UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 4

TÍTULO: Evaluación de las Metodologías MaSE y MAS-CommonKADS en un caso de estudio propuesto

TRABAJO DE DIPLOMA EN OPCIÓN AL TÍTULO DE
INGENIERO EN CIENCIAS INFORMÀTICAS

Autoras:
Laura Fortún Milián
Sahily Ramos García

Tutor:
Ing. Rolando Pérez Pinto

Ciudad de la Habana
Junio, 2007

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Evaluación de las Metodologías MaSE y MAS-CommonKADS en un caso de estudio propuesto.

Autores: Laura Fortún Milián y Sahily Ramos García.

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero en Ciencias Informática; y propongo que se le otorgue al Trabajo de Diploma la calificación de: _____

Ing. Rolando Pérez Pinto.

Firma _____ Fecha _____

DECLARACIÓN DE AUTORÍA

Declaramos que somos las únicas autoras de este trabajo y autorizo a la Facultad 4 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Autores:

Laura Fortún Milián

_____

Sahily Ramos García

_____

Tutor:

Ing. Rolando Pérez Pinto

_____

Dedicatoria

A mi mamá, a mi papá y en especial a tía Olga.

A mi familia.

Laura

A mi familia y en especial a Sandra.

Sahily

Agradecimientos

Muchos han contribuido de una forma u otra con la realización de este proyecto, a todos ellos nuestro más sincero agradecimiento, en especial a Rolando Pérez. A la UCI por darnos la oportunidad de formar parte de esta tropa del futuro y formarnos como los profesionales que somos.

De Sahily

Agradezco a todos aquellos que de una forma u otra me han brindado su cariño y apoyo, muy en especial a:

-Mi mamá (chuli linda), mi papá (Pito), mi mimix, Diego, Sandra, Titi y Gilberto, por ser lo más grande que tengo en el mundo.

-Mis amistades de Pinar en especial a Odelkys por su constante cariño

-Gisellys: Por estar siempre a mi lado

-Yanay: Por su incalculable amistad y apoyo en las buenas y las malas.

-Pupo: Por ser tan maravillosa para mi.

-Laura: Por ser la uri bella.

-Doctor GC: a quien admiro muchísimo y llevo siempre en mi corazón.

De Laura

-A mis padres y a tía Olga: por el inmenso amor dado, por estar siempre presentes. Los quiero mucho.

-A mi familia y en especial a mi tío Ale, a Iro y a Migue: por sus traducciones, apoyo y preocupación.

-A Taimy, Sayli, Frank, Ruly, Titonga, Day, Sahily, Yanay, Lien, Yaima, Sachel y Osley: por su ayuda, su amistad sincera, por los buenos momentos.

-A Raissel: por sus consejos, apoyo, incondicionalidad y todo su cariño durante estos años.

A todos: Mil gracias.

Resumen

En los últimos diez años la Inteligencia Artificial ha tomado auge y con ello la creación de Sistemas Multi-Agentes, los que se conciben como una poderosa rama de esta ciencia, que a su vez juega un papel fundamental en el quehacer de los principales desarrolladores.

La Universidad de las Ciencias Informáticas no ha quedado exenta de esta tecnología, y una vez más se suma al grupo de investigadores que han incursionado en el tema. Actualmente el Centro de Estudios de Ingeniería de Sistemas (CEIS) del Instituto Superior Politécnico “José Antonio Echeverría” (ISPJAE) trabaja en un proyecto relacionado con la creación de Sistemas Multi-Agentes. Por consiguiente, los nuevos sistemas requerirán de metodologías que resulten en una descripción detallada de su ciclo de vida.

El actual trabajo de tesis propone el análisis de dos metodologías ampliamente difundidas en la especificación de Sistemas Multi-Agentes: MaSE (del inglés MaSE: Multiagent System Engineering) y MAS-CommonKADS. En el mismo se expone una breve panorámica sobre el estado del arte de la Ingeniería de Software Orientada a Agentes, abordando de forma precisa los aspectos teórico-prácticos de las Metodologías MaSE y Mas-CommonKADS. Seguido a ello se aplican ambas metodologías a un caso de estudio propuesto, que permite a su vez desarrollarlas y arrojar resultados que posteriormente serán comparados, para de esta forma determinar entre ellas la de mayor eficiencia en el diseño de un Sistema Multi-Agente.

PALABRAS CLAVE

Metodología

Inteligencia Artificial

Inteligencia Artificial Distribuida

MaSE

MAS-CommonKADS

Agente

Sistema Multi Agente

Índice

Introducción

En la actualidad cubana se ha escrito e investigado poco sobre los temas de agentes, Sistemas Multi-Agentes (SMA) y metodologías empleadas para éstos. Algunas universidades y centros de investigación han hecho aportes al tema, pero ciertamente es aún incipiente en desarrollo y madurez.

LA Ingeniería del Software Orientada a Agentes (ISOA) es una disciplina que carece de conocimientos, pues dada la complejidad de los SMA, y los requisitos de recursos de los mismos, hacen que su implementación sea costosa en cuanto a personal y equipamiento. Existe una falta de cultura de programación sobre este tema, lo que conlleva a la no existencia de estándares debidamente aceptados. No solo en Cuba, sino a nivel mundial, no se cuenta con metodologías lo suficientemente robustas para la modelación de Sistemas Multi-Agentes y se presentan diversos problemas entre las ya existentes, en las que algunas son inconclusas e incapaces de abarcar el proceso en su totalidad.

Es necesario aludir al hecho de que los Sistemas Multi-Agentes poseen un conjunto de características pertenecientes a tópicos complejos que son tratados en toda el área de computación actual, por lo que es precisamente ello lo que los convierten en sistemas factibles en la solución de grandes problemas.

Por otra parte la investigación en metodologías orientadas a agentes es un campo que no ha alcanzado suficiente desarrollo, ya que es precisamente un tema que comienza a dar sus primeros pasos, por lo que en estos momentos continúa bajo investigaciones.

El acelerado avance del paradigma orientado a agentes, da lugar a que la Universidad de las Ciencias Informáticas (UCI) comience a incursionar y se sume con sus aportes e investigaciones al grupo de centros que han presentado trabajos al respecto.

Si se lleva a cabo un estudio minucioso de MAS-CommonKADS [5] y MaSE [6] aplicadas a un caso de estudio, ello permitirá a nuestra universidad contar con un material preciso de apoyo, basado en metodologías que respalden el diseño de futuros proyectos sustentados por agentes.

La necesidad de existencia de investigaciones concretas sobre las metodologías que se emplean para el trabajo con agentes, les facilitará de una vez a los desarrolladores centrarse en la solución y creación de estos sistemas, disminuyendo por tanto el esfuerzo que se pueda emplear en la búsqueda y documentación sobre este tipo de metodologías.

La presente investigación contempla como objeto de estudio la Inteligencia Artificial y la Ingeniería del Software, teniendo como campo de acción la Ingeniería del Software Orientada a Agentes y las Metodologías MaSE y MAS-CommonKADS.

En este sentido el presente trabajo propone como objetivo general, dado un caso de estudio, el establecimiento de comparativas entre los resultados arrojados por dos metodologías importantes que se emplean en el diseño de Sistemas Multi-Agentes: MaSE [6] y MAS-CommonKADS [4] arribando a conclusiones respecto a las mismas y determinando entre ellas la de mayor efectividad. La investigación, además contempla entre sus objetivos específicos la realización del estado del arte de la ISOA, así como una evaluación crítica de cada una de las metodologías dado el caso de estudio propuesto.

Finalmente, otro objetivo constituye el acaparamiento de la atención de los informáticos cubanos, así como despertar motivación en ellos para futuras investigaciones sobre Sistemas Multi-Agentes.

Es importante aludir al hecho de que este trabajo se encuentra enmarcado en una investigación de mayor rango que implica el uso de varias metodologías, las que ya se han desarrollado en el caso de estudio propuesto, quedando como pendientes de aplicación MaSE y MAS-CommonKADS. El estudio de dichas metodologías forma parte de la Tesis de Maestría del tutor de esta investigación.

A fin de lograr una mejor comprensión el presente documento se estructuró en tres capítulos, cuyos títulos son: “Ingeniería del Software Orientada a Agentes: MaSE y MAS-CommonKADS”, “Aplicación de las metodologías MaSE y MAS-CommonKADS a un caso de estudio”, y “Evaluación de MaSE y MAS-CommonKADS”.

El primer Capítulo tiene en cuenta importantes definiciones para el entendimiento de las metodologías, así como las características específicas de cada una. En el mismo se alude además, al paradigma de agentes, Sistemas Multi-Agentes, y la ISOA.

El segundo Capítulo presupone la aplicación de las metodologías: MaSE y MAS-CommonKADS al caso de estudio propuesto, donde se llevan a cabo las etapas de cada una de ellas que fueron identificadas en el Capítulo 1.

