solicitud de servicio, elaborar estadísticas.

Sistema-Agente *Administrador*
arquitectura
Arquitectura híbrida.
implementa
Coordinar acción de agentes, ingresar datos a DB, obtener datos de DB.

Sistema-Agente *Restaurador*
arquitectura
Arquitectura híbrida.
implementa
Elaborar cadena de servicio, crear accesos directos, restaurar archivos de configuración del equipo, obtener datos de DB.

Sistema-Agente *Ejecutor*
arquitectura
Arquitectura híbrida.
implementa
Elaborar cadena de servicio, ejecutar aplicación.

Sistema-Agente *Instalador*
arquitectura
Arquitectura híbrida.
implementa
Elaborar cadena de servicio, instalar software, introducir parámetros de instalación, evaluar datos de pc, obtener datos de DB.

Sistema-Agente *Configurador*
arquitectura
Arquitectura híbrida.
implementa
Elaborar cadena de servicio, configurar, obtener datos de DB.

3.9.3 Diseño de la plataforma

En este apartado se han documentado las decisiones del software y hardware empleado, basándose en el diseño de la arquitectura de agente seleccionada.

Del grupo de herramientas que se han descrito en la sección 2.4, se tiene que seleccionar una para llevar a cabo el desarrollo del sistema (prototipo) que pretende solucionar las problemáticas detectadas (sección: *Descripción de la problemática encontrada*) en el área de Soporte Informático de la Gerencia de Tecnología Informática. El primer criterio de selección se basa en el tipo de distribución de la herramienta, ésto es, si es un producto comercial o freeware, y según el criterio se descartan para realizar el desarrollo las herramientas de carácter comercial, al no estar disponibles en el marco de la tesis. Por lo tanto se descartan las herramientas AgentBuilder, Concordia y Aglets.

El siguiente criterio empleado para la selección de la herramienta es que permita ejecutarse en equipos que cuenten con sistema operativo Windows, por lo cual, se tiene que descartar D'Agents ya que esta herramienta sólo puede ejecutarse en equipos que cuenten con sistemas Unix. Ahora bien se descarta JATLite ya que sólo permite la creación de agentes localmente y como el sistema está diseñado para que los agentes además de ejecutarse localmente deben en algunas situaciones especiales ejecutarse remotamente, por tanto ha sido desechada esta herramienta. Las herramientas restantes (FIPA-OS, JADE y ZEUS) permiten la creación de agentes tanto local como remotamente, sin embargo tomaremos para esta tesis la que esté basada en los estándares internacionales para la construcción de agentes de la FIPA, por lo cual, ZEUS es descartada.

Las herramientas FIPA-OS y JADE son las que tratan de cumplir totalmente con los estándares marcados por FIPA, pero en el desarrollo del prototipo de esta tesis me inclino por la elección de la plataforma para construcción de agentes JADE ya que de entre las dos herramientas se ha utilizado en un mayor número de

sistemas de agentes, no sólo para proyectos académicos, sino también en proyectos comerciales, aunado a que las distribuciones de las diferentes versiones de FIPA-OS no son totalmente compatibles entre ellas y ésto en un futuro podría causar problemas ya que se tiene en mente la construcción de otros sistemas dentro del Instituto Mexicano del Petróleo y si se desea que los demás proyectos cooperen entre ellos, ésto podría involucrar retrasos en los proyectos al tratar de hacer que las plataformas de agentes sean compatibles. Otra cosa que perjudicaría si las plataformas de agentes no fueran totalmente compatibles sería la administración remota de las plataformas y por ende de los agentes.

3.9.3.1 Plataforma de Agentes JADE

FIPA ha establecido un modelo estándar de una plataforma de agentes, que se encuentra representada por la siguiente figura (Fig. 3.14), donde el *agente* es fundamentalmente el actor que reside en la plataforma de agentes, combina uno o más servicios capaces de unificar e integrarlo en un modelo de ejecución que pueda incluir el acceso a software externo, a usuarios humanos y permita facilitar la comunicación; el *Sistema de Administración de Agentes* (AMS) es el agente que permite llevar el control de supervisión durante la ejecución, sobre el acceso y el uso de la plataforma de agentes. Solamente puede existir un sólo AMS en la plataforma. El AMS proporciona los servicios de páginas blancas y el ciclo de vida, manteniendo un directorio de identificación de agentes (AID) y del estado de los agentes. Cada agente debe registrarse con un AMS para obtener un AID válido; el *Facilitador de Directorio* (DF) es el agente quien provee el servicio de páginas amarillas en la plataforma por default; el *Sistema de Transporte de Mensajes*, también llamado Canal de Comunicación de Agentes (ACC), es el componente de software que controla el intercambio de mensajes dentro de la plataforma, incluyendo los mensajes entre otras plataformas; la *plataforma de agentes* (AP) provee la infraestructura en la cual los agentes se pueden desempeñar. La AP está constituida por mecanismos, del sistema operativo, software de soporte de agentes, componentes FIPA para la administración de agentes (DF, AMS, y MTS) y agentes [31].

Ya descritas las características primordiales de la plataforma de agentes JADE se describe en la plantilla los requerimientos necesarios para la utilización del prototipo de sistema de agentes.

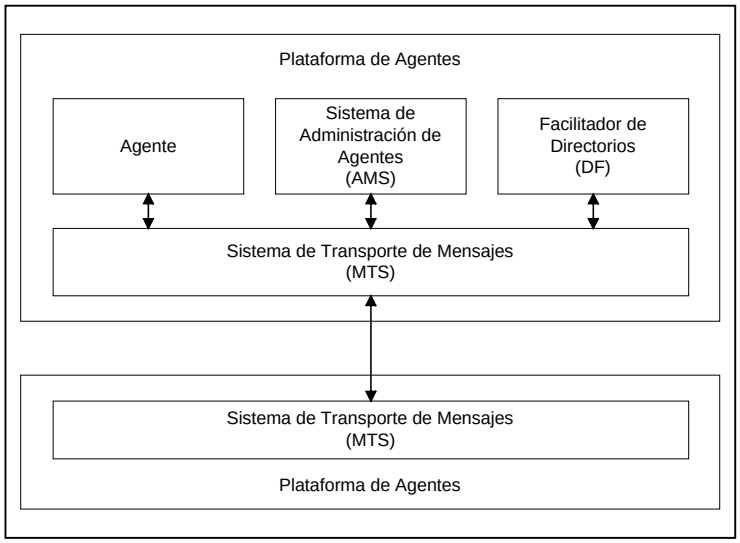

Fig. 3.14 Arquitectura de referencia de una plataforma de agentes FIPA

Plataforma *Sistema-Multi-agente*

descripción

El MAS pretende instalar software en la máquina de los usuarios, corregir problemas de configuración, así como reparar algunas anomalías que presente el sistema operativo, ésto de forma remota y a petición de los usuarios.

usa-lenguaje

Java, sql, FIPA-ACL.

hardware-requerido

Servidor de trabajo con Windows 2000 y PC's con Windows 98 en adelante.

software-requerido

Plataforma de agentes JADE 3.0b1, j2sdk1.4.0, Borland JBuilder versión 8, Microsoft Access 2000.

u**suario**

Técnico, Usuarios, Secretaria y Supervisor.

CAPÍTULO 4
Prototipo del Sistema.

4.1 Introducción

El prototipo del MAS para la instalación remota de software se enfoca únicamente a proporcionar el servicio de instalación, por tanto, los agentes involucrados directos a dar el servicio son:

- Asistente.- agente que captura los requerimientos del usuario.
- Administrador.- agente que captura las solicitudes y elige al agente que puede proporcionar el servicio y que administra el software sobre el cual puede proporcionar sus servicios.
- Instalador.- agente que instala el software requerido por el usuario.

Dentro de este capítulo se describe cómo se ha trabajado con la implementación del prototipo, las pruebas que se han realizado, una sugerencia de cómo seguir la metodología MAS-CommonKADS para implementar y por último la descripción general de los componentes que integran al prototipo.

4.2 Secuencia de trabajo en el desarrollo del prototipo

Antes de iniciar con la construcción del prototipo multi-agente se procedió a conocer la estructura de la plataforma de agentes, básicamente enfocándose a la parte de registrar los agentes dentro de la plataforma y el intercambio de mensajes entre agentes, ya que son algunos de las actividades adicionales que se observan con respecto a la creación de software tradicional.

Cuando se adquirió el conocimiento básico de cómo trabajan estos mecanismos, se procedió a desarrollar las tareas que se describen en el modelo de tareas del sistema. Si se revisan las tareas del modelo, se podrá encontrar que existen algunas actividades que necesitan haberse realizado previamente, como por ejemplo tener la interfaz gráfica para obtener información del usuario del sistema o bien haber creado alguno de los agentes, por lo cual al ir desarrollando algunas de las tareas se encontraban algunos inconvenientes para conformar las tareas descritas y necesarias para la implantación del prototipo.

En el caso de la tarea que necesita la creación de un interfaz se procedió a construirla. La creación de la interfaz no fue un problema debido a que se tomó la información contenida en los modelos de comunicación y conceptualización.

Como se describe, se dejó pendiente la construcción de tareas y se comenzó a trabajar con la implementación de otros modelos de la metodología que son requeridos para continuar la implementación de tareas.

El modelo sobre el cual continuó la implementación del prototipo fue el de agentes. Este modelo también permite resolver algunas problemáticas detectadas para la implementación de tareas.

Se comenzó a trabajar con los agentes que requiere el prototipo del sistema (Asistente, Administrador e Instalador). Durante la implementación de los agentes se notó que nuevamente existe la necesidad de trabajar previamente con la implementación de otros modelos de la metodología, como es el caso del modelo de coordinación, que permite implementar la parte de comunicación requerida por los agentes para negociar la prestación de servicios. Por tanto se prosiguió con la implementación del modelo de coordinación.

Al revisar los requerimientos para implementar el modelo de coordinación se observó que se debe contar con la ontología (conceptos) del dominio de instalación de software. La información necesaria para la implementación de la ontología es la que contiene el modelo de experiencia. El siguiente paso fue implementar el modelo de experiencia que es una labor sencilla, ya que sólo hay que definir los conceptos con el tipo de atributos asociados y los métodos respectivos para introducir u obtener los datos de los atributos.

Cabe destacar que en la implementación de este modelo se requiere identificar si los conceptos descritos son una acción, un predicado o simplemente un

concepto, ya que la plataforma JADE así lo requiere para implementar la ontología.

Ya implementada la ontología se retomó el modelo de coordinación. Los diagramas y plantillas presentados en este modelo facilitan estructurar el tipo de acto comunicativo (llamados por FIPA “preformativas”), y la asociación del contenido del mensaje, así como del receptor y transmisor.

Posterior al modelo de coordinación se trabajó con el modelo de comunicación que permite mostrar u obtener datos del usuario que requiera el sistema. Este modelo debe complementarse con los datos contenidos en la sección de conceptualización para la construcción de la interfaz de los agentes con la cual puede trabajarse con las tareas que se dejaron pendientes en el modelo de tareas, que a su vez permite construir los agentes. Con respecto al modelo de organización, fue consultado durante la construcción de los agentes para estructurar y conocer cómo y con qué objetos u otros agentes interactuar.

La secuencia empleada para la implementación de la metodología MAS-CommondKADS surgió de la idea de que, en cierto modo, la metodología se encontraba estructurada de dicha forma que facilitaba la recolección de datos, así como en la implementación de sistemas Multi-agentes

4.3 Propuesta para implementar el prototipo

Después de haber concluido la implementación del prototipo siguiendo la secuencia de modelos descrita en la sección anterior, se propone seguir la siguiente secuencia de modelos para la implementación de un sistema Multi-agentes con mayor eficiencia y rapidez.

Se propone iniciar la implementación del modelo de experiencia que en gran medida va a servir de motor para la colaboración entre los agentes, además de que en esta implementación no se requiere de otros modelos para ser implementado.

El modelo de comunicación que se complementa con el modelo de conceptualización, es el que debe proseguir al modelo de experiencia para la creación de las interfaces de cada uno de los agentes, ya que sobre ellos van a obtener o desplegar información al usuario. Este modelo se sugiere en la secuencia ya que no requiere a otros modelos.

El modelo de coordinación podría ser tratado ahora, ya que se tiene definida la ontología del dominio, claro que no se podrá verificar en este paso el intercambio de mensajes, ya que los agentes no han sido creados aún, pero se puede estructurar y cuando se encuentre implementado el agente sólo se retomará para ser agregado e interactuar.

Ahora el modelo de agentes es el que debe de trabajarse, este modelo se debe trabajar a la par del modelo de tareas, ya que dentro del modelo del agente se encuentran definidas las tareas específicas que debe realizar cada agente y apoyándose del modelo de tareas se obtendrán los detalles de cada tarea. El modelo de organización debe ser revisado durante la construcción de los agentes, debido a que se muestra la relación que guardan los agentes con su entorno (otros agentes y objetos).