El tercer Capítulo se centra en el análisis evaluativo de MaSE y MAS-CommonKADS. En el mismo se propone además un framework para comparar ambas metodologías, donde se resaltan las principales fortalezas y debilidades de las mismas, obteniéndose resultados que ayudan a identificar la más eficiente.

Capítulo 1 Ingeniería del Software Orientada a Agentes: MaSE y MAS-CommonKADS

Introducción

El presente Capítulo propone brindar un acercamiento a las metodologías: MaSE y MAS-CommonKADS, así como dar una panorámica sobre la inteligencia artificial, los conceptos de agentes y Sistemas Multi-Agentes. Se presenta una compilación de los principales documentos propuestos por los desarrolladores de ambas metodologías, abordando de forma sintética y completa las diversas etapas por las que transitan para obtener el ciclo de vida de un proyecto.

1.1 Evolución de la inteligencia artificial

La tecnología avanza a gran velocidad y los constantes cambios son apreciados con mayor frecuencia cada día. La programación de software no va enajenada al desarrollo del hardware y mantiene su paso firme, enriqueciéndose y expandiendo exitosamente en sus diferentes lenguajes todo un cúmulo de aportes que la realzan y contribuyen en gran medida al desarrollo de la informática.

Paralelamente al auge alcanzado por la Programación Orientada a Objetos en los ochenta, comienza a afianzarse la Inteligencia Artificial (IA) como una disciplina de amplio espectro de desarrollo. Es precisamente el objetivo de "desarrollar sistemas que piensen y actúen racionalmente" [2]. Todo esto ha devenido en el creciente desarrollo y evolución de la Inteligencia Artificial, la cual se ha expandido incluso con más auge que las restantes disciplinas, motivado probablemente por su propia inmadurez. Las investigaciones realizadas en el campo de la Inteligencia Artificial han hecho notables aportes en la creación de software que simule características humanas tales como: inteligencia, creatividad, comunicación en lenguaje natural, aprendizaje, entre otras. Todo ello ha posibilitado que las computadoras se conviertan en asistentes para las personas y que brinden servicios sumamente elevados en nivel y calidad. El desarrollo de redes de computadores (redes corporativas y redes locales) ha sido un

gran paso hacia la evolución de organizaciones de computadores, ya que la colaboración entre individuos requiere que se establezcan ligas de comunicación, y que éstas se usen en forma efectiva.

Sin lugar a dudas, la IA puede ser aplicada hoy en día a infinidad de disciplinas científicas y es que la IA es susceptible de aparecer allí donde se requiera el intelecto humano. Actualmente en la IA ha surgido un nuevo paradigma conocido como: paradigma de agentes, el cual está tomando un gran auge entre los investigadores. Dicho paradigma aborda el desarrollo de entidades que puedan actuar de forma autónoma y razonada [2].

Por otra parte surge la Inteligencia Artificial Distribuida (IAD), la cual es definida actualmente como un sub-campo de la IA y la misma tiene su basamento en los resultados que se obtienen producto a la interrelación y cooperación de diversas entidades denominadas agentes, además investiga modelos de conocimiento, así como técnicas de comunicación y el razonamiento que necesitan los agentes computacionales para participar en la solución de problemas que son comunes a ellos mismos. Desde sus inicios ésta se centraba en temas de concurrencia de la IA, tales como la creación de lenguajes de programación que sirvieran para mejorar los sistemas inteligentes y lograr mayor eficiencia de los mismos. A pesar de ser una disciplina joven, podemos distinguir tres periodos cronológicos bien diferenciados:

La IAD “clásica”, la cual se centra el estudio de la conducta colectiva, opuestamente a la IA, que estudia la conducta individual

La IAD “autónoma”, la cual se centra en el estudio de los agentes individuales situados en un mundo social.

La IAD “comercial”, la cual se centra en la aplicación de la IAD clásica y autónoma, desarrollando agentes (denominados genéricamente agentes software) con características muy diferenciadas (agentes móviles, personales) que están siendo explotados de forma comercial.

Actualmente la IAD ha afianzado su estudio sobre el concepto de Agentes y Entidades Colaboradoras. La concepción de un agente como sistema autónomo, capaz de interactuar recíprocamente con otros agentes en aras de satisfacer una solicitud bien sea propia o externa, introduce a la IAD, lo que se convierte en el centro de las investigaciones actuales en la rama de la IA: los Sistemas Multi- Agentes [4].

La investigación en la Solución de Problemas Distribuidos (SPD) considera la forma en que se puede dividir la tarea de solucionar un problema entre un número de módulos (o nodos) que cooperan en dividir y compartir conocimiento acerca del problema y su correspondiente solución. En un sistema SPD puro, todas las estrategias de interacción (cooperación y coordinación) se incorporan como una parte integral del sistema.

1.2 Agentes

Hasta hace algún tiempo atrás los campos de la IA funcionaban de forma independiente del resto, por lo que no existía un mecanismo común que los agrupase, es por ello que surgen los agentes, los cuáles tienen como objetivo fundamental la integración de todos los campos de la IA, aplicándolos a problemas reales. El agente se puede decir que constituye el elemento básico de construcción y representa sólo una porción del problema el cual se analiza precisamente dividiéndolo en agentes que trabajan conjuntamente.

Las definiciones de agentes resultan diversas y aún no se tiene alguna formal que precise exactamente qué es un agente, éstos por lo general son vistos como entidades inteligentes, equivalentes en términos computacionales a un proceso del sistema operativo, que existen dentro de cierto contexto o ambiente, y que se pueden comunicar a través de un mecanismo de comunicación inter-proceso, usualmente un sistema de red, utilizando protocolos de comunicación.

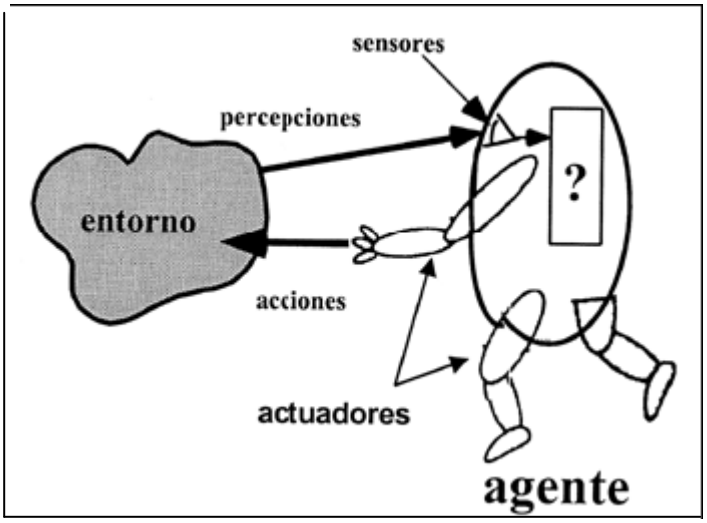

Figura 1: Visión esquemática de un Agente Inteligente

1.2.1 ¿Cómo es caracterizado un agente?

Un agente va a venir caracterizado por una serie de calificativos, los cuales vienen a denotar ciertas propiedades a cumplir por el agente [2]

Proactivo: A su vez el agente debe ser capaz de intentar cumplir sus propios planes u objetivos.

Reactivo: El agente es capaz de responder a cambios en el entorno en que se encuentra situado.

Social: Debe de poder comunicarse con otros agentes mediante algún tipo de lenguaje de comunicación de agentes.

Continuidad Temporal: Se considera un agente un proceso sin fin, ejecutándose continuamente y desarrollando su función.

Autonomía: Un agente es completamente autónomo si es capaz de actuar basándose en su experiencia. El agente es capaz de adaptarse aunque el entorno cambie severamente. Por otra parte, una definición menos estricta de autonomía sería cuando el agente percibe el entorno.

Sociabilidad: Este atributo permite a un agente comunicar con otros agentes o incluso con otras entidades.

Racionalidad: El agente siempre realiza «lo correcto» a partir de los datos que percibe del entorno.

Reactividad: Un agente actúa como resultado de cambios en su entorno. En este caso, un agente percibe el entorno y esos cambios dirigen el comportamiento del agente.

Pro-actividad: Un agente es pro-activo cuando es capaz de controlar sus propios objetivos a pesar de cambios en el entorno

Adaptatividad: Está relacionado con el aprendizaje que un agente es capaz de realizar y si puede cambiar su comportamiento basándose en ese aprendizaje.

Movilidad: Capacidad de un agente de trasladarse a través de una red telemática.

Veracidad: Asunción de que un agente no comunica información falsa a propósito.

Benevolencia: Asunción de que un agente está dispuesto a ayudar a otros agentes si esto no entra en conflicto con sus propios objetivos.

Los agentes proporcionan soluciones a aplicaciones:

Descentralizadas: Pueden actuar proactivamente e independientemente (sin ser invocados).