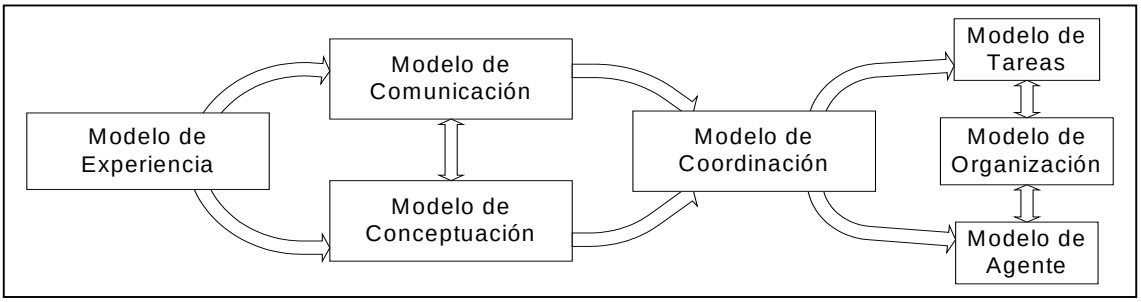

Fig. 4.1 Diagrama de secuencia de implementación.

4.4 Pruebas del prototipo

La metodología MAS-CommondKADS sugiere la realización de pruebas al sistema de agentes que se implementa, pero no existe dentro de ella una descripción o propuesta de cómo llevar a cabo las pruebas, por tanto, durante la implementación del prototipo desarrollado se realizaron dichas pruebas dentro de la implementación de cada uno de los modelos que comprende la metodología que a continuación se procede a describir.

En el modelo de experiencia del prototipo se definen la ontología (conceptos) que emplean los agentes en el dominio de instalación de software, una vez implementados los conceptos con sus respectivos métodos de insertar y obtener los atributos con que cuentan, lo que se realizó fue crear un agente con los elementos básicos para el envío de los conceptos, introduciéndoles sus atributos y a otro agente que pudiése recibirlos. Así se comprobó con cada uno de los conceptos para validar su funcionamiento. Ésto se debe realizar porque si existe un error en alguno de los conceptos, podría ser difícil encontrarle, ya que algunos conceptos son agrupados dentro de otros, según se describa en el análisis realizado.

En el caso de las pruebas realizadas en la implementación del modelo de coordinación se realizan tareas similares a las del caso del modelo de experiencia solo que aquí se tiene que realizar la construcción de las estructuras del contenido de cada uno de los mensajes y también definir que actos del habla (preformativas) van a ser válidos para cada uno de los agentes, así como la

extracción del contenido del mensaje. La prueba de este módulo consiste en alimentar con datos el contenido de los mensajes del agente transmisor y al agente receptor agregarle la estructura de los actos del habla que tiene definidos y extraer el contenido del mensaje que puede ser validado mostrando la información contenida en el mensaje que debe ser igual a la que está introduciéndose en el transmisor.

Las pruebas a la implementación del modelo de comunicación son fáciles para llevarlas a cabo, ya que se está más acostumbrado a trabajar con interfaces hombre-máquina, por lo cual hay que verificar que si ocurre un evento se acceda al método donde se encontrará el comportamiento del evento, verificar que los datos sean convertidos al tipo requerido y si no fuese así desplegar indicaciones, etc..

En lo que respecta a las pruebas del modelo de tareas estas pueden trabajarse de forma modular, para ser ejecutadas cuando se crea conveniente verificar que está realizando ciertos procedimientos. Esto es posible, ya que las tareas son actividades que tienen que llevarse a cabo sobre los elementos que envuelven el entorno de trabajo de los agentes.

Por último las pruebas a la implementación del modelo de agentes se realizan conforme se anexan los componentes que integran a cada uno de los agentes. Se tienen que realizar las pruebas del comportamiento de los agentes de forma individual. Ya que estas pruebas han sido alcanzadas, se prosigue haciendo la prueba de forma conjunta con los demás agentes, para lo cual la plataforma de agentes permite apoyarse en herramientas con que cuenta para verificar el intercambio de mensajes que existe entre los agentes y ver que respuestas se están generando. En un inicio todas las pruebas se realizan dentro de la misma computadora. Ya que las pruebas son satisfactorias, se prosigue a realizar las pruebas al prototipo en dos computadoras para verificar que el sistema de agentes tenga la comunicación en diferentes equipos.

4.5 Requerimientos y ejecución del prototipo

En la presente sección se describen los requerimientos para la ejecución del prototipo en un ambiente de trabajo. Como primer punto se menciona la necesidad de contar con una plataforma de agentes en este caso JADE y posteriormente los requerimientos propios del prototipo y por último cómo se ejecuta el prototipo del sistema.

4.5.1 Requerimientos para ejecutar JADE

El prototipo del Sistema Multi-agente para Instalación Remota de Software necesita para su ejecución tener instalada previamente la plataforma de agentes

JADE que es la que va a dar el soporte a los agentes que han sido creados para implementar el prototipo, la cual, está conformada con las referencias de la arquitectura especificadas por FIPA.

JADE necesita la máquina virtual de JAVA, que es uno de los requerimientos que demanda para su ejecución.

Para obtener la plataforma de agentes JADE puede ingresarse al sitio Web http://sharon.cselt.it de Telecom Italia Lab, que es el grupo responsable de su implementación . La versión sobre la que se ha trabajado en el proyecto es la 3.0b1.

El archivo JADE-bin-3.0b1.zip contiene las librerías necesarias para ejecutar la plataforma de agentes. Descomprimiendo el archivo se encuentra un directorio llamado “lib” que contiene 4 archivos JAR, estos archivos sirven para la ejecución del ambiente JADE. Los archivos JAR deben ser dados de alta en la variable de ambiente CLASSPATH del sistema operativo, colocando la ruta completa de la localización de cada uno de estos archivos.

Para poder levantar la plataforma de agentes JADE, ya realizado el procedimiento descrito en el párrafo anterior, tiene que ejecutarse la siguiente línea desde la consola del sistema:

java jade.Boot –gui

La instrucción presentada además de levantar la plataforma de agentes, permite con el parámetro que se ha dado (gui), solicitar la visualización de la interfaz gráfica del Administrador Remoto de Agentes (RMA), que como ya ha sido mencionado en la sección 2.4.1.3, es el agente que permite tener el control de la plataforma y por ende de los agentes que se han registrado en ella.

4.5.2 Requerimientos para la ejecución del prototipo

Para su ejecución el prototipo del Sistema Multi-agente requiere de algunos procedimientos previos como son:

- Colocar en el CLASSPATH un archivo *jbcl.jar* que emplea la interfaz gráfica y la ruta donde se encuentra el paquete del prototipo (sars).
- Dar de alta la Base de datos en el sistema operativo de la máquina donde va a residir el agente Administrador.
- Tener instalado un servidor Web, con algunos contenedores definidos, para que los agentes pueda obtener las herramientas necesarias para dar el servicio al equipo del usuario.

Para más referencias de cómo realizar estos procedimientos consulte el apéndice B (Manual de instalación del prototipo)

4.5.3 Inicialización de los agentes

Para iniciar y activar el agente Administrador que desempeña el papel de administrar el software y decidir qué agente es el que tiene que inicializar para dar un servicio, es necesario levantar la plataforma de agentes, en la máquina donde se alojará el software que empleará el prototipo para la instalación de software. Este procedimiento ha sido descrito en la sección 4.5.1. Ya en ejecución la plataforma de agentes, se puede iniciar el agente de dos formas: la primera es utilizando la interfaz gráfica de la plataforma donde hay que indicar que se va a crear un nuevo agente, se desplegará una ventana donde se pide introducir el nombre que se dará al agente y el nombre de la clase del agente junto con el paquete dónde está contenida la clase. La segunda forma es emplear en el símbolo del sistema la siguiente instrucción:

java jade.Boot –container administador:sars.agentes.Administrador

donde:

- *jade.Boot.*- Es la instrucción que está haciendo referencia a la plataforma de agentes.
- -*container*.- Especifica que esta instancia es un contenedor de JADE, y como tal, ésto puede unirse con el contenedor principal donde se alojan los agentes propios de JADE.
- *administrador*.- Es el identificador del agente.
- *sars.agentes.Administrador*.- Es el nombre de la clase del agente que se encuentra contenido en el paquete *agentes*, que a su vez está contenido en el paquete *sars*.

Para ejecutar el agente Asistente que debe alojarse en la máquina del usuario se ha creado un archivo ejecutable que iniciará al agente y lo registrará en la plataforma de agentes que aloja al prototipo. Básicamente, este ejecutable realizará los pasos descritos en la inicialización del agente administrador con la adición del parámetro:

- host dirección

donde:

- *host*.- hay que indicar la dirección de la máquina donde se encuentra la plataforma de agentes.

Con respecto a la inicialización del agente instalador, éste será iniciado por el agente Administrador, cuando reciba una solicitud de servicio de instalación de alguna de las aplicaciones que se encuentren registradas en su base de datos.

Para la inicialización de la plataforma de agentes junto con el agente administrador se tiene al igual que en el caso de la inicialización del agente

asistente, un archivo ejecutable que el usuario ejecutará y levantará estos dos componentes, sólo que aquí se indicó las alternativas que pueden realizarse para ejemplificar cómo se crea un agente dentro de la plataforma, y así ilustrar el contenido del archivo ejecutable.

4.6 Descripción de las clases de agentes del prototipo

En esta sección se han colocado diagramas de paquetes que ilustra algunas de las clases, por lo cual primero se da una descripción de los paquetes que conforman al prototipo.

Paquete sars.agentes

Este paquete contiene la implementación de los agentes del prototipo (Asistente, Administrador e Instalador), que corresponde al análisis descrito en el modelo de agentes de la metodología.

Paquete sars.agentes.gui

Paquete que contiene los componentes que integran la interfaz gráfica de los agentes. La implementación de este paquete corresponde al análisis explicado en el modelo de comunicación.

Paquete sars.agentes.tareas

Contiene las tareas que desarrollan los agentes en su entorno, corresponde a la descripción del modelo de tareas.

Paquete sars.contenido.ontologia

Contiene la ontología necesaria para el entendimiento entre los agentes. El paquete pertenece a los datos expuestos en el modelo de experiencia.

Paquete sars.agentes.comportamientos

Contiene las clases que permiten a cada uno de los agentes manipular la información que es transmitida y recibida para llevar la comunicación. Es equivalente a la información del modelo de coordinación.

4.6.1 Agente Asistente

El agente asistente como se ha mencionado, es el que se va a encontrar físicamente en la máquina del usuario. Va a contar con una interfaz gráfica, en la cual el usuario va a suministrar la información necesaria para que el sistema le brinde un servicio.

El paquete *sars.agentes* contiene la clase del agente Asistente, que como se observa en la figura 4.1 tiene los atributos: 1) *GuiAsistente* que va a ser una instancia de la clase con el mismo nombre encontrada en el paquete *sars.agentes.gui*, para obtener y manipular la interfaz gráfica con el usuario. 2) *ReporteDeSolicitud* que contiene la información de la solicitud de servicio que va a enviar al agente Administrador. También en la clase se observan las acciones del agente, que son: 1) *setup*, inicializa al agente, registrando la ontología, el lenguaje que utiliza, inicia la interfaz con el usuario y fija su comportamiento. 2) *onGuiEvent*, maneja los eventos generados en la interfaz del agente. 3) *solicitarAplicaciones*, lo que hace es entablar una comunicación con el agente para obtener la información de las aplicaciones con que puede interactuar el sistema.

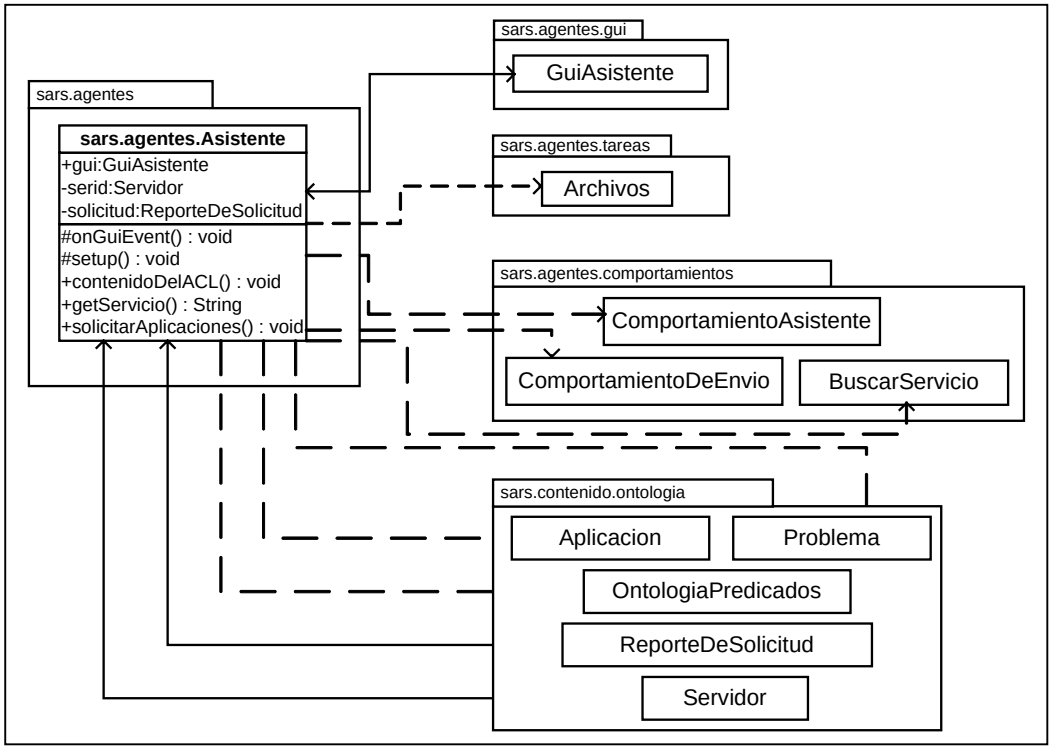

Fig. 4.2 Diagrama de paquetes que interactúan la clase Asistente

4.6.2 Agente Administrador

El agente administrador, es el que se va a encontrar físicamente en la máquina del técnico, cuenta con una interfaz gráfica, en la cual se administra el software con que interactúa el prototipo del sistema de instalación, además de ser quien inicia a los agentes que pueden proporcionar los servicios a la máquina del usuario.

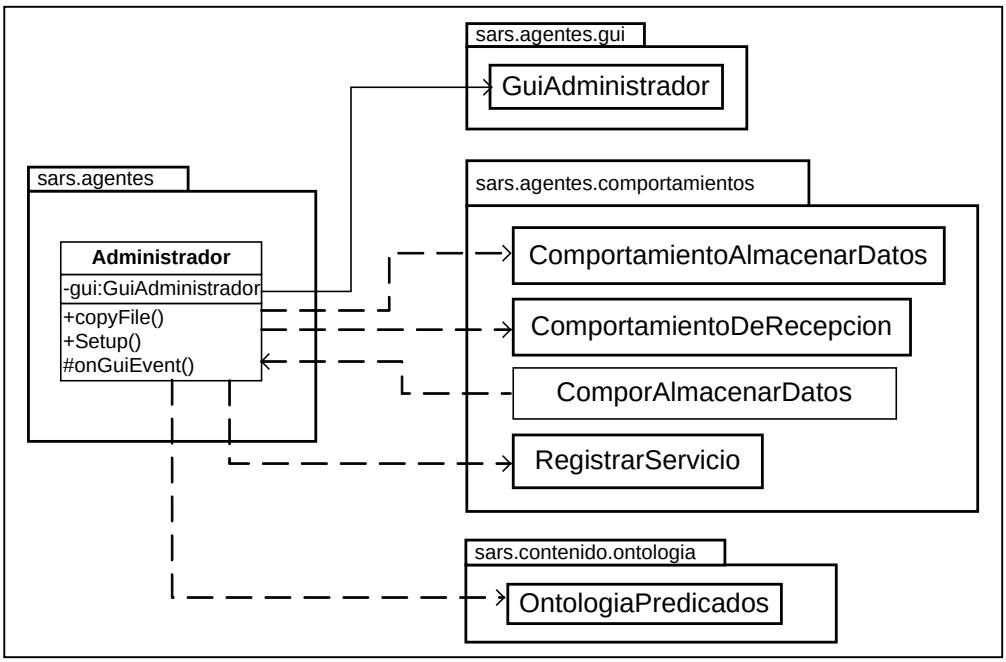

Fig. 4.3 Diagrama de paquetes que interactúan la clase Administrador

Al igual que en el caso de la clase Asistente, la clase Administrador se encuentra en el paquete *sars.agentes* ilustrado en la figura 4.2, esta clase contiene un atributo *GuiAdministrador*, que es una instancia de la clase con el mismo nombre contenida en el paquete *sars.agentes.gui*, con la cual el técnico va a interactuar con el agente. La clase también cuenta con acciones como son: 1) *setup*, registra los servicios que puede prestar el sistema, la ontología y lenguaje que empleará el agente propios del sistema, así como el de movilidad, fija su comportamiento para la recepción de solicitudes e inicia la interfaz con el técnico. 2) *onGuiEvent*, al igual que en la clase anterior permite manejar los eventos suscitados en la interfaz. 3) *copyFile*, permite copiar el software al contenedor donde los agentes que proporcionan los servicios van a tomarlo.

4.6.3 Agente Instalador

El agente instalador, es el que físicamente es iniciado en la máquina del técnico y después de la comunicación con el agente administrador se traslada a la máquina del usuario para instalar el software que ha sido solicitado.

El paquete mostrado en la figura 4.3 ilustra la clase Instalador contenida dentro del paquete *sars.agentes*, la cual cuenta con el atributo *EjecutarAplicacion* y también cuenta con las acciones: 1) *setup*, fija el comportamiento de recepción de las instrucciones cuando se encuentre en otra máquina. 2) *ubicaciónDeSoftware*, obtiene de la base de datos la información del software que va a manipular. 3) *init*, registra el lenguaje y la ontología de movilidad del agente. 4) *instalar*, ejecuta la aplicación a ser instalada. 5)*copiarInstalador*, se conecta al contenedor y toma el software que va a utilizar para dar el servicio. 6) *afterMove*, acción que tiene contenidas las actividades que va a realizar el agente cuando llegue a la máquina destino. 7) *beforeMove*, contiene el comportamiento que debe realizar el agente antes de trasladarse a la máquina destino.

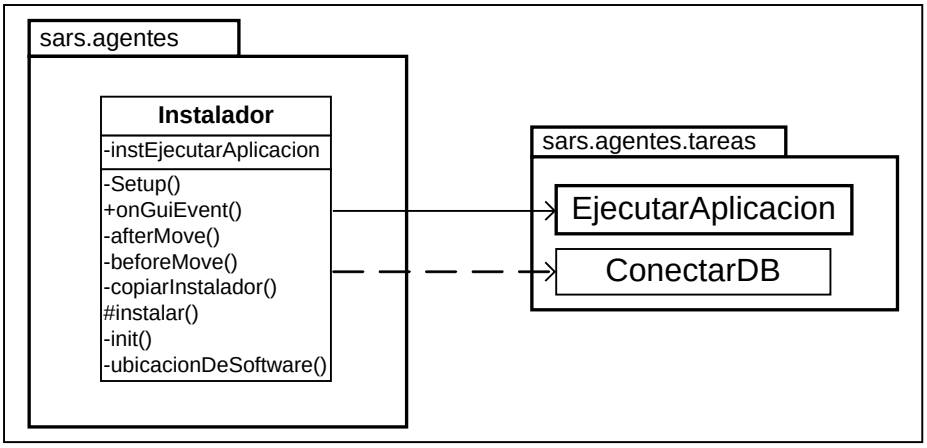

Fig. 4.4 Diagrama de paquetes que interactúan la clase Instalador.

4.7 Implementación de los modelos de la metodología

Para ejemplificar cómo se lleva a cabo la inicialización de un agente en JADE, en este caso específico del agente Administrador, se ha colocado el método *setup*, donde los agentes inician sus comportamientos y se registran con la plataforma. Este fragmento además ilustra que se cuenta con un interfaz y registra los servicios que se encuentran descritos en el modelo de agentes, sección 3.4.6

```
public void setup(){

  //Registrar servicio
  String servicio[] = {"Instalacion", "Configuracion","Ejecutar"};
  RegistrarServicio serv = new RegistrarServicio(this, servicio, "software" );

  //Registro del Lenguaje y la Ontologia empleadas
  manager.registerOntology(ontology);
  manager.registerLanguage( codec );

  managerMov.registerOntology(ontologyMov);
  managerMov.registerLanguage( codecMov );

  System.out.println("[" + getLocalName() +
                  "] creación del agente Administrador");

  //Agrega el comportamiento para la recepcion de solicitudes
  addBehaviour( new ComportamientoDeRecepcion( this ) );

  //Inicia la interfaz del agente
  gui = new GuiAdministrador("[Administrador] Registrar Nuevo Software", this);
}
```

El siguiente fragmento de código muestra que pasar del análisis a la implementación es una labor sencilla en el caso del modelo de experiencia, donde sólo hay que crear los métodos para colocar y obtener las propiedades del concepto descrito. El código muestra la implementación del concepto Software que posee los atributos procesador, os, ram, dd, y video. Para más detalles ver la sección 3.6.2.

```
public Software( String software, String procesador, String os, int ram, int
dd, String video ) {
    setNombre( software );
    setProcesador( procesador );
                                        .
                                        .
                                        .
}

  public void setNombre( String software ){
    this.nombre = software;
  }

  public void setProcesador( String procesador ){
    this.procesador = procesador;
  }
                                        .
                                        .
                                        .
   public String getNombre(){
     return this.nombre;
   }

   public String getProcesador(){
     return this.procesador;
   }
```

El fragmento de código que se muestra a continuación pertenece a la conversación *Comunicar problemática* contenida en el modelo de coordinación. Donde se puede ver que se está estructurando el mensaje con los datos del agente que envía, el receptor, el lenguaje, la ontología y el contenido del mensaje, además del tipo de acto de comunicación, en este caso una petición.

```
//CREA UN MENSAJE DEL TIPO REQUEST
  msg = new ACLMessage(ACLMessage.REQUEST);

  try {
    //FIJA LOS ELEMENTOS DEL ACL:
    msg.setSender(getAID());//FIJA AL TRANSMISOR
    msg.addReceiver(receptor);//FIJA AL RECEPTOR
    msg.setLanguage(codec.getName());//FIJA LENGUAJE
    msg.setOntology(ontology.getName());//FIJA ONTOLOGIA
  }
  catch(Exception e) {
    e.printStackTrace();
  }
  //AGREGA EL COMPORTAMIENTO DE ENVIO DE LOS DATOS NECESARIOS PARA LA
SOLICITUD DE UN SERVICIO
  addBehaviour( new ComportamientoDeEnvio( this, msg, solici ) );
```

El siguiente fragmento de código ilustra la tarea *ElaborarCadenaSolicitudDeServicio*, correspondiente al modelo de tareas de la sección 3.3.1. Se puede observar que se obtienen los datos del usuario contenidos en la interfaz gráfica y también por medio del método *getDatos* se obtienen los datos del problema que presenta la máquina del usuario, para colocarlos en el objeto del tipo *ReporteDeSolicitud*, los cuales son pasados al agente por medio del método *postGuiEvent.*

```
  GuiEvent ev = new GuiEvent(this,miAgente.ACEPTAR);
  String datosDeUsuario[] = {
    datos.getNombre(),      //Guardando Nombre de Usuario
    datos.getClave(),       //Clave del Usuario
    datos.getExtension(),   //Extension telefonica
    datos.getGerencia()     //Gerencia a la que pertenece
  };

  //Si los datos obtenidos no corresponden manda un mensaje y termina el envio
  ReporteDeSolicitud datos = getDatos();

  miAgente.doActivate();

  //Instruccion para esconder la ventana
  hide();
  //Fija los parametros en el Evento de la GUI
  ev.addParameter(datosDeUsuario);
  ev.addParameter(datos);
  miAgente.postGuiEvent(ev);
```

La ilustración de la implementación del modelo de comunicación se ha omitido, ya que esta parte no presenta interés alguno, debido a que se implementa de forma similar al software tradicional, la única diferencia radica en el paso de la información de la interfaz al agente, lo cual se puede ver en el fragmento de código anterior que pertenece a la interfaz del agente Asistente que por medio del método *postGuiEvent* pasa los datos que el agente procesará.

CAPÍTULO 5
Conclusión y Futuros trabajos.

5.1 Conclusión

En este trabajo de investigación “Prototipo de un Sistema Multi-agente para Instalación Remota de Software”, se ha proporcionado en una primera etapa los antecedentes de la problemática detectada en la Gerencia de Tecnología Informática del Instituto Mexicano del Petróleo, para después tomar lo que es la tecnología de agentes donde se proporciona una breve historia, la teoría, las arquitecturas y los lenguajes de agentes. Posteriormente se realizó un estudio de las metodologías y herramientas existentes para la construcción del prototipo de un Sistemas Multi-agentes, donde se valoraron diversos aspectos para la elección de la metodología y la herramienta con la cual se desarrolló el prototipo. A partir de ésto se documentaron los requerimientos del sistema empleando la metodología MAS-CommonKADS. Después, con la ayuda de la documentación generada, se implementó con la plataforma de agentes JADE.

Concluyendo con el enfoque de agentes se notó que permite modelar de mejor forma a individuos de una organización y los papeles que desempeñan dentro, ya que además de realizar sus propias labores permite establecer una comunicación con otros agentes y así negociar la colaboración para resolver tareas o problemáticas, mismas actividades que realiza un individuo en la organización.

Con respecto a la herramienta que se seleccionó para la implementación del prototipo se cree que ha sido una buena elección JADE, ya que cuenta con recursos que permiten monitorear la sociedad de agentes, las comunicaciones y

el contenido de los mensajes, también permite enviar mensajes desde una herramienta con que cuenta y así comprobar los comportamientos que tienen los agentes según los diversos actos comunicativos existentes, cuenta con una interfaz que permite verificar los servicios que han sido registrados en la plataforma. Además JADE cumple con las normas dictadas por la FIPA, y es un proyecto que lo integran tres miembros importantes del área de las telecomunicaciones como son TILAB, Motorola y Whitestein Technologies AG.