Mal estructuradas: En algunas aplicaciones es imposible especificar completamente ni sus componentes ni su interacción (interfaces), y los agentes ofrecen una solución real. Los agentes comparten los beneficios de la modularidad que permite la tecnología orientada a objetos, pero además su percepción del entorno es interna y sus variables de estado pueden ser un subconjunto de las del entorno.

Cambiantes: la modularidad permite modificaciones individuales y la descentralización minimiza el impacto de los cambios (sólo aquellos con los que interacciona).

Complejas: como las necesarias para el diseño, la planificación y el control de un elevado número de sistemas y datos en diferentes máquinas (distribución, concurrencia). Los Sistemas Multi-Agentes

permiten la gestión de la interacción en tiempo de ejecución (Ejemplo: 100 agentes con 10 comportamientos diferentes: el sistema resultante mostrará 10100 comportamientos para codificar).

En múltiples áreas son numerosas las aplicaciones que presenta este paradigma, fundamentalmente en dos áreas, que en este caso sería la industria y el comercio.

En el marco de las aplicaciones industriales, la tecnología basada en agentes es considerada muy apropiada para el desarrollo de sistemas industriales distribuidos. Dentro de esta línea podríamos destacar aquellas aplicaciones que se encargan de [2]:

Control de procesos: Gestión autónoma de edificios inteligentes en cuanto a su seguridad y consumo de recursos, gestión del transporte de electricidad, control de un acelerador de partículas, monitorización y diagnóstico de fallos en plantas industriales, como por ejemplo nucleares o refinerías, control en el proceso de bobinado del acero y robótica. En otro tipo de área se han desarrollado aplicaciones para el control del tráfico aéreo en aeropuertos como el de Sydney en Australia.

Producción: Aspectos como la planificación y scheduling de la producción o fabricación de productos serían tratados desde la perspectiva de agencia. Se ha aplicado con éxito, por ejemplo, a sistemas encargados de las fases de ensamblaje, pintado y almacenamiento de productos. Algunos ejemplos serían AARIA, ABACUS, CORTES, MASCOT, Sensible Agents, YAMS, entre otros.

Gestión de información: Como por ejemplo el filtrado inteligente de correo electrónico, de grupos de noticias o la recopilación automática de información disponible en la red. Tareas para las cuales el agente necesita ser capaz de almacenar, aprender y manipular las preferencias y gustos e cada usuario, así como sus cambios. La imposibilidad en ocasiones de gestionar todo tipo de información suministrada por la red ha provocado que el agente se especialice en la búsqueda de determinados tipos de documentos. Otra posible línea sería la planificación de la agenda personal, en otras palabras, disponer de una secretaría virtual o asistente personal.

Comercio electrónico: En este caso la tecnología se emplea para proporcionar el entorno virtual donde realizar posibles operaciones comerciales (compra-venta de productos) o también para realizar tareas de búsqueda de productos (comparando precios, consultando disponibilidad) todo ello de manera

automatizada. En este caso, el agente debe poder comunicarse con las tiendas en línea utilizando protocolos que permitan trabajar con las interfaces de estas tiendas; actualmente, los usuarios pueden comprar y vender artículos comunes como libros y CD's de música. El empleo de agentes aumentará el impacto del comercio electrónico en un futuro muy cercano, revelando asimismo cómo los agentes basados en la Web pueden proporcionar un enorme poder añadido a los consumidores.

Monitorización: Proporcionan al usuario la información cuando sucede un determinado acontecimiento; por ejemplo cuando la información ha sido actualizada, trasladada de lugar o borrada. Este tipo de agentes permite tener alerta a un usuario frente a eventos en la red interesantes para el mismo. La forma en que este tipo de agentes sirve la información a su usuario puede ser el indicar únicamente qué página o páginas han cambiado y desde cuando ha sucedido esto o llegar a bajarse el texto de las páginas actualizadas, filtrando en este caso imágenes, gráficos y demás.

Mediador de diferentes fuentes de información: Se están realizando esfuerzos para desarrollar agentes que permitan interoperar a diferentes fuentes de información independientemente del sistema en que se hayan desarrollado.

Bajo el punto de vista de esta tecnología, los sistemas distribuidos pasan a ser Sistemas Multi-Agentes (SMA).

1.3 Sistemas Multi-Agentes

En los momentos actuales los Sistemas Multi-Agentes están acaparando la atención por parte de los desarrolladores de software, ya que resulta un nuevo paradigma que aunque es incipiente, promete logros para el creciente desarrollo informático.

La investigación en Sistemas Multi-Agentes (SMA) se orienta al estudio del comportamiento de un conjunto de agentes autónomos que tratan de dar solución a un problema dado. Un Sistema Multi-Agente puede definirse como una red de solucionadores de problemas que trabajan juntos para dar solución a problemas que están más allá de sus capacidades individuales. Estos solucionadores de problemas, que se les llama "agentes", son autónomos y pueden ser homogéneos o heterogéneos en su naturaleza

dependiendo de sus capacidades en la solución de problemas. A la investigación en Sistemas Multi-Agentes también se le conoce como Inteligencia Artificial Descentralizada (IADz). Por estos tiempos ha tomado auge la hipótesis de que si los programas que componen los sistemas distribuidos fueran inteligentes, si pudiesen reorganizar sus funciones para corregir o tolerar los fallos y si su coordinación pudiese estructurarse en términos intuitivos, el diseño y el mantenimiento de dichos sistemas sería más fácil de elaborar, más adaptable y fiable.

Para llevar a cabo el diseño de un SMA, se desarrollan un grupo de agentes como entes con motivación. El desarrollador tiene que centrarse en los objetivos que el sistema requiere y las tareas que se necesitan llevar a cabo para conseguir tales objetivos

En cierto modo, un Sistema Multi-Agente es un sistema distribuido en el cual los nodos o elementos son sistemas de inteligencia artificial, o bien un sistema distribuido donde la conducta combinada de dichos elementos produce un resultado en conjunto inteligente.

Es válido resaltar que los agentes no son necesariamente inteligentes. Existen como en todo el resto del dominio de la inteligencia artificial, dos enfoques para construir Sistemas Multi-Agentes:

El enfoque formal o clásico, que consiste en dotar de los agentes de la mayor inteligencia posible utilizando descripciones formales del problema a resolver y de hacer reposar el funcionamiento del sistema en tales capacidades cognitivas. Usualmente la inteligencia es definida utilizando un sistema formal (por ejemplo, sistemas de inferencia lógica) para la descripción, raciocinio, inferencia de nuevo conocimiento y planificación de acciones a realizar en el medio ambiente.

El enfoque constructivista, que persigue la idea de brindarle inteligencia al conjunto de todos los agentes, para que a través de mecanismos ingeniosamente elaborados de interacción, el sistema mismo genere comportamiento inteligente que no necesariamente estaba planeado desde un principio o definido dentro de los agentes mismos (que pueden ser realmente simples). Este tipo de conducta es habitualmente llamado comportamiento emergente.

La construcción de SMA integra tecnologías de distintas áreas de conocimiento: técnicas de ingeniería del software para estructurar el proceso de desarrollo; técnicas de inteligencia artificial para dotar a los

programas de capacidad para tratar situaciones imprevistas y tomar decisiones, y programación concurrente y distribuida para tratar la coordinación de tareas ejecutadas en distintas máquinas bajo diferentes políticas de planificación.

Un SMA tiene grandes ventajas con respecto a un sistema centralizado y monolítico:

Solución de problemas con mayor rapidez, debido al aprovechamiento de procesamiento paralelo.

Comunicación mínima, pues se transmite solamente soluciones parciales de alto nivel a otros agentes en lugar de tener que enviar datos básicos a un sistema central.

Mayor flexibilidad, pues se tienen agentes con diferentes habilidades que en forma dinámica cooperan entre sí para resolver problemas.

Mayor confiabilidad, pues otros agentes pueden tomar las responsabilidades de los agentes que llegasen a fallar en su operación.

Las características inherentes a los agentes, incrementada en los SMA, le aportan una elevada complejidad a su proceso de construcción, sobre todo cuando no se cuentan con los métodos y herramientas lo suficientemente completas y fáciles de utilizar en este sentido. A pesar de que actualmente se pueden encontrar en la literatura muchos trabajos relacionados con el proceso de desarrollo de SMA, es todavía un problema a resolver ya que, cada vez más se está requiriendo de métodos, técnicas y herramientas que faciliten aún más este proceso. La ingeniería de software asociada a los SMA, se ha llevado a cabo en los últimos años a partir del desarrollo de abstracciones del paradigma orientado a objetos, considerando al agente como una especialización de los objetos [3].

Son ya varias las áreas de aplicación de este tipo sistemas, tales como: el control de procesos, procesos de producción, control de tráfico aéreo, aplicaciones comerciales, gestión de información, comercio electrónico, aplicaciones médicas, juegos, etc. Entre las aplicaciones principales, uno de los campos esenciales lo constituyen los agentes software, los cuales se focalizan principalmente en la explotación cooperativa de recursos accesibles por Internet.

El otro campo de aplicaciones incluye la utilización de agentes físicos y ha dado lugar a la robótica cooperativa. Los SMA, conjugan los modelos de la IA con los desarrollos de los sistemas concurrentes y distribuidos, tienen un gran potencial de aplicación para la solución de problemas de la vida práctica y por ende para el desarrollo de nuevos productos. Algunos de los campos de aplicación de los SMA incluyen: asistentes personales, supervisión hospitalaria, asistentes financieros, manipulación y filtrado de información aplicados a diferentes dominios, agentes bancarios, ventas y comercio electrónico, difusión de noticias y publicidad, realidad virtual y avatares, control de procesos y manufactura,

La industria ha comenzado a prestar atención a esta tecnología y grandes empresas como IBM y Microsoft, por citar algunas, tienen en su poder laboratorios de agentes inteligentes que desarrollan sus propios productos de los cuales algunos han sido lanzados al mercado, fundamentalmente agentes de usuario e Internet. En cambio se necesita que este paradigma siga en expansión, por lo que es crucial que se desarrollen técnicas de análisis y diseño con el mismo, lo cual posibilitará desarrollar aplicaciones genéricas.

La introducción de la tecnología de agentes en la industria requiere de metodologías que asistan en todas las fases del ciclo de vida del sistema de agentes. La necesidad del empleo de estas técnicas será más importante a medida que el número de agentes de los sistemas aumente. Aunque en la comunidad Multi-Agente se ha mencionado que es necesario el desarrollo y aplicación de metodologías [5]. De esta forma cada incursión en un nuevo campo supone la aplicación de una metodología de aplicación adoptada por el desarrollador en cuestión; lo cual se debe a la no existencia de estándares que garanticen la extensión de la Ingeniería del Software a las áreas a las que se aplica [2]

Sin lugar a dudas estamos frente a un notable auge en la aplicación de la tecnología de agentes para el desarrollo de nuevos servicios que se encuentran sólidamente dotados de características de personalización en entornos distribuidos como es el caso de Internet.

1.4 Metodologías para el diseño de SMA

“La producción de software actual no se concibe sin un proceso de diseño controlable, documentado y reproducible a su vez por los diversos desarrolladores, que permita definir etapas evolutivas sobre la base de abstracciones y de herramientas.”

Nick Jennings [10].

Una metodología según define Rumbau 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” [8]

Las primeras metodologías que se desarrollaron tuvieron una concepción muy cercana al paradigma orientado a objetos y en algunas de ellas se proponía la utilización de UML. Estas propuestas reportan limitaciones ya que los agentes no son especificaciones de objetos, son mucho más que eso, lo que significa que hay que buscar otras alternativas que contemplen la mayor parte de las características de este nuevo paradigma. En los últimos años han aparecido investigaciones que tratan de presentar metodologías propias para el desarrollo de SMA, a partir de modificaciones de propuestas existentes y otras que presentan concepciones completamente nuevas para desarrollo de software [3]

Es necesario destacar que una metodología para el desarrollo del software debe satisfacer en su medida las etapas de desarrollo tales como: análisis, diseño, desarrollo, prueba y mantenimiento de forma jerárquica.

Por otra parte las metodologías tradicionales no toman en cuenta las necesidades de diseño de las arquitecturas orientadas a agentes, como puede ser la especificación de tareas, la motivación de los componentes del sistema y la comunicación mediante lenguajes específicos.

En la actualidad encontramos diversas metodologías que encaminan el desarrollo de un Sistema Multi-Agente, las cuales a su vez no presentan aplicación real por parte de los desarrolladores y sus deficiencias vienen ilustradas por la incapacidad de cubrir totalmente todas las etapas de desarrollo de un software. Entre las metodologías más importantes tenemos: GAIA [1] que propone cómo realizar un

análisis basado en roles del Sistema Multi-Agente, Ingenias [7] que proporciona un conjunto de herramientas para modelar y generar código de Sistemas Multi-Agentes, MaSE [6] que además propone agentes como extensiones de objetos y proporciona la herramienta AgentTool para análisis, diseño e implementación y Mas-CommonKADS [5] la cual extiende la metodología CommonKADS para sistemas expertos a agentes utilizando estructuración orientada a objetos y lenguajes de especificación de protocolos.

Metodologías más conocidas:

GAIA

MaSE

MAS-CommonKADS

AGILE PASSI

MESSAGE

ZEUS

Vowel Engineering.

Las principales actividades de una metodología son:

La definición y descripción del problema que se desea resolver.

El diseño y descripción de una solución que se ajuste a las necesidades del usuario.

La construcción de la solución.

La prueba de la solución implementada.

Entre los requisitos que debe cumplir una metodología se pueden citar los siguientes:

La metodología está documentada: el procedimiento de uso de la metodología está contenido en un documento o manual de usuario.

La metodología es repetible: cada aplicación de la metodología es la misma.

La metodología es enseñable: los procedimientos descritos tienen un nivel suficientemente detallado y existen ejemplos para que personal cualificado pueda ser instruido en la metodología.

La metodología está basada en técnicas probadas: la metodología implementa procedimientos fundamentales probados u otras metodologías más simples.

La metodología ha sido validada: la metodología ha funcionado correctamente en un gran número de aplicaciones.

La metodología es apropiada al problema que quiere resolverse.

La definición de una metodología de ingeniería software normalmente no parte de cero, sino que es un proceso de refinamiento, añadiendo los nuevos aspectos y perspectivas de los sistemas y lenguajes e integrando los ingredientes exitosos de metodologías previas.

1.5 MaSE: Ingeniería de Sistemas Multi-Agentes

La metodología MaSE (Multi-Agent Systems Software Engineering) se concibe como una abstracción del paradigma orientado a objetos donde los agentes son especializaciones de objetos. En lugar de simples objetos, con métodos que pueden invocarse desde otros objetos, los agentes se coordinan unos con otros vía conversaciones y actúan proactivamente para alcanzar metas individuales y del sistema [6]

MaSE es una metodología que se desarrolló en el Air Force Institute of Technology. La misma tiene el propósito de cubrir todas las etapas en el proceso de construcción de un SMA, partiendo de la especificación del mismo hasta su implementación, no obstante a ello este es un objetivo que queda insatisfecho.

La metodología es iterativa lo que le permite al diseñador moverse sin dificultades entre las etapas. Cada etapa transcurrida, se le pueden añadir nuevos detalles en dependencia de lo que se necesite y al final es obtenido un producto completo e independiente.

Esta metodología es independiente del dominio y presenta un modelado basado en roles. Dispone de un lenguaje de especificación basado en UML+OCL (Object Constraint Language), lo cual evidencia un gran acercamiento a los conceptos orientados a objetos, asumiendo al agente como un objeto que puede ser o no inteligente. Tiene además la habilidad de localizar cambios a lo largo del proceso: cada objeto creado durante las fases de análisis y diseño puede ser localizado delante o atrás, durante los diferentes pasos con otros objetos relacionados.

MaSE se sustenta sobre el agentTool, el cual constituye un software gráficamente-basado, totalmente interactivo que diseña herramientas para esta metodología. El agentTool apoya el análisis y diseña en cada uno de los siete caminos de MaSE. Esta metodología, tanto como el agentTool, es independiente de cualquier arquitectura particular de agentes, lenguaje de programación, o frameworks de comunicación.

MaSE es una especificación de la más tradicional ingeniería del software. La misma consta de dos fases básicas que son: Análisis y Diseño.