En lo referente a la implementación del prototipo en un sistema Multi-agente, podrá verse sobrada la tecnología de agentes al momento, pero se piensa que ha sido una buena opción ya que el modelado que utiliza facilita la identificación de los requerimientos al analista dado que se preocupa en los objetivos que deben alcanzar y las tareas necesarias para conseguirlo, que de alguna forma es como razonamos. Además sólo se presenta el prototipo que no cuenta con la parte que permite evaluar el estado de las aplicaciones para así interactuar con ellas, pero cuando sea implementado el sistema totalmente, la perspectiva será diferente, ya que contará con módulos para tomar decisiones dependiendo de los problemas que se encuentre al intentar reparar la configuración de alguna aplicación. Y cómo es una tecnología que relativamente tiene poco tiempo en que se está trabajando, aún existe mayor posibilidad de explotar sus capacidades y en versiones futuras no se verá sobrepasada la tecnología para realizar la implementación del sistema.

Con la presentación del prototipo de agentes para la instalación remota de software se cree que es factible la implementación del sistema de forma total en el área administrativa del Instituto, ya que al momento el prototipo tiene implementado el servicio de instalación de software llevándolo a cabo en equipos remotos, lo que reducirá el tiempo de respuesta a la prestación de este servicio debido a los inconvenientes tanto del traslado del personal como de su disponibilidad. Siendo este el punto por el cual se pensó en la creación de un sistema que resuelva los puntos citados. Además se disminuirá la carga de trabajo de la Gerencia de Tecnología Informática ya que su personal dedicaría mayor parte de su tiempo a las labores competentes de la Gerencia.

5.2 Trabajos futuros

El trabajo realizado puede continuarse realizando la implementación de los componentes que no fueron implementados en el prototipo, como es el caso de los agentes:

Secretario.- Debe desempeñar el papel de administrador de los documentos o reportes que está generando el sistema.

Supervisor.- Tendrá por función el monitoreo de los servicios que está realizando el sistema, desde que el usuario ha solicitado uno hasta su culminación, así como la generación de estadísticas.

Ejecutor.- Interactuará con algunas aplicaciones como son el Anti-virus y software que permita reparar problemas en la máquina del usuario (Scandisk y desfragmentador de archivos).

Configurador.- Realizará la acción de interactuar con el software para personalizarlo según el usuario.

Restaurador.- Manipulará los registros del sistema operativo para corregir problemas que presenten las aplicaciones instaladas, así como crear accesos directos a aplicaciones cuando el usuario las haya borrado accidentalmente.

Además de agregar los agentes mencionados para completar así el sistema, se puede mejorar algunas labores de los agentes con que cuenta el prototipo, como son los casos de:

El agente instalador donde puede agregarse un módulo para que interactúe directamente con la aplicación que está trabajando, y así el usuario no introduzca datos erróneos o falsos.

Que el agente asistente permita la recolección de más datos para que diagnostique que servicio debe solicitar y si se tratáse de un problema que necesite reconfigurar algunos datos inicie al agente Restaurador para solucionar los problemas.

Se podría integrar el sistema de instalación de software vía remota con un Sistema Multi-agentes que permita configurar el hardware y así integrar los dos componentes de una máquina (lógicos y físicos) en la solución de problemas (actualmente se está desarrollando un proyecto al respecto).

Anexo A
Agent UML

A.1 Introducción

En este apartado se presenta un resumen del Agent-UML que es una extensión del UML y es un lenguaje gráfico de esquemas que permite una visión intuitiva del intercambio de mensajes entre entidades que pueden comunicarse y su entorno. En primer termino se mencionan los antecedentes del Agent-UML y aspectos generales (sección A.2), posteriormente se tratan los diversos tipos de diagramas empleados por el Agent-UML (sección A.3).

A.2 Antecedentes del Agent-UML

El UML [32] es un Lenguaje de Modelado Unificado basado en una notación gráfica, la cual permite: especificar, construir, visualizar y documentar los objetos de un sistema programado. Este lenguaje es el resultado de la unificación de los métodos de modelado orientados a objetos de Booch (Booch Method), Rumbaugh (OMT: Object Modeling Technique) y Jacobson (OOSE: Object-Oriented Sotfware Engineering).

El objetivo del *UML* es organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras persona involucradas en el desarrollo del sistema lo comprendan y convengan con él. El UML proporciona tal organización, por tal motivo los organismos encargados de administrarlo OMG (Object Management Group) y la FIPA (Foundation of Intelligent Phisical Agents),

han cooperado tratando de incrementar la aceptación de la tecnología de agentes en la industria relacionándolo con un estándar de facto y apoyando el ambiente de desarrollo a lo largo del ciclo de vida del sistema.

Como primer resultado de esta cooperación, han proporcionado el Agent-UML [33] [34], una extensión del UML.

El Agent-UML puede ser utilizado por cualquier metodología de análisis y diseño orientado a agentes para expresar los diseños, por tanto como se mencionó en la tesis, se han utilizado algunos diagramas del Agent-UML en la metodología MAS-CommonKADS, para modelar de forma clara e intuitiva el análisis y diseño del sistema, sobre la propuesta realizada por la metodología con los diagramas MCS.

A.3 Diagramas del UML con extensión a agentes (A-UML)

El UML unifica y formaliza los métodos de muchas formas para el ciclo de vida del software orientado a objetos y soporta los siguientes tipos de modelos, a los cuales, se les agrega la extensión para la creación de sistemas basados en agentes.

- Modelos estáticos: tal como Diagramas de clases y de paquetes, los cuales describen la semántica estática de datos y mensajes.

  Diagramas de Clases

  Objetivo

  - Modelar las clases participantes.
  - Identificar atributos.
  - Identificar operaciones
  - Mostrar las relaciones estáticas entre clases.

  Diagrama de Paquetes

  Su objetivo es agrupar diversos tipos de:

  - Módulos
  - Modelos
  - Subsistemas
  - Grupos Físicos, etc.

  La extensión del UML para la creación de agentes contempla que un paquete despliegue dentro de un cuadro dividido por tres líneas horizontales tres categorías: 1) papel que desempeñan los parámetros, 2) condición de activación y 3) acciones de comunicación.

Otro agregado que tienen estos diagramas es cuando el paquete sobre el que se trabaja presenta la propiedad de servir como interfaz, para lo cual, se tiene que indicar con una caja que enlaza al paquete indicando la propiedad de la interfaz.

- Modelos Dinámicos incluyen a los diagramas de interacción (diagramas de colaboración y secuencia), de estados y de actividades.

Diagramas de Colaboración

Objetivo:

- Mostrar un grupo de objetos y ligas describiendo un escenario.
- Mostrar los eventos (mensajes) pasados entre los objetos.
- Mostrar el orden relativo de los eventos.

Básicamente, este tipo de diagramas no cambia en su estructura, ya que sólo se muestra cómo están colaborando los agentes. Los diagramas de colaboración describen un modelo de interacción entre agentes, la secuencia de interacciones debe ser numerada.

Diagramas de Secuencia

Objetivo:

- Mostrar las operaciones (mensajes) entre objetos en secuencia.
- Mostrar los valores de retorno.

La extensión recomendada soporta hilos concurrentes de interacción. Mientras que los hilos concurrentes no son prohibidos en la programación orientada a objetos, ellos no son comúnmente empleados. La siguiente figura describe tres modos de expresar múltiples hilos. La figura 2(a) indica que todos los hilos (desde CA-1 a CA-n) son mandados concurrentemente. La figura 2(b) incluye una caja de decisión indicando que decidirá qué hilo o hilos ejecuta (puede o no ejecutar alguno). Si más de uno es enviado, la comunicación es concurrente. La figura 2(c) indica una OR exclusiva, esto es exactamente que un solo hilo será ejecutado.

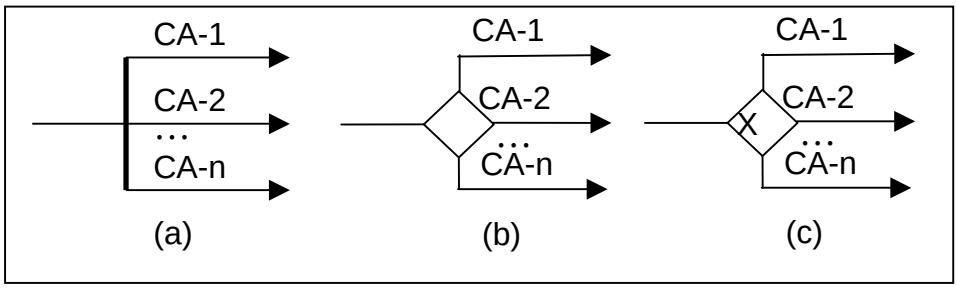

Fig. A1. Hilos concurrentes de interacción.

Diagramas de Estados

Objetivos:

- Mostrar los estados, los eventos y las transiciones.
- Describir los ciclos de los objetos.
- Identificar operaciones relacionadas con estados y eventos.

Un diagrama de estados es una gráfica que representa un estado de la máquina.

Diagramas de Actividades

Objetivo:

- Mostrar las relaciones temporales entre diversas actividades.
- Mostrar el flujo de trabajo entre casos de uso.
- Identificar paralelismo.

Los diagramas de actividades expresan operaciones y eventos que activan a los agentes, describen un orden del protocolo de procesamiento entre agentes. Los diagramas de actividades difieren de los demás diagramas de interacción, porque proveen un hilo explícito de control y proveen una representación gráfica que hace posible procesos simples, facilitando el diseño y comunicación de los modelos de comportamiento. También pueden representar procesamiento concurrente y asíncrono. Por último, pueden expresar comunicación simultánea con varios agentes.

- Casos de Uso. Especifícan las acciones que un sistema o clase pueden desempeñar con actores externos.

Diagramas de Casos de Uso

El Objetivo de estos diagramas es:

- Ilustrar los roles tomados en una situación por los actores.
- Modelar los procesos que un actor requiere de un sistema desde su propia perspectiva.

- Modelos de implementación tal como modelos de componentes y diagramas de diseño que describen la distribución de componentes en diferentes plataformas.

Diagramas de Distribución

Objetivo:

- Mostrar la arquitectura de distribución.
- Mostrar los nodos de hardware: procesadores y/o dispositivos.
- Mostrar las conexiones físicas entre los nodos.

La movilidad es una importante propiedad de los agentes. Una manera para indicar la trayectoria de movilidad y lugar de partida, es emplear la extensión para agentes del UML, consistente en un arco que parte de un nodo origen a uno destino con una etiqueta que indica la propiedad de movilidad.

Diagramas de Componentes

Objetivos:

Representar paquetes físicos:

- Componentes reutilizables.
- Archivos fuente.
- Archivos ejecutables.
- Librerías, etc.

A.4 Diagramas del MAS para Instalación Remota de Software

A continuación se presentan algunos diagramas del sistema de agentes que no fueron incluidos en el capítulo 3, donde se encuentra la documentación del análisis.

Las figuras A2 y A3 muestran los diagramas de casos de uso de la secretaria y del supervisor respectivamente, correspondientes al modelo de Conceptuación. Para conocer los detalles de cada uno de los casos de usos hay que remitirse a la sección 3.2.4.1 (Descripción textual de los casos de uso).

También se colocan algunos de los diagramas de secuencia del modelo de conceptuación referentes a la descripción de las interacciones de los casos de uso. No se han colocado todos los diagramas debido a que en algunos sólo cambia el mensaje de la acción que se requiere.

En la figura A4 se presenta el diagrama de secuencia referente al caso de uso *Crear características de software*, donde se puede ver cómo interactúa el sistema con el técnico. Básicamente lo que se pretende es presentar cómo se tiene en mente la interacción entre el sistema y el usuario.