La fase de Análisis tiene tres pasos:	La fase de Diseño tiene cuatro pasos:
○ Captura de Objetivos	○ Crear las Clases de Agentes
○ Aplicación de Casos de Uso	○ Construir las Conversaciones
○ Refinamiento de Roles.	○ Configurar Clases de Agentes
	○ Desarrollar el Diseño

Tabla 1: Fases de MaSE

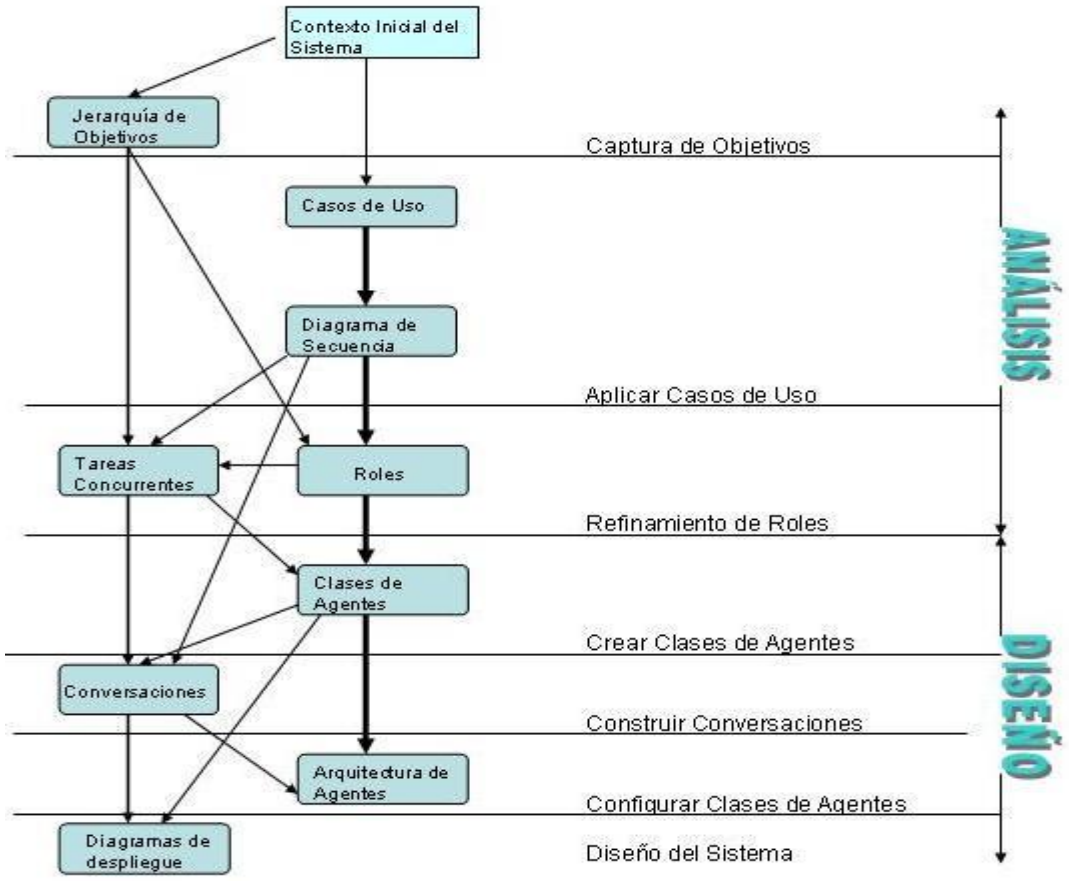

Figura 2: Metodología MaSE

1.5.1 Fase de Análisis

El propósito de esta fase consiste en confeccionar un set de roles que lleven implícito lo que el sistema necesita desarrollar, o sea, crear un grupo de tareas que definan y describan lo que el sistema debe hacer para reunir sus requisitos globales.

Un rol es más bien una entidad que realiza una función dentro del sistema. En MaSE, cada rol es responsable de llevar a cabo o contribuir al cumplimiento de las metas específicas del sistema así como también las sub-metas. MaSE contempla roles análogos o roles que pueden ser jugados por actores en una obra o por miembros de una estructura típica de compañía.

Captura de Objetivos

La primera fase en MaSE es la captura de Objetivos, los cuales toman la especificación del sistema inicial representadas en un Diagrama de Jerarquía de Objetivos y la transforman en un conjunto de objetivos que requiere el sistema. El contexto inicial del sistema es la colección de algo que se le da al analista como un punto de comienzo para el análisis. Se asume que este contexto incluye una especificación de requisitos de software que tiene un juego bien definido de requisitos funcionales. Los requisitos funcionales le dicen al analista los servicios que el sistema debe proporcionarle y cómo debe o no comportarse basado en entradas al sistema y su actual estado. Las metas incluyen lo que el sistema está intentando lograr y generalmente permanecen constantes a lo largo del análisis y proceso del plan

Se tienen dos pasos en la captura de metas: identificación de los objetivos y estructuración de los objetivos.

Un objetivo es definido como una abstracción de los requisitos detallados del sistema. Se llevan a cabo por los roles y capturan qué está intentando conseguir el sistema así como plasman los requisitos críticos del mismo. El objetivo general de cada sistema consiste en llevar a cabo los deseos del usuario.

Identificación de los objetivos

- El analista debe ser capaz de especificar objetivos tan abstractamente como sea posible sin perder la esencia del requisito.
- Los objetivos son identificados separando la esencia del set de requerimientos.
- Los requerimientos pueden incluir documentos técnicos detallados, historia de usuarios o especificaciones de gobierno formalizadas.
- Se elaboran escenarios a partir de la especificación inicial y se describe el objetivo de ese escenario.
- Los objetivos son poco propensos a cambiar. No sucede así con los pasos que se detallan y las actividades involucradas en el cumplimiento de éstos.
- Los objetivos proporcionan los fundamentos para el modelo de análisis. Los roles y tareas definidos en pasos posteriores deben soportar cada uno de estos objetivos.

Estructuración de los objetivos

Los objetivos son analizados y estructurados en una forma que puede ser llevada a niveles superiores y usados en el diseño de fases de la metodología. Los mismos se estructuran a través de un diagrama de Jerarquía de Objetivos el cual permite identificar el objetivo global del sistema y descomponer en sub objetivos: cada sub objetivo define qué debe hacerse para cumplir el objetivo padre.

Una vez que se han capturado las metas y han sido explícitamente declaradas, éstas llegan a ser menos propensas a cambiar que cualquiera de los pasos y las actividades que se involucraron para el logro de las mismas. Es válido saber que una descomposición de objetivos no se centra en cómo debería llevarse a cabo un objetivo. Además no es una “descomposición funcional”, una descomposición funcional produciría como resultado un conjunto de pasos para conseguir el objetivo (“como”).

Existen cuatro tipos de objetivos:

1. Resumen: a partir de un conjunto de objetivos al mismo nivel obtener un objetivo padre común.

2. Combinado: varios subobjetivos que son idénticos o muy similares pueden agruparse (sucede cuando un objetivo es subobjetivo de dos objetivos diferentes)

3. Dividido: objetivo con un conjunto de subobjetivos, que tomados colectivamente cumplen el objetivo

4. No funcional.

Aplicación de Casos de Uso

La aplicación de Casos de Uso guarda mucha semejanza con la realización de los Casos de Uso en RUP ya que esta actividad se realiza por medio de los Diagramas de Interacción de UML pero con mucho menos detalle. Ésta tiene como objetivo fundamental la producción de Casos de Uso para ayudar al analista a identificar un conjunto inicial de roles del sistema.

Se puede decir que la conversación entre agentes es la base fundamental de los sistemas Multi-Agentes, o sea, la misma constituye la columna vertebral de dichos sistemas. A partir de ello, esta fase se propone trazar el camino hacia la construcción de las antes mencionadas conversaciones y crear además Casos de Uso para disminuir las complicaciones.

Esta fase captura Casos de Uso de los requerimientos iniciales del sistema y los reestructura como un diagrama de secuencia. Este diagrama de secuencia por tanto dibuja la secuencia de mensajes entre los múltiples roles de agente.

Por otra parte el analista es quien deduce los casos del uso. El mismo debe ser capaz de capturar tanto casos de uso positivos (describe lo que sucedería durante el funcionamiento normal del sistema) como negativos (en caso de interrupción o error).

Los Casos de Usos constituyen descripciones narrativas de una secuencia de eventos que define el comportamiento deseado del sistema. Ellos son los ejemplos que evidencian cómo el usuario piensa que el sistema debe comportarse. Una parte de Aplicar Casos de Uso, Crear Casos de Uso, puede realmente aportar más información o aclarar la ya existente sobre las metas del sistema. Si esto llegase a suceder, el analista inmediatamente debe retornar al Diagrama Jerárquico de Objetivos y agregar o modificar la información que tiene en el mismo.

Los Casos de Usos pueden existir como parte del contexto inicial del sistema o pueden haber sido extraídos por el analista. Él los puede sacar de las especificaciones de requisitos, historias del usuario, o cualquier otra fuente disponible. Contar con un gran número de casos de uso, puede resultar muy beneficiosos para entender el sistema. La meta de crear casos del uso es identificar caminos de comunicación, no definir todas las posibles combinaciones de eventos y datos en el sistema. El analista debe intentar recoger la mayor cantidad de casos del uso para cubrir tanto como sea posible sucesiones de evento sin repetir la misma sucesión muchas veces con diferentes datos o eventos. En general, el analista debe esforzarse por mostrar cómo cada meta puede lograrse.

La transformación de los casos del uso a los Diagramas de Sucesión es relativamente directa. Para ayudar a determinar las comunicaciones reales requeridas dentro del sistema, se reestructura el caso de uso en el Diagramas de Secuencia, y así quedan representados una sucesión de eventos entre los

diferentes roles. Las entidades individuales nombradas en el caso del uso corresponden a los roles, mientras cualquier tipo de comunicación o información que pasa entre las entidades del caso de uso se transforman en eventos.