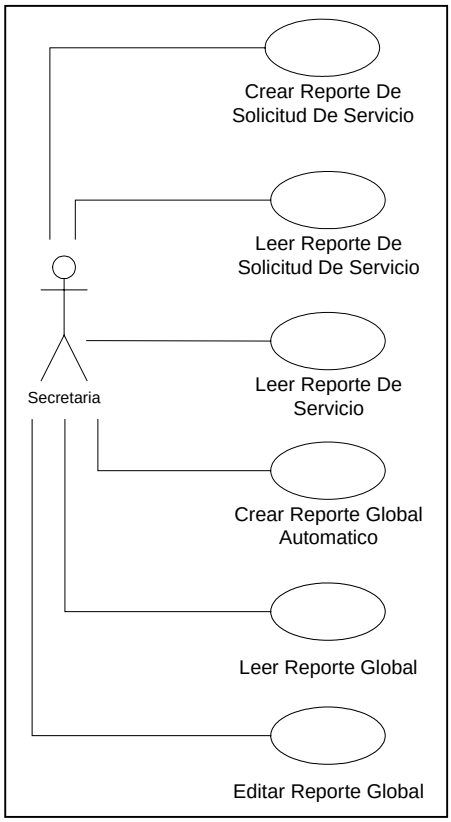

Fig. A2 Caso de Uso de la secretaria

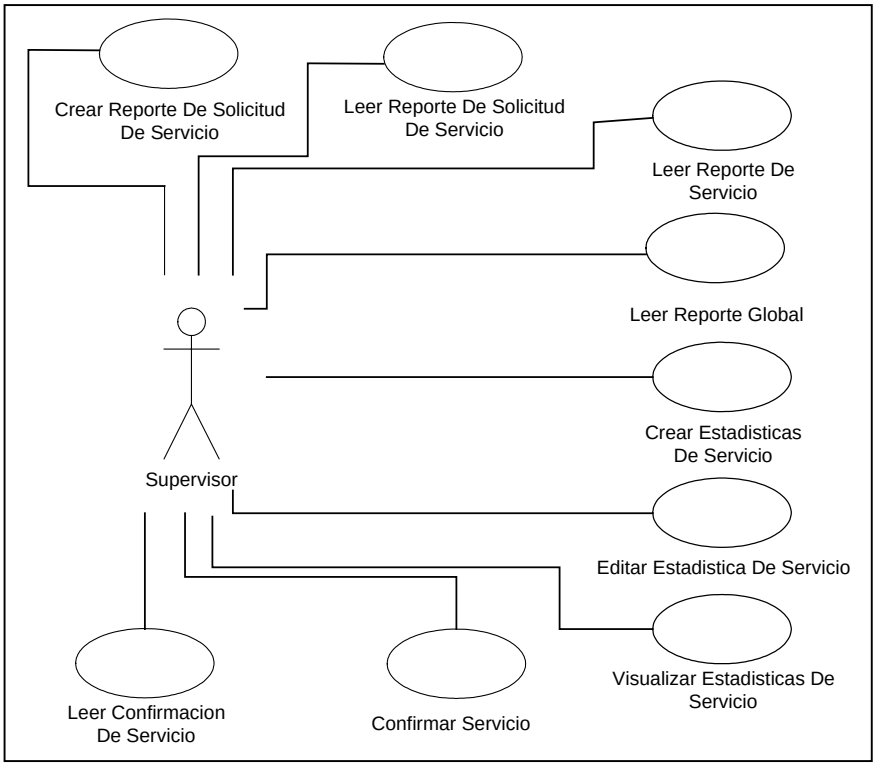

Fig. A3 Caso de uso del Supervisor

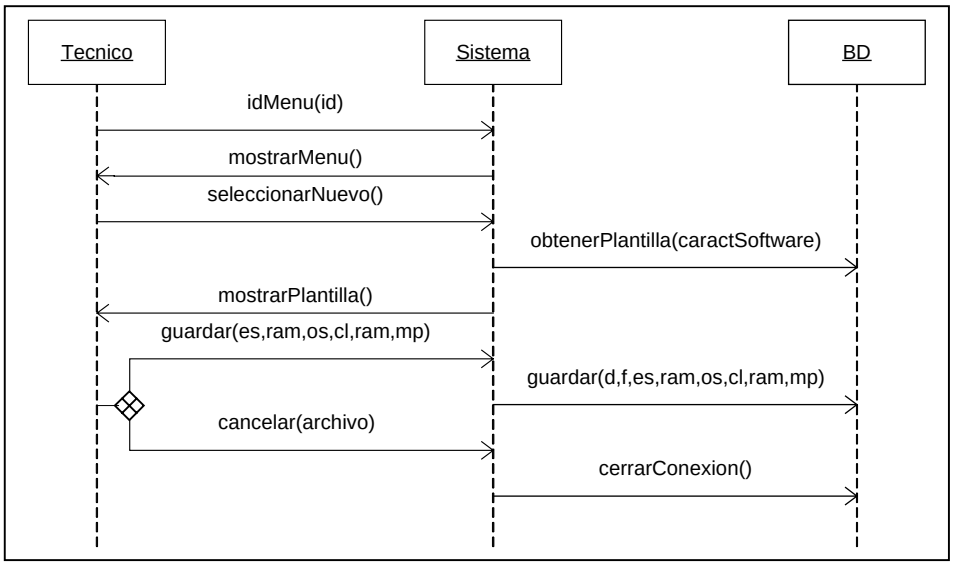

Fig. A4 Crear características de Software.

El siguiente diagrama muestra la secuencia de mensajes que se han detectado cuando el usuario (técnico) requiere editar las características de algún software que previamente ha sido declarado en el sistema. Para el caso de uso *Eliminar las Características de Software*, los mensajes intercambiados básicamente son los mismos, por lo cual, no se ha colocado este diagrama.

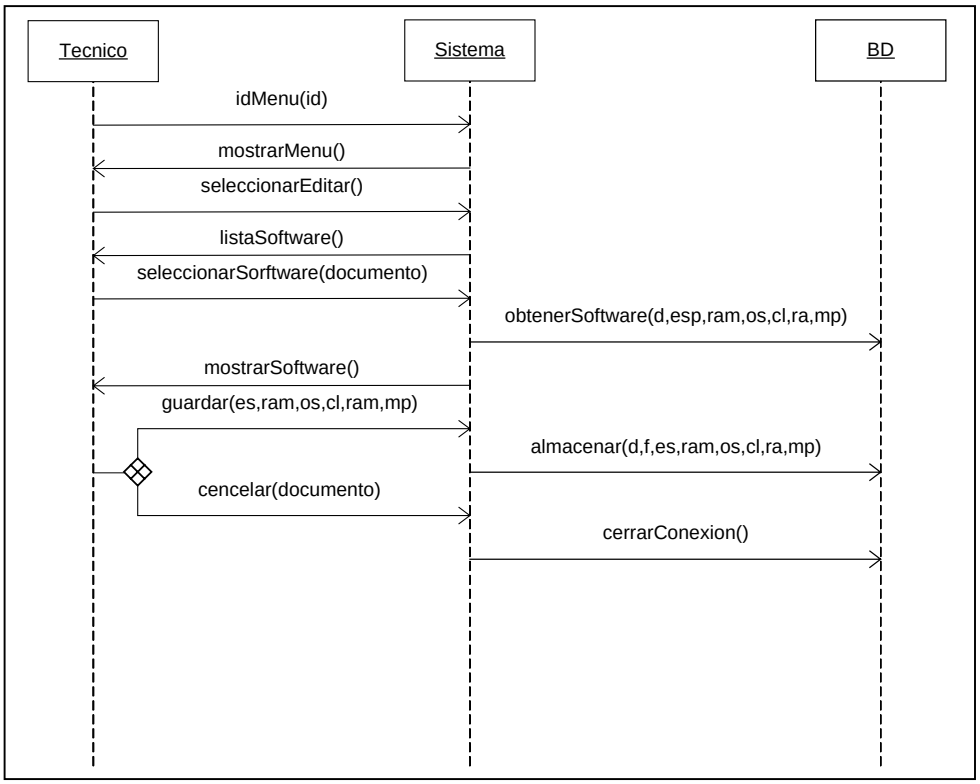

Fig. A5 Editar características de software

La figura A6 muestra el intercambio de mensajes requeridos para *crear el reporte global automático* donde el reporte se crea de acuerdo a un rango de fechas definidas por el usuario previamente.

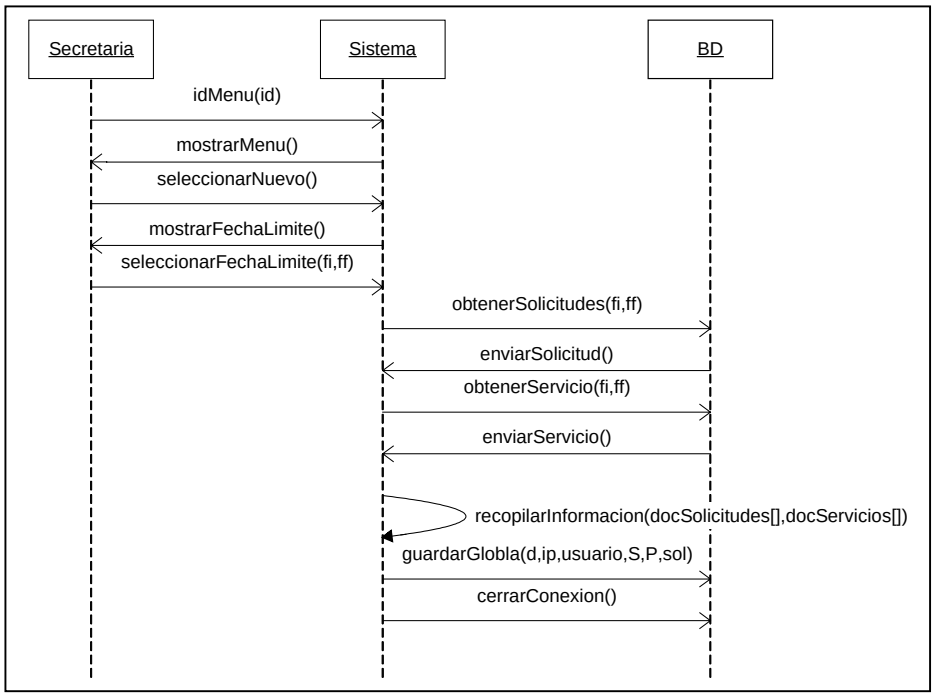

Fig. A6 Crear reporte global automático.

La figura A7 presenta concretamente la secuencia de mensajes para *editar el reporte de servicio*, pero también ilustra de una manera adecuada la edición de los demás tipos de reportes así como su lectura.

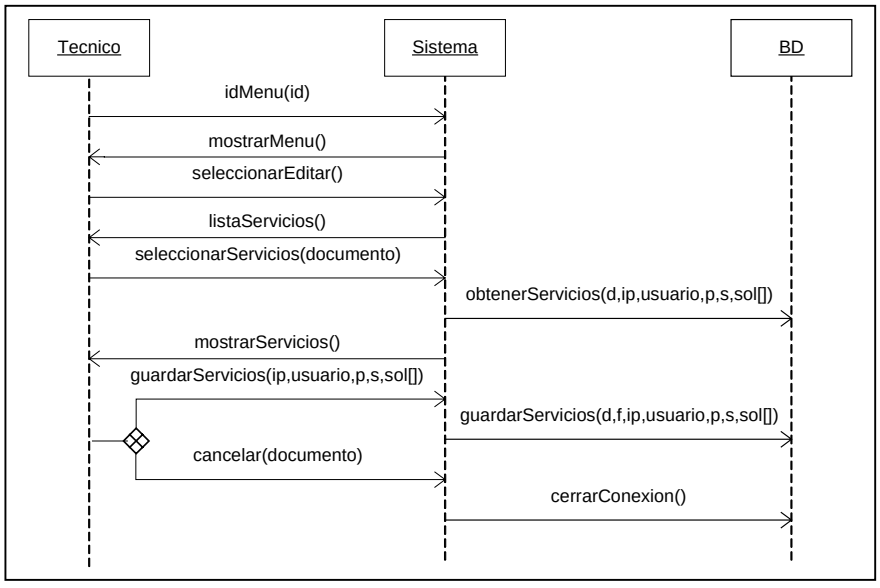

Fig. A7 Editar reporte de servicio

La figura A8 muestra el intercambio de mensajes para el caso de uso *crear estadística de servicio*.

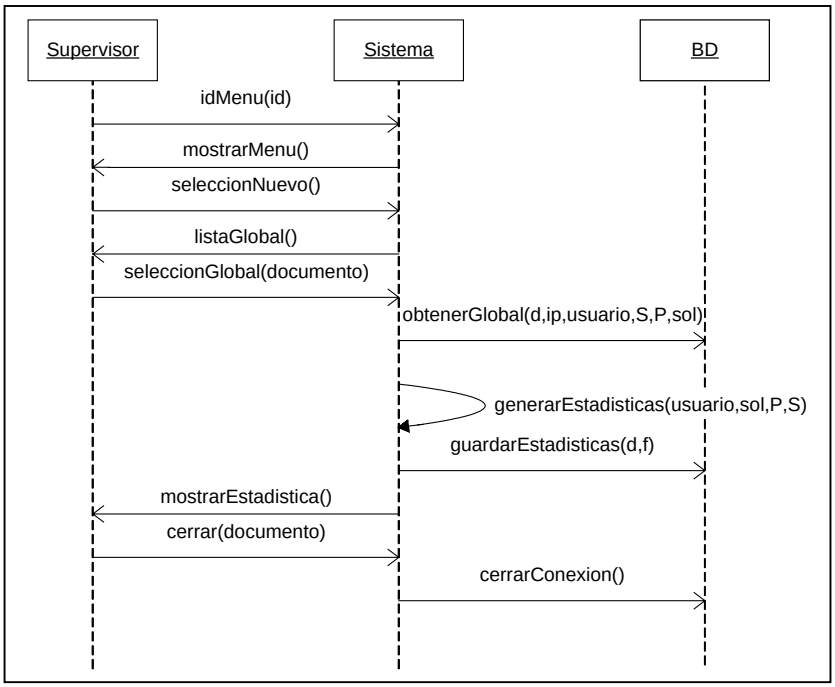

Fig. A8 Crear Estadísticas de servicio.

La figura A9 muestra el intercambio de mensajes requerido para el caso de uso *editar estadísticas de servicio*. Este diagrama muestra una descripción similar a la del caso *visualizar estadística de servicio*, por lo cual no ha sido incluido.

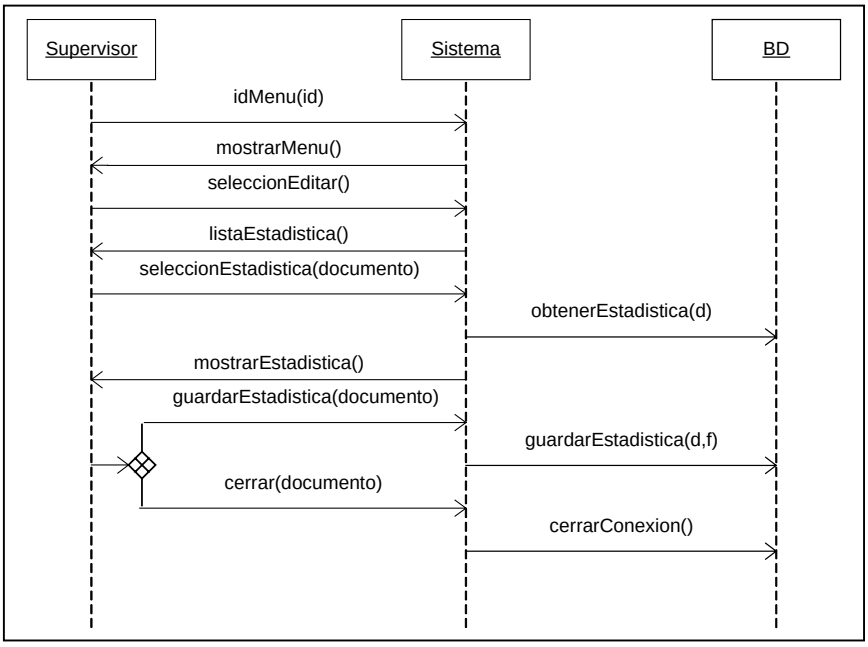

Fig. A9 Editar estadísticas de servicio.

La figura A10 muestra el intercambio de mensajes para el caso de *confirmación de servicio*, donde el usuario tiene que responder al cuestionamiento de sí se ha atendido su requerimiento de servicio.

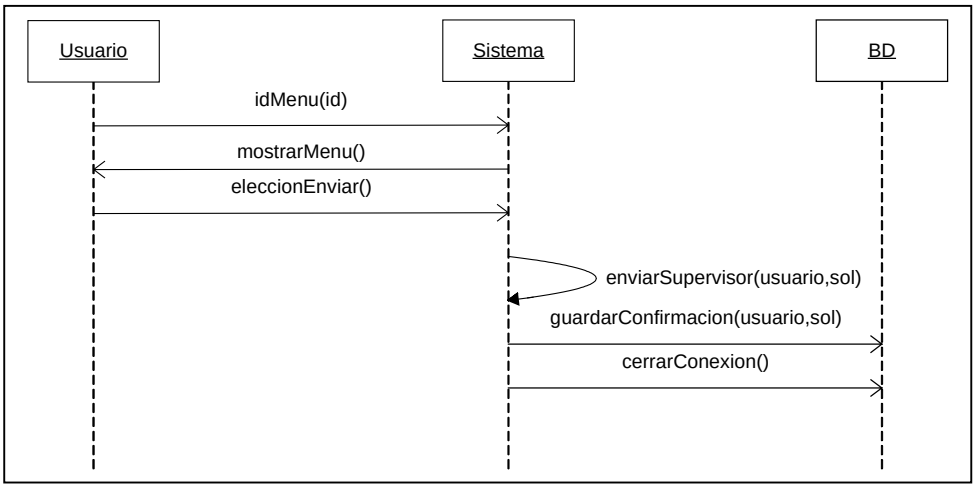

Fig. A10 Confirmación de servicio.

Los dos diagramas que se muestran a continuación pertenecen al modelo de coordinación de la metodología MAS-CommondKADS. Estos diagramas se desprenden del diagrama de la figura 3.7 que muestra los mensajes que intercambian los agentes.

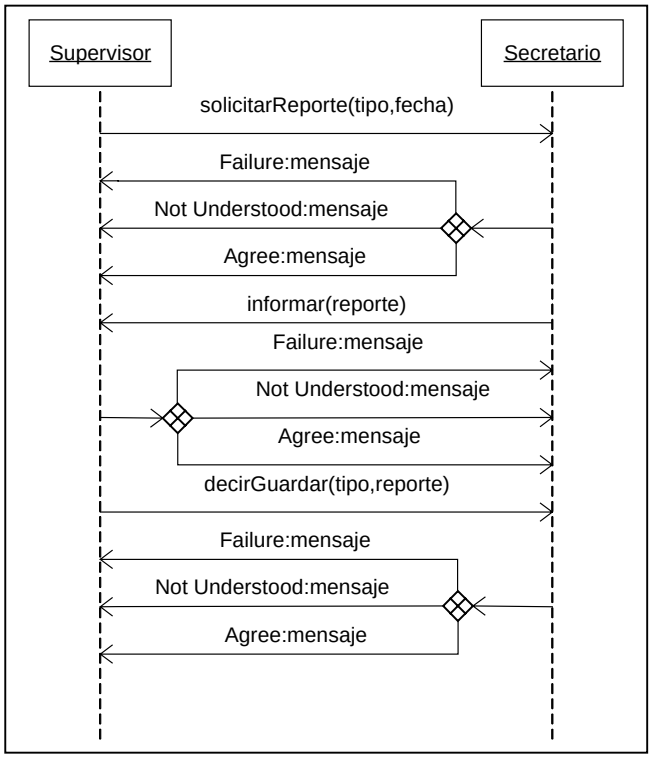

Fig. A11 Intercambio de reporte.

El diagrama de la figura A11 muestra la secuencia de mensajes intercambiados entre los agentes supervisor y secretario, donde el supervisor solicita un reporte especificando la fecha y el tipo de reporte. Si el secretario comprende el mensaje, éste envía el reporte. Cuando el supervisor termina su lectura, se comunica nuevamente al secretario para que cierre la conexión.

El diagrama mostrado en la figura A12 ilustra el intercambio de mensajes entre los agentes que realizan el servicio en la máquina del usuario, con el agente secretario (encargado de administrar los reportes), donde le informan el problema presentado, si ha sido resuelto el problema, las acciones realizadas y los resultados obtenidos.

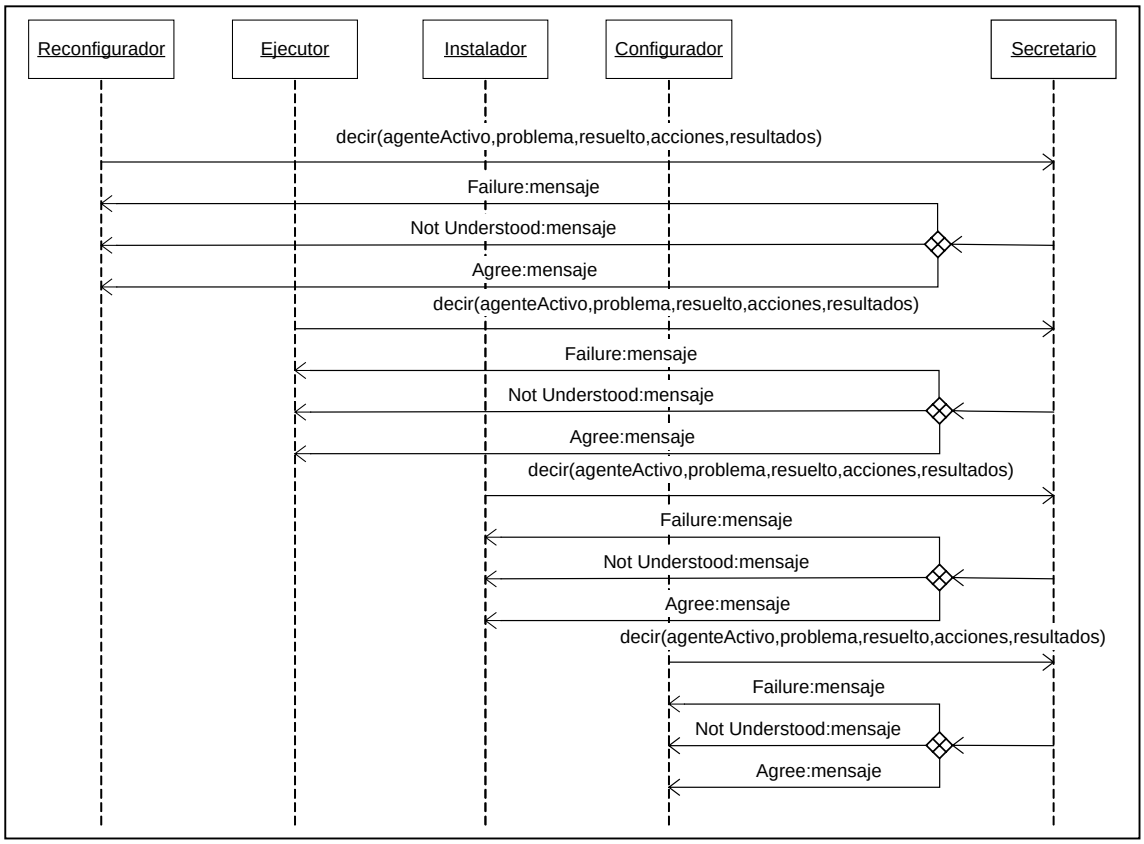

Fig. A12 Enviar datos de servicio.

Por ultimo, se presenta el diagrama de componentes(Fig. A13), que ilustra como se encuentra la distribución física de los agentes, distinguiendo dos tipos de equipos: un servidor y un pc. Además se puede observar que dentro del servidor se encuentran los agentes: Secretario, Administrador, Supervisor, Instalador, Configurador y Ejecutor; y en la pc: Asistente y Restaurador.

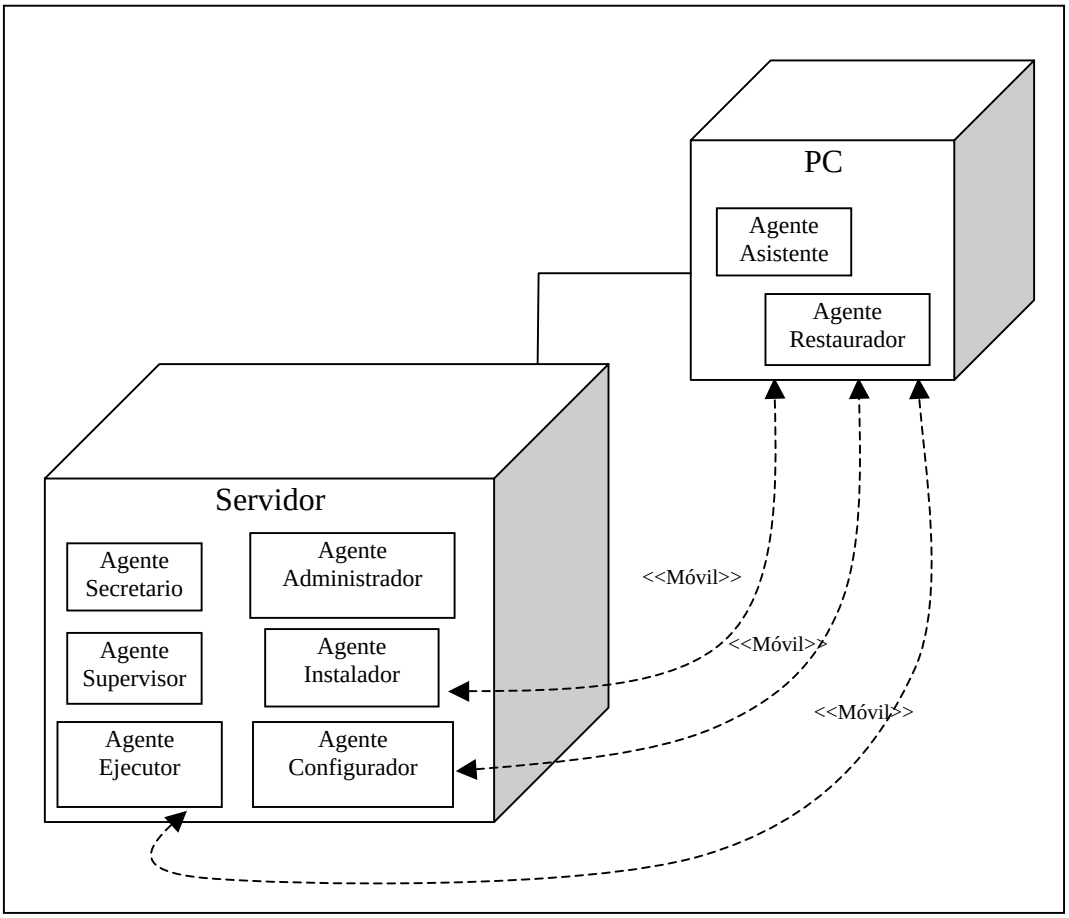

Fig. A13 Diagrama de distribución

Anexo B
Manual de instalación del prototipo

B.1 Instalación de JADE

Para la instalación de la plataforma de agentes JADE, se requiere, como se menciona en el capítulo 2, contar con la versión JDK 1.2 o una superior. El procedimiento de la instalación del programa JDK se ha omitido, ya que no presenta mayor dificultad y existe información al respecto en el sitio Web de Sun.