Los eventos capturan el flujo de información entre entidades, y la secuencia de los eventos está basada en la descripción del caso de uso. En este diagrama se define además, el mínimo de comunicación que debe tener lugar entre los roles. Si un mensaje es atravesado por dos roles, entonces debe haber un camino de correspondiente comunicación entre ellos. Un camino de comunicación entre roles separando clases de agentes significa que debe existir una conversación entre dos clases agentes para pasar el mensaje. El agente clase que juega el rol que comenzó la comunicación se vuelve el iniciador de esa conversación mientras que la clase de agente que recibe es quien da la respuesta. Mayormente se crea una sola secuencia de un Caso de Uso. Si hay más posibles escenarios, múltiples diagramas de secuencia son creados. Los roles identificados en este paso forman el conjunto inicial de roles a utilizar para definir los roles del sistema en el siguiente paso.

Refinamiento de Roles

El correspondiente paso es para transformar los objetivos estructurados del Diagrama Jerárquico de Objetivos y los Diagramas de Secuencia en una forma más útil para la construcción de los sistemas Multi-Agentes que son los roles. Como tal un rol es una entidad que desempeña alguna función dentro del sistema.

Cada rol es responsable de conseguir, o ayudar a conseguir objetivos específicos del sistema o sub objetivos. Los roles son además, bloques de construcción que se usan para definir las clases de agentes y capturar los objetivos del sistema durante la fase de diseño. Los objetivos del sistema son responsables de llevar a cabo el aseguramiento de que cada meta esté ligada con un rol, y que cada rol sea jugado por una clase agente. Es válido resaltar que existen situaciones en las que un único rol será responsable de múltiples objetivos.

Las interfaces con recursos externos o internos generalmente requieren un rol separado para que actúe como interfaz con el resto del sistema. Se suele considerar un usuario humano como un recurso externo,

por lo que se crea un rol especifico para encapsular la interfaz de usuario. Las metas similares o compatibles pueden ser combinadas en un solo rol por conveniencia o eficiencia. Las metas comunes implican a menudo roles que pueden ser rehusados antes de previos esfuerzos. Los roles, tareas y protocolos se capturan en el Modelo de Rol. Cada tarea es responsable de un objetivo.

Los roles son denotados por rectángulos, mientras las tareas del rol son denotadas por óvalos adjuntados al rol. La descripción detallada de la definición de una tarea es proveída por la vía del Diagrama de Tareas. Las líneas entre las tareas denotan la relación de iniciador/respuesta del protocolo con la flecha apuntando desde el iniciador a la respuesta. Las líneas fuertes indican comunicaciones par-a-par que generalmente se lleva a cabo como protocolos de comunicación externos. Los protocolos externos llevan consigo mensajes que pasan entre roles que pueden llegar a ser mensajes reales si sus roles terminan siendo implementados en agentes separados.

Estos protocolos se derivan del diagrama de secuencia desarrollado en el paso anterior. Las líneas golpeadas denotan comunicación entre las tareas concurrentes llevadas a cabo dentro del mismo rol. Una línea es arrojada si sus protocolos denotan comunicaciones que sólo ocurren dentro de la misma instancia del rol. Los roles no pueden compartirse o duplicar tareas. Las tareas compartidas deben ser puestas en diferentes roles, los que pueden ser combinados en varias clases de agentes en la fase de diseño.

Diagrama de Tareas Concurrentes

Después de creados los roles, las tareas son asociadas con cada uno que describa la conducta que el rol debe exhibir para lograr sus metas con éxito. Un solo rol puede tener múltiples tareas concurrentes que definen el comportamiento del rol requerido. Las tareas coexistentes son especificadas gráficamente usando un estado finito autómata.

Existen dos tipos de tareas: persistente y transitoria.

Persistente:

- Tiene una transición nula del estado de la salida al primer estado (no presenta un evento que inicie su ejecución)
- Comienza cuando el agente es inicializado y continúa ejecutándose hasta que éste termine por sí mismo o hasta que la tarea llegue al final de su estado.

Transitoria:

- Presenta una señal específica en la transición del estado inicial.
- No se ejecuta cuando los agentes comienzan, sino que espera hasta que la señal sea recibida por el agente.
- Con este tipo de tareas es posible tener múltiples y concurrentes tareas ejecutándose al mismo tiempo.

Las tareas concurrentes consisten en estados y transiciones, los cuales son similares a los estados y transiciones de la mayoría de los modelos finitos autómatas. Los estados abarcan el proceso que continúa interno al agente mientras que las transiciones permiten comunicación entre agentes o entre tareas. Una transición consiste en un estado de la fuente, estado destino, señal, condición preservada y transmisiones. Para ello se usa la sintaxis: trigger [guard] ^ transmission(s). Las tareas concurrentes tienen actividades predefinidas que les permiten movilidad y tiempo. El resultado de la actividad en movimiento especifica que el agente tiene que trasladarse a una nueva dirección.

1.5.2 Fase de Diseño

La fase de diseño consta de cuatro pasos esenciales los cuales son: Crear Clases de Agente en el que el diseñador asigna roles a los tipos de agentes específicos, Construir Conversaciones donde se definen las conversaciones entre agentes, Configurar Clases de Agentes donde la arquitectura interna de éstos, así como los procesos de razonamiento de las clases de agentes son diseñados y por último el Diseño del Sistema donde el diseñador define el número real y situación de agentes en el despliegue del sistema.

Crear las Clases de Agentes

Las clases de agentes se identifican a partir de los roles que fueron definidos en la fase de análisis donde se asignan roles a cada clase de agentes y se documentan en el Diagrama de Clases de Agentes. Este diagrama describe la organización global del sistema de agentes utilizando clases de agentes y conversaciones entre ellas. Presenta una notable diferencia del Diagrama Orientado a Objetos ya que las clases constituyen roles, no atributos y métodos, así como que las relaciones entre clases son las denominadas conversaciones. El producto final de esta fase es el diagrama de Clases de Agentes, el cual es el primer objeto de diseño en MaSE que describe el Sistema Multi-Agente íntegramente.

Una clase de agente es una plantilla para un tipo de agente en el sistema y es análogo a una clase de objeto en orientación-objeto. Un agente es un caso real de una clase agente. Durante este paso, se definen clases agentes en términos de los roles que ellos jugarán y las conversaciones en las que ellos deben participar.

Durante esta fase, las clases de agentes están estructuradas en dos componentes: roles y conversaciones y los detalles internos serán agregados en fases posteriores.

Las conversaciones de una clase del agente son aquellas en los que él participa, ya sea como iniciador o como el que emite una respuesta.

En el diagrama de Clases de Agentes, las relaciones definen conversaciones que se sostienen entre las clases del agente. Por consiguiente, el propósito primario de esta fase es identificar las clases del agente que "unen" cada lado de una conversación, no entrar en demasiados detalles sobre las mismas ya que esto será llevado a cabo en la Construcción de Conversaciones.

Los roles son el basamento en el que se diseñan las clases del agente. Éstos corresponden al juego de metas del sistema definido en la fase de Análisis, ellos forman un puente de lo que el sistema está intentando archivar: la fase de Análisis y metas. El analista puede cambiar fácilmente la organización y asignación de roles entre las clases del agente durante el diseño, ya que los roles pueden ser manipulados modularmente.

Es importante tener en cuenta que al determinar qué roles se tienen para establecer una combinación entre ellos, el tamaño y frecuencia de comunicaciones son esenciales, no se puede limitar a sólo el

número de caminos de comunicación. Un volumen elevado en la conversación entre roles implica que esos roles deber ser parte de la misma clase de agentes. Dos roles con amplios requisitos computacionales es mejor que sean jugados por diferentes clases de agentes por lo que ellos deben ser ejecutados en diferentes CPU.

Construir Conversaciones

Este resulta el próximo paso en el cual el aún no se tienen definidas las comunicaciones entre agentes, sólo están declaradas. La meta de este paso es precisamente detallar las comunicaciones. Los detalles internos de las tareas concurrentes son indispensables en esta persecución. El diseñador puede realizar los pasos de construir las conversaciones y ensamblar las clases en forma paralela, los dos están estrechamente relacionados. Una conversación en MaSE define un protocolo de coordinación entre dos agentes. Específicamente, una conversación consiste en dos Diagramas de Clase de Comunicación, uno para el iniciador y otro para el que emite la respuesta. Un Diagrama de Clases de Comunicación es un par de máquinas de estados finitos que definen la comunicación entre dos clases de agentes participantes.

Por otra parte el diseño detallado de conversaciones es derivado de las tareas concurrentes asociadas con los roles. Básicamente cada tarea que define una comunicación externa crea una o más conversaciones. Si todas las comunicaciones dentro de la tarea tienen un solo rol, o juego de roles que hayan sido trazados a una única clase del agente, la tarea puede ser trazada directamente a una sola conversación.

Una vez que se ha capturado toda la información de los modelos de tareas concurrentes como parte de las conversaciones, el diseñador debe asegurarse de que otros aspectos, tales como robustez y falla de tolerancia se tengan en cuenta.

Ensamblar Clases de Agentes

En este paso, se realiza el diseño interno de cada clase de agentes donde se define la arquitectura de agente y además se definen los componentes de dicha arquitectura.

Un diseñador no puede definir componentes desde el principio o usar componentes pre-existentes. Los componentes por su parte pueden presentar arquitecturas sub-alternas que éstas a su vez posean componentes.

Desarrollar el Diseño

La última etapa de MaSE define la configuración del sistema actual que va a ser implementado, esta es la etapa más fácil ya que la mayoría de las cosas se hicieron en las anteriores. Todas estas fases se llevan a cabo de forma iterativa e incremental, ya que se van refinando todos los productos resultantes. Se usan diagramas de despliegue para mostrar número, tipo y localización de las instancias de agentes en el sistema. Las cajas tridimensionales son agentes, y las líneas que unen representan conversaciones entre ellos.

Como productos de estas etapas, MaSE espera: diagramas de secuencia para especificar interacciones, diagramas de estados para representar procesos internos a las tareas y modelar interacciones, descomposición del sistema (agente) en subsistemas (componentes del agente) e interconexión de los mismos (definición de la arquitectura del agente).

1.6 MAS-CommonKADS: Ingeniería de Sistemas Multi-Agentes

Esta metodología extiende CommonKADS aplicando ideas de metodologías orientadas a objetos para su aplicación a la producción de Sistemas Multi-Agentes [5]. La metodología CommonKADS gira alrededor del modelo de experiencia y está pensada para desarrollar sistemas expertos que interactúen con el usuario. De hecho considera sólo dos agentes básicos: el usuario y el sistema. MAS-CommonKADS extiende los modelos de CommonKADS para tener en cuenta la posibilidad de que dos o más componentes del sistema interactúen.

MAS-CommonKADS ha sido la primera en plantear un desarrollo de Sistema Multi-Agente integrado con un ciclo de vida de software, concretamente el espiral dirigido por riesgos.

MAS-CommonKADS propone los siguientes modelos para el desarrollo de Sistemas Multi-Agentes:

- Modelo de Agente (AM): específica las características de un agente: sus capacidades de razonamiento, habilidades, servicios, sensores, efectores, grupos de agentes a los que pertenece y clase de agente. Un agente puede ser un agente humano, software, o cualquier entidad capaz de emplear un lenguaje de comunicación de agentes.
- Modelo de Organización (OM): es una herramienta para analizar la organización humana en que el sistema Multi-Agente va a ser introducido y para describir la organización de los agentes software y su relación con el entorno.
- Modelo de Tareas (TM): describe las tareas que los agentes pueden realizar: los objetivos de cada tarea, su descomposición, los ingredientes y los métodos de resolución de problemas para resolver cada objetivo.
- Modelo de la Experiencia (EM): describe el conocimiento necesitado por los agentes para alcanzar sus objetivos. Sigue la descomposición de CommonKADS y reutiliza las bibliotecas de tareas genéricas.
- Modelo de Comunicación (CM): describe las interacciones entre un agente humano y un agente software. Se centra en la consideración de factores humanos para dicha interacción.
- Modelo de Coordinación (CoM): describe las interacciones entre agentes software.
- Modelo de Diseño (DM): mientras que los otros modelos tratan del análisis del sistema Multi-Agente, este modelo se utiliza para describir la arquitectura y el diseño del sistema Multi-Agente como paso previo a su implementación.

MAS-CommonKADS tiene una fase de conceptuación donde se analiza el sistema desde el punto de vista del usuario del sistema. Es una fase un poco informal, donde se propone un análisis centrado en el usuario para la captura de requisito. El empleo de los casos de uso es la técnica por la que se apuesta en esta fase. Ayudar a probar el sistema.

- Análisis: determinación de los requisitos de nuestro sistema partiendo del enunciado del problema. Durante esta fase se desarrollan los siguientes modelos: organización, tareas, agente, comunicación, coordinación y experiencia.

- Diseño: determinación de cómo los requisitos de la fase de análisis pueden ser logrados mediante el desarrollo del modelo de diseño. Se determinan las arquitecturas tanto de la red Multi-Agente como de cada agente.
- Codificación y prueba de cada agente.
- Integración: el sistema completo es probado.
- Operación y mantenimiento.

1.6.1 Conceptuación en MAS-CommonKADS

MAS-CommonKADS incorpora la técnica de casos de uso en el ciclo de vida de CommonKADS. La notación gráfica empleada se basa en la notación propuesta por Regnell [9], que normaliza las interacciones de los casos de uso empleando la técnica de descripción formal MSC (Diagrama de Secuencia de Mensajes). Esta notación ha sido modificada para distinguir entre usuarios software y usuarios humanos del sistema..

La técnica de casos de uso parece adecuada para el desarrollo de sistemas basados en conocimiento y, en particular, de Sistemas Multi-Agentes. El empleo de casos de uso permite obtener los requisitos de los usuarios y generar casos de prueba.

Ciclo de desarrollo de los Casos de Uso

En este apartado se presentan el conjunto de actividades que se suceden en el desarrollo de la fase de conceptuación.

Actividad: Identificación de los actores

- Descripción: identificación de los agentes externos al sistema y que interactúan con él.
- Técnicas: los actores se identifican partiendo de los posibles usuarios del sistema.
- Producto: ninguno.

Actividad: Descripción de los actores

- Descripción: descripción de los agentes externos al sistema y que interactúan con él.
- Técnicas: los actores se describen empleando la siguiente plantilla textual:
- Actor entidad. Nombre de una entidad que interactúa con el sistema.
- Nombre ranura, texto, (Obl.) . El nombre del agente es una cadena de texto corta. El nombre debe ser único dentro de la fase de conceptuación, ya que es el principal mecanismo de referencia.
- Descripción ranura, texto, (Obl.). Descripción de las características y usos del sistema por parte del actor.
- Producto: diccionario de actores.

Actividad: Identificación de los casos de uso

- Descripción: identificación de los posibles usos que da cada actor al sistema.
- Producto: ninguno.

Actividad: Descripción de los casos de uso

- Descripción: identificación de los posibles usos que da cada actor al sistema.
- Técnicas: los casos de uso pueden describirse de forma textual y gráfica.
- Producto: plantillas de casos de uso.

Notación de los casos de uso

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

Notación textual

Se muestra a continuación la notación textual propuesta por Rumbaugh (J.Rumbaugh 1995). Los actores se describen con plantillas diferentes de las de los agentes para facilitar la reutilización de los casos de uso, sin dificultar por ello la trazabilidad.

Caso-de-uso entidad. Interacción o uso de un elemento externo con el sistema.

nombre ranura, texto, (Obl.). El nombre del caso de uso es una cadena de texto corta. El nombre debe ser único dentro de la fase de conceptuación, ya que es el principal mecanismo de referencia para especificar las relaciones entre casos de uso.

resumen ranura, texto, (Obl.). Breve descripción de la secuencia de interacciones entre los actores y el sistema.

actores ranura, texto, (Obl.), muchos a muchos. Actores que participan en el caso de uso. Debe ser un valor definido en la ranura nombre de una entidad Actor.

precondiciones ranura, texto, (Opt.), muchos a muchos. Condición que debe ser cierta para que el caso de uso pueda darse.

descripción ranura, texto, (Obl.). Descripción detallada e informal de las interacciones, referenciando las posibles excepciones.

excepciones ranura, texto, (Opt.), muchos a muchos. Descripción de cada excepción y de sus ocurrencias.

postcondiciones ranura, texto, (Opt.), muchos a muchos. Descripción de las postcondiciones del caso de uso.

Notación gráfica

La notación gráfica de los casos de uso sigue la propuesta por Regnell [9] que extiende la notación de Jacobson [5].

La notación para los casos de uso distingue entre agentes software y agentes humanos, tal como se muestra en la siguiente figura.