Con respecto al procedimiento que hay que seguir para trabajar con la plataforma JADE, se requiere el archivo JADE-bin-3.0b1.zip, que contiene las librerías necesarias para ejecutar la plataforma de agentes, y puede ser descargado del sitio Web http://sharon.cselt.it. Como se trata de un archivo zip, hay que descomprimirlo en algún directorio, para este ejemplo se realizó en el directorio raíz. Una vez descomprimido el archivo se crea una carpeta llamada jade que contiene los archivos necesarios para ejecutar la plataforma. Los archivos se encuentran dentro de jade/lib y son:

- Base64.jar
- jade.jar
- iiop.jar
- jadeTools.jar

Estos archivos tienen que direccionarse por medio de la variable de ambiente CLASSPATH, que se encuentra en el caso del sistema Microsoft Windows XP en *Inicio>Panel de control>Sistema* donde se muestra una ventana que dice *Propiedades del sistema*, luego hay que dirigirse a la pestaña que dice *opciones avanzadas* y oprimir el botón de *Variables de entorno*, se despliega una nueva

ventana que dice *Variables de entorno*, donde se tiene que indicar dentro de la variable CLASSPATH la ruta donde se encuentran los archivos jar como se muestra en la figura siguiente.

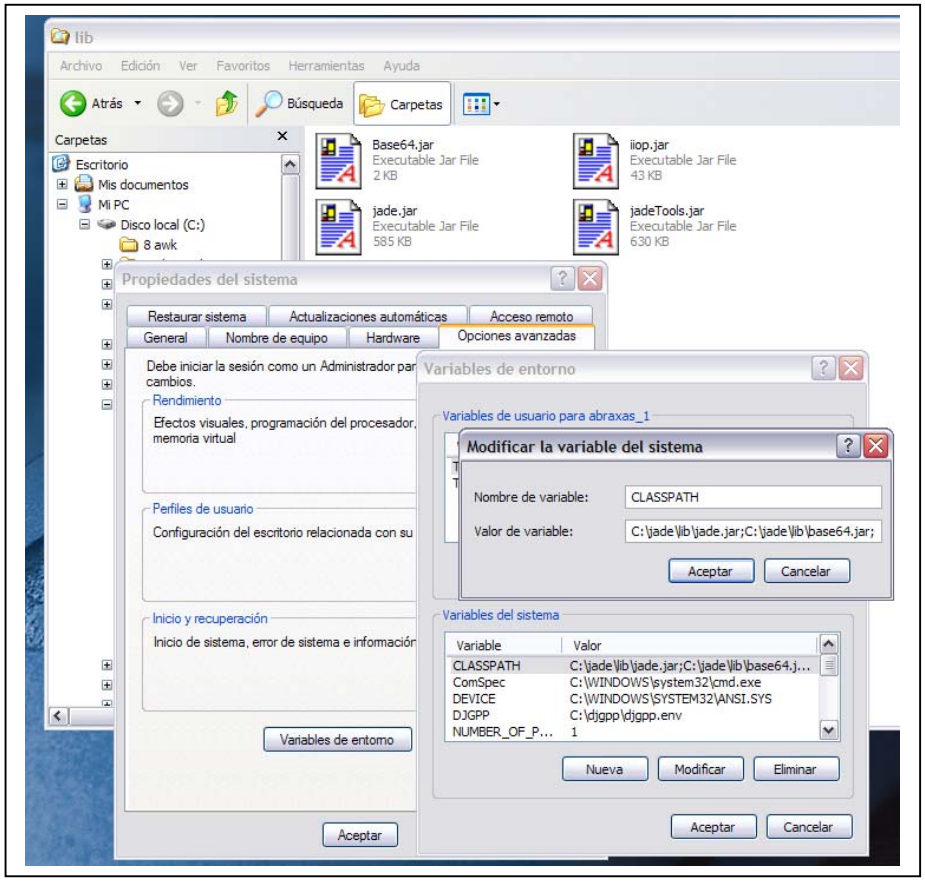

Fig. B1. Variable de entorno CLASSPATH.

Ya que se ha realizado este procedimiento, la plataforma de agentes JADE esta lista para ser ejecutada desde el símbolo del sistema (Consola) por medio de la instrucción:

java jade.Boot –gui

B.2 Instalación del prototipo

Ahora se mencionan lo pasos que se requieren para completar la instalación del sistema multi-agente. Lo primero que se tiene que hacer es declarar en la variable de ambiente CLASSPATH el archivo jbcl.jar que pertenece a la aplicación Borland JBuilder 8, de la misma forma en que se hizo con los archivos de la plataforma JADE.

Como segundo procedimiento, hay que indicar también en la variable de ambiente CLASSPATH, dónde se encuentra el paquete que contiene las clases de los agentes. Para esto sólo se indica la carpeta donde se encuentra el paquete principal, llamado **sars**, y no cada uno de los archivos que conforman el prototipo. En este ejemplo el paquete *sars* ha sido colocado en la carpeta *C:\jade*, y por tanto, la figura B2 muestra como se debe realizarse el procedimiento en el CLASSPATH.

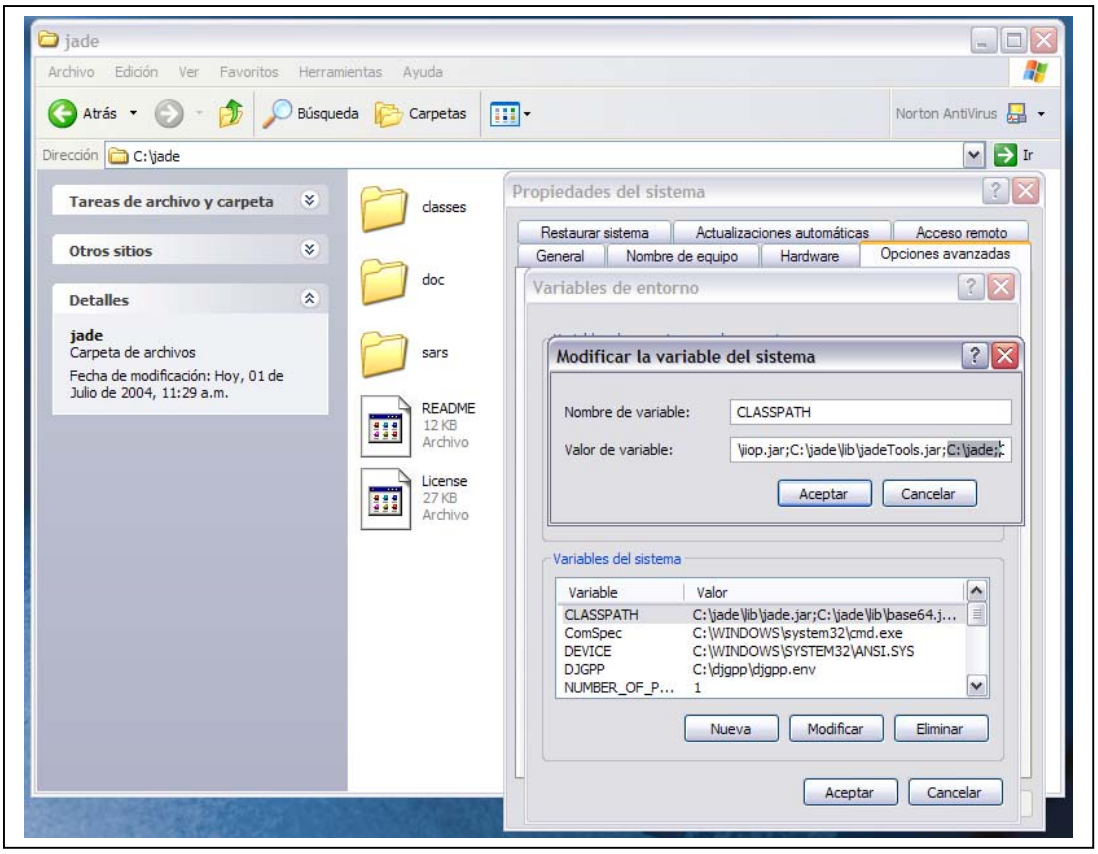

Fig. B2. Declaración del paquete del prototipo.

Los siguientes pasos, sólo se realizan en la máquina donde se encontrará el agente Administrador. Lo que prosigue es dar de alta la Base de Datos en el sistema operativo, para ilustrar la forma de hacerlo se oprime el botón de la barra de herramientas *Inicio>Panel de control*, luego seleccionar *Herramientas administrativas* y de la nueva ventana se selecciona el icono *Orígenes de datos (ODBC)*, aparecerá una ventana con titulo Administrador de orígenes de datos (ODBC), donde se debe seleccionar la pestaña *DNS de sistema*, se oprimirá el botón agregar desplegándose una ventana con título crear nuevo origen de datos como se muestra en la figura B3

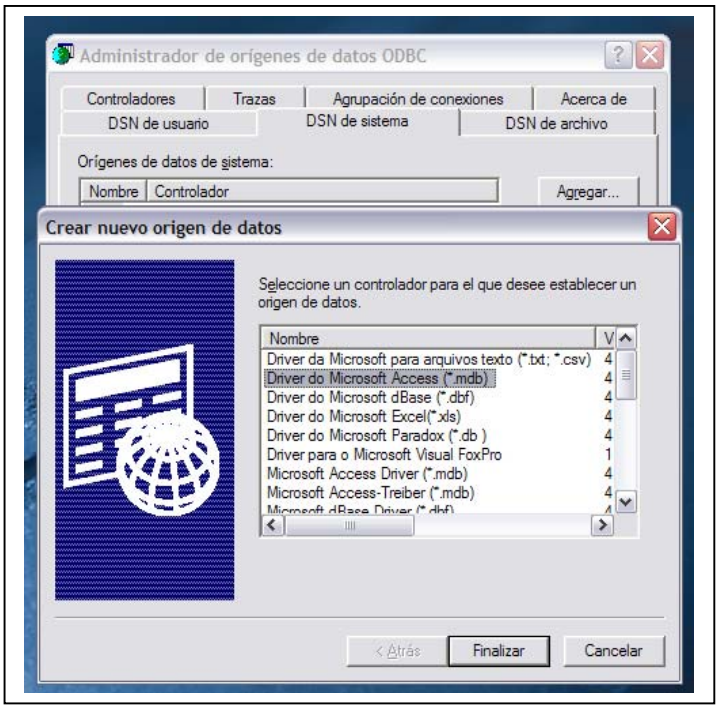

Fig. B3 Crear controlador para la base de datos.

Hay que seleccionar la opción *Driver de Microsoft Access (*.mdb)* y oprimir el botón *Finalizar*. Aparecerá la ventana de la figura B4, donde se introducen los datos de la base de datos. En el campo Nombre de origen de datos hay que colocar el nombre *DB2* que es el nombre de la Base de datos que el agente Administrador reconoce. Se requiere oprimir el botón *Seleccionar* para indicar dónde se encuentra físicamente la base de datos *DBPrueba.mdb*.

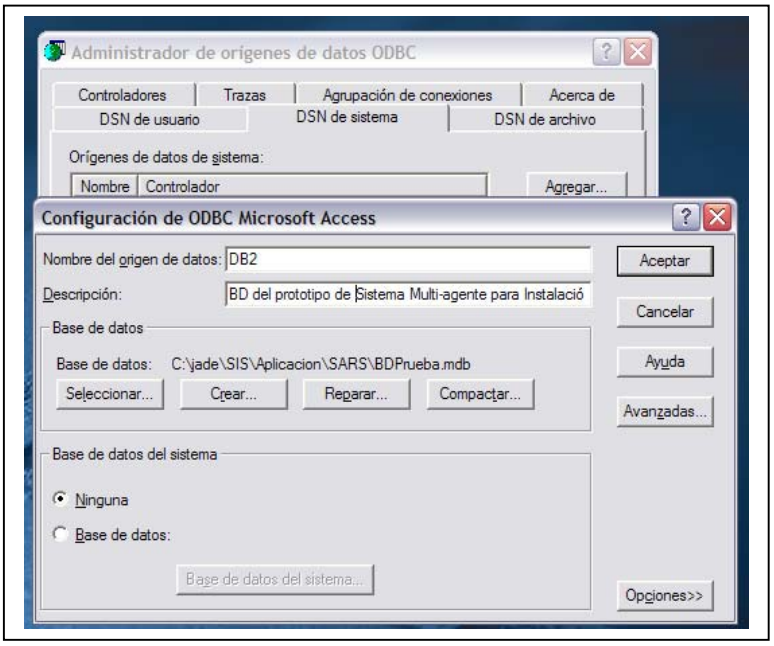

Fig. B4 Configurar ODBC para base de datos

Realizados los pasos anteriores solo hay que aceptar y la base de datos se habrá dado de alta en el sistema operativo.

Por último se tiene que instalar el servidor Web para que se realice el acceso a los archivos que tiene a su resguardo el agente Administrador. En este caso se instaló el servidor Apache 2.0 descargado de la página http://www.apache.org/. La instalación del servidor no será descrita ya que el procedimiento es ampliamente documentado en la pagina, además de poseer una interfaz amigable el instalador y solo se describirá la creación de un nuevo contenedor.

Para el caso del prototipo se crearon dos contenedores llamados *software* y *utilerias*, que se encuentran localizados en la ruta *C:/Documents and Settings/abraxas_1/Mis documentos/Mis descargas/* pero puede indicarse cualquier otra ruta, esto tiene que indicarse en el archivo de configuración de apache llamado *httpd.conf* localizado dentro de la carpeta dónde se instaló apache como es *Apache2\conf.*

En este archivo se tiene que agregar las siguientes líneas :

```
Alias /software/ "C:/Documents and Settings/abraxas_1/Mis
documentos/Mis descargas/Programas/"

<Directory "C:/Documents and Settings/abraxas_1/Mis documentos/Mis
descargas/Programas">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

```
Alias /utilerias/ "C:/Documents and Settings/abraxas_1/Mis
documentos/Mis descargas/Utilidades/"

<Directory "C:/Documents and Settings/abraxas_1/Mis documentos/Mis
descargas/Utilidades">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Con estos procedimientos se completan los pasos necesarios para poder ejecutar el prototipo del sistema de agentes para la instalación de software vía remota.

Índice de siglas.

ACC	Agent Communication Channel.
ACL	Agent Communication Language.
AI	Artificial Intelligent.
AID	Agent Identifier.
AMS	Agent Management Service.
AOP	Agent Oriented Program.
AP	Agent Plataform.
API	Application Programming Interface.
ATM	Asynchronous Transfer Mode.
ATP	Agents Transport Protocol.
AUML	Agent Unified Modeling Language.
BDI	Belief Desire Intention.
CORBA	Common Object Request Broker Arquitecture.
CRC	Class Responsibility and Collaboration.
DF	Diretory Facilitator.
DRAE	Diccionario de la Real Academia Española.
FIPA	Foundation of Intelligent Phisical Agents.
FTP	File Transfer Protocol.
GUI	Graphical User Interface.
HTTP	Hyper Text Transfer Protocol.
IDEF	Integration Definition for Function modeling.
IIOP	Internet Inter-ORB Protocol.
JADE	Java Agent Development Framework.
JDK	Java Development Kit.
JRE	Java Runtime Enviroment.
KIF	Knowledge Interchange Format.

KQML	Knowledge Query and Manipulation Language.
MAS	Multi-Agent Systems.
MSC	Message Sequence Chart.
MTS	Message Transport Service.
OMG	Object Management Group.
OMT	Object Modeling Technique.
OOSE	Object-Oriented Software Engineering.
ORB	Object Request Broker.
PDA	Personal Digital Assistant.
POP3	Post Office Protocol 3.
PSM	Problem Solving Methods.
RADL	Reticular Agent Definition Language.
RMA	Remote Management Agent.
SMTP	Simple Mail Transfer Protocol.
TCP/IP	Transmission Control Protocol/Internet Protocol.
URL	Universal Resourse Locator.

Consultas y Referencias

Consultas

1. Gómez S., Antonio F. Y Botía B. Juan A., 2002, “Tecnologías y Plataformas de Agentes”, Murcia, España.

2. Botía B., Juan A., 2001, “Introducción a los agentes software-MAS”, Murcia, España.

3. GSI-DIT, “Agentes Inteligentes. Introducción”, España.

4. GSI-DIT, “Agentes inteligentes. Parte 1. Introducción SBC”, España.

5. GSI-DIT, “Agentes Inteligentes. Parte 2. Lenguaje CLIPS”, España.

6. GSI-DIT, “Agentes Inteligentes. Parte 3. Comportamiento”, España.

7. GSI-DIT, “Comunicación entre agentes”, España.

8. GSI-DIT, “Arquitectura de agentes inteligentes”, España.

9. Iglesias F., Carlos A., 1999, “Introducción a la Inteligencia Artificial Distribuida”, Madrid, España.

10. Tveit, Amund, 2001, “A survey of Agent-Oriented Software Engineering”, Norwegian, USA.

11. Wooldridge, Michael, 2002, “An Introduction to MultiAgent Systems”, Liverpool, UK.

12. Müller, Jörg, et al, 1997, “Intelligent Agents III, Agent Theories, Architectures, and Languages”, Germany.

13. V. Julián, V. Botti, 2002, “Agentes Inteligentes: el siguiente paso en la Inteligencia Artificial”

14. Gómez S., Jorge, et al, 2001, “Análisis y Diseño de Sistemas Multi-Agente”, Madrid, España.

15. Schafroth, Mika, 2002, “JAVA AGENTS”, University of Kuopio.

16. Camacho, David, et al, 2002, “MAPWEB: Cooperation between Planning Agents and Web Agents”.

17. Nwana, Hyacinth S., 1996, “Software Agents: An Overview”,BT Laboratories Martlesham Heath Ipswich, U.K..

18. Nava M., S. E., 2002, “Federación de Bibliotecas Digitales utilizando Agentes Móviles”, Puebla, México.

19. Wooldridge, Michael, “Intelligent Agents”.

20. Arroyo C., Ángel, “Agentes Inteligentes”, Madrid, España.

21. DeLoach, Scott A., “Internal Agent Architectures”, Kansas State University.

22. Henao C., Mónica, 2000, “CommonKADS, una buena herramienta para la Gerencia del Conocimiento”, Revista Universitaria EAFIT, abril.

23. Cernuzzi, Luca y Rossi, Gustavo, 2002, “On The Evaluation Of Agent Oriented Modelling Methods”, Universidad Catolica, Asunción, Paraguay.

24. Vicente J., Julián y Vicente J., Botti, 2003, “Estudio de métodos de desarrollo de sistemas multiagente”, Universidad Politécnica de Valencia, Revista Iberoamericana de Inteligencia Artificial, No. 18.

25. Iglesias, Carlos A.,et al, “Metodologías orientadas a agentes”, Universidad Politécnica de Madrid, España.

26. Caire, Giovanni, et al, 2000, “Agent Oriented Analysis using MESSAGE/UML” EURESCOM.

27. Evans, Richard, et al, 2001, “Methodology for Agent-Oriented Software Engineering (final) 20/09/2001”, EURESCOM.

28. Gómez Sanz, Jorge J., 2003, “Metodologías para el desarrollo de sistemas multi-agente”, Universidad Complutense, Revista Iberoamericana de Inteligencia Artificial, No. 18.

29. Glaser, Norbert, 1996, “The CoMoMAS Approach: From Conceptual Models to Executable Code”, L’Universitité Henri Poincaré, France

30. Amor, M., 2002, “Interroperatibidad entre Plataformas de Agentes FIPA: Una Aproximación Basada en Componentes”,Universidad de Málaga, España.

31. Campo V. Maria C., et al, 2002, “TagentsP y PDP: propuesta para una plataforma de agentes en computación ubicua”, Madrid, España.

32. “Un Marco de Compunicación Inter.-Agentes en una Biblioteca Digital”, Jardín Botánico de Missouri y Universidad de las Américas-Puebla.

33. Sánches R., David, et al, “How to upgrade source code written for jade 2.3 to 2.6” Torragona, España.

34. Díaz A., Belén, Fuentes F., Rubén, 2003, “Introducción a Jade”, Depto. De Sistemas Informáticos y Programación, UCM.

35. Isern A., David, 2001, “Desenvolupament de Sistemes MultiAgent JADE (Java Agent Development Framework)”, Universitat Rovira I Virgili.

36. Pérez D., Jesús A., 2000, “SAHARA: Arquitectura de seguridad integral para sistemas de agentes móviles”, Universidad de Oviedo, España.

37. Giang, Nguyen T., y Tung, Dang T., 2002, “Agent Platform Evaluation and Comparison”, Institute of Informatics, Slovak Academy of Sciences.

38. Mangina, Eleni, 2002, “Review of Software Products for Multi-Agent Systems”, AgentLink, UK.

39. Sitio Web de FIPA-OS.
http://fipa-os.sourceforge.net/

40. Sitio Web de Herramientas y plataformas para la construcción de agentes.
http://www.agentbuilder.com/AgentTools/index.html

41. Sitio Web de JADE.
http://sharon.cselt.it

Referencias

[1]	Nwana, Hyacinth S., 1996, “Software Agents: An Overview”,BT Laboratories Martlesham Heath Ipswich, U.K..
[2]	Iglesias F., Carlos A., “Fundamentos de los agentes inteligentes”, Universidad Politécnica de Madrid, España
[3]	Diccionario de la Real Academia Española http://www.rae.es
[4]	“Agentes Inteligentes. Introducción”, GSI-DIT, España.
[5]	1999, "Uso de la Orientación a Objetos para el diseño e implementación de Agentes", Universidad Católica Nuestra Señora de la Asunción, Paraguay.
[6]	Maes, P., 1995, “Artificial Life Meets Entertainment: Life like Autonomous Agents”, Communications of the ACM, Vol. 38, Nº. 11.
[7]	1995, Wooldridge, Michael y Jennings, Nicholas R., “Intelligent Agents: Theory and Practice”
[8]	Russell, S. J. y Norvig P., 1995 ,“Artificial Intelligence: A Modern Approach”. NJ, USA.
[9]	FIPA, 2001, “FIPA Agent Management Specification”, pp 2.
[10]	Cadena S., Carlos A. y Núñez E., Gustavo,1999, “Arquitectura de los Agentes Adaptativos: Un estado del Arte”,No. 15, Serie verde, D.F.,México.
[11]	Maes P., 1991, “Agents that reduce work and information overload”, Communication of the ACM, Vol. 37, Nº. 7, pp. 31-40.
[12]	http://ai.eecs.umich.edu/cogarch2/specific/homer.html
[13]	Amandi, Analía, 2001, “Desarrollo de Sistemas Multi-Agentes”, Revista Iberoamericana de Inteligencia Artificial, No. 13, Buenos Aires, Argentina.
[14]	Feerguson, Innes A., 1992, “TouringMachines: An Architecture for Dynamic, Racional, Mobile Agents”, University of Cambridge.
[15]	Iglesias F., Carlos A., 1997, “Fundamentos de los agentes inteligentes”, Universidad Politécnica de Madrid, España.
[16]	Amandi, Analía, 2001, “Desarrollo de Sistemas Multi-Agentes”, Revista Iberoamericana de Inteligencia Artificial, No. 13, Buenos Aires, Argentina.
[17]	Gómez B., Gabriel J. Y Zea R., Claudia M., “Incorporación de Agentes Inteligentes en Ambientes de Aprendizaje”,Medellín, Colombia.
[18]	Acero M., Andrés D., 2002, “Software Agents: an Overview Hyacinth Nwana, and Michael Wooldridge, BT Techonology Journal 14(4) 1996”, Universidad Nacional de Colombia, Medellin, Colombia.
[19]	Rumbaugh, J., Blaha, et al, 1991, “Object-Oriented Modeling and Design”. Prentice-Hall,.
[20]	Iglesias F., Carlos A., et al, “A Survey of Agent-Oriented Methodologies”, Valladolid, Spain.
[21]	Shoham, Yoav, 1993, “Agent-oriented programming”. Artificiall Intelligence, Vol. 60, Nº. 1, Marzo.
[22]	Salvador I. Jorge, 2003, “Elaboración de una Aproximación Metodológica para el desarrollo de Software Orientado a Sistemas Multiagente”, Universidad Politécnica de Madrid, España.

[23]	Wooldridge, Michael, 2000, “The Gaia Methodology for Agent-Oriented Analysis and Design”, Kluwer Academic Publischers.
[24]	Glaser, N., 1996, “Contribution to Knowledge Modelling in a Multi-Agent Framework (the CoMoMAS Approach)”, L’Universitité Henri Poincaré, France
[25]	Iglesias F., Carlos A., 1998, “Definición de una metodología para el desarrollo de sistemas multiagente”, Madrid, España.
[26]	2000, “AgentBuilder an Integrated Toolkit for Constructing Intelligent Software Agents”, Reticular Systems.
[27]	Mosquera F., Celestino, 2001, “Análisis y estudio experimental de herramientas basadas en agentes”, Coruña, España.
[28]	Fonseca, Steven P., et al, 2001, “Evaluation of the Zeus MAS Framework”, HP.
[29]	Poslad, Stefan, et al, “The FIPA-OS agent platform: Open Source for Open Standards”, London, UK.
[30]	Avancini, Henri H., 2000, “FraMaS: Un Framework para Sistemas Multi-agente basado en Composición”, Buenos Aires, Argentina.
[31]	Bellifemine, Fabio, et al, 2002, “Jade Administrator’s Guide”, TILAB, University of Parma.
[32]	Cruz R., Alejandro, 1998, “UML Lenguaje Unificado de Modelado”, D.F., México
[33]	Bauer, Bernhard, et al, 2000, “Agent UML: A Formalism for Specifying Multiagent Interaction”, München, Germany.
[34]	Odell, James, 2000, “Extending UML for Agents”.