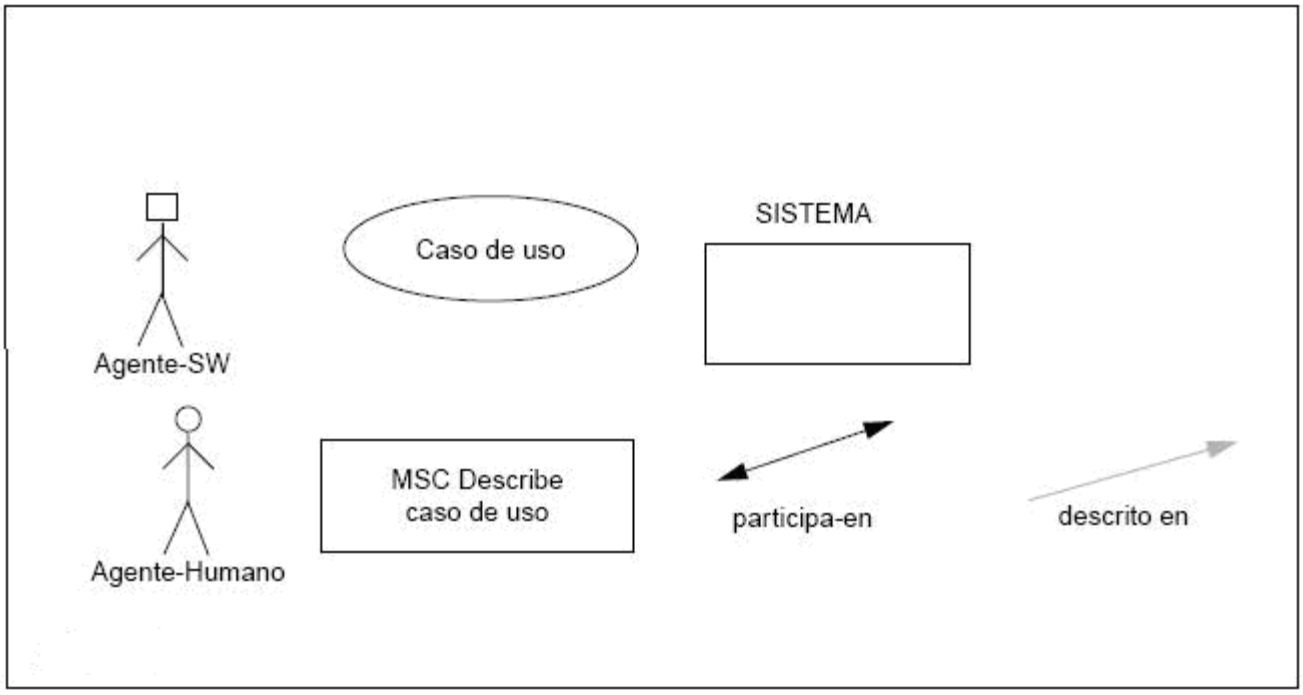

Figura 3: Leyenda

Actividad: Descripción de las intervenciones de los casos de uso

- Descripción: descripción de las intervenciones de los casos de uso.
- Técnicas: la descripción de las intervenciones se realizará de forma textual en cada caso de uso y de forma gráfica empleando MSCs
- Producto: ninguno.

Actividad: Descripción de las relaciones entre casos de uso

- Descripción: descripción de las relaciones entre los casos de uso.
- Producto: ninguno.

Modelos para el desarrollo de Sistemas Multi-Agente de MAS-CommonKADS.

Modelo de Agente

Antecedentes históricos

El modelo de agente de MAS-CommonKADS tiene su origen en el modelo de comunicación de CommonKADS. Dicho modelo tenía sus bases en KADS-1, que definía un modelo de cooperación entre el sistema y el usuario para resolver una tarea, pero no modelaba explícitamente a los agentes involucrados en la tarea.

El modelo de agente, que es el modelo central de la metodología, y permite identificar y describir las características de los agentes del sistema.

El modelo de agente de MAS-CommonKADS sirve de enlace entre el resto de modelos del análisis, mediante el modelado de las capacidades y restricciones de los agentes del sistema Multi-Agente.

Fundamentos del modelo de agente

¿Qué problemas de IAD deben ser resueltos en el desarrollo del modelo de agente?

El modelo de agente sirve para especificar las características de los agentes involucrados en la resolución de un problema, siguiendo la perspectiva de agente. El modelo de agente está pensado para recoger los requisitos que debe tener un agente para poder realizar las tareas (responsabilidades) asignadas. No asumiremos, por tanto, ninguna arquitectura de agente concreta. Esta decisión será tomada en el modelo de diseño.

¿Cuándo es el modelo útil, y cuándo debe ser incluido en el conjunto de modelos?

El modelo de agente es el modelo central de MAS-CommonKADS y debe ser incluido siempre. Es considerado como modelo motor del proyecto. El propósito de su construcción es:

- La descripción de agentes existentes.

- La descripción de los agentes que se van a desarrollar.
- Permite enlazar en cada agente las tareas, conocimientos (e inferencias) y comunicaciones descritas en el resto de modelos de análisis.
- Restricciones: ¿qué no puede ser modelado?
- Las siguientes restricciones, derivadas del modelo de comunicación, han sido identificadas:
- El modelo de agente no permite modelar requisitos de las habilidades necesarias para realizar una comunicación humano-humano.

Se asume que el sistema Multi-Agente no está siendo empleado para trabajo cooperativo asistido por ordenador, en el que no se han realizado casos de estudio, y no se ofrecen guías para su aplicación.

Estructura del modelo

Constituyentes del modelo

El modelo de agente se basa en extender el modelo homónimo de CommonKADS. El constituyente central del modelo es el agente. Cada ejemplar del constituyente agente tiene relaciones uno-a-uno con los constituyentes de capacidades, y muchos-a-muchos con los constituyentes servicio y objetivo. La definición de los constituyentes se realiza con plantillas que son rellenadas durante el análisis y están compuestas de atributos (ranuras, slots). Los atributos pueden ser obligatorios (Obl) u optativos (Opt).

Los rasgos más relevantes de cada agente y las relaciones estáticas entre los agentes pueden observarse en la representación gráfica propuesta en el modelo de organización, mientras que las relaciones dinámicas se muestran en los modelos de comunicación y coordinación.

Ciclo del desarrollo del modelo

Normalmente pueden desarrollarse tantos ejemplares de los modelos de MAS-CommonKADS como se deseen. En el caso del modelo de agente, sólo se contempla el posible desarrollo de dos ejemplares: un ejemplar del modelo del sistema antes de la introducción del sistema Multi-Agente y otro ejemplar del modelo tras la introducción del sistema Multi-Agente. Estos ejemplares del modelo deben realizarse siguiendo el conjunto de estados de modelado definidos previamente.

Situaciones de bajo riesgo para el desarrollo del modelo de agente.

Las situaciones de bajo riesgo en el desarrollo del modelo de agente coinciden con las definidas en el modelo homónimo de CommonKADS. La presencia de estas situaciones de bajo riesgo, en especial de las últimas enunciadas, pueden aconsejar la selección de otro paradigma para desarrollar el sistema (orientado a objetos, a funciones, etc.). Vamos a clasificar estas situaciones (junto con algunas nuevas) en dos grupos, dependiendo de si consideramos una relación hombre-máquina o entre sistemas informáticos:

a) Situaciones de bajo riesgo en el desarrollo del modelo de agente debido a la interfaz de usuario. Estas situaciones tendrán en cuenta los riesgos procedentes del análisis de factores humanos.

- No se considera importante si los usuarios deben entrenarse para utilizar el sistema o los conocimientos que adquieran a través de su uso.
- El sistema Multi-Agente interactúa con un sólo usuario.
- El grupo de usuarios se ha definido y comprendido bien en el modelo de organización y no presenta ninguna habilidad ni discapacidad que puedan afectar al uso del sistema.
- El usuario toma la iniciativa en la utilización del sistema. No es necesario que el sistema planifique cómo conseguir los objetivos del usuario.

b) Situaciones de bajo riesgo en el desarrollo del modelo de agente debido a la interacción con otros sistemas informáticos.

- El sistema Multi-Agente que se va a desarrollar no necesita interactuar con ningún sistema informático, o el usuario no utilizará simultáneamente el nuevo sistema y el anterior.
- El sistema Multi-Agente interactúa con un único sistema informático, mediante una invocación de funciones.

c) Situaciones de bajo riesgo debido a la falta de distribución en el problema

- El sistema no está distribuido geográficamente, ni tiene que interactuar con otros programas en una red.
- El sistema no está distribuido lógicamente.

- El sistema no presenta distribución de conocimiento.

Desarrollo estándar del modelo de agente

Para la aplicación de MAS-CommonKADS se deberá dar alguna situación de alto riesgo del desarrollo del modelo de agente, que es considerado modelo motor del proyecto.

Los estados hitos del desarrollo estándar del modelo de agente son:

1. Identificación inicial de los agentes. Se debe identificar un conjunto inicial de los agentes del sistema y de sus capacidades. Esta identificación inicial será la entrada del modelo de coordinación y comunicación así como la base para la asignación de tareas (objetivos) en el modelo de tareas.

2. Identificación de clases y grupos de agentes. Basándose en la similitud de varios agentes es posible realizar una generalización y definir una clase. De forma semejante, basándose en la colaboración de varios agentes se puede definir un grupo con propiedades comunes.

3. Descripción de los agentes, clases y grupos. A medida que otros modelos se van desarrollando el modelo de agente es mejor comprendido y se va describiendo.

4. Descripción de los objetivos y servicios. El modelo de tareas y de coordinación permiten identificar objetivos y servicios, respectivamente, que son descritos en este modelo.

5. Validación de relaciones entre el modelo de agente y el resto de modelos.

Dependencias en la construcción del modelo

Modelo de Tareas

La relación entre el modelo de agente y el modelo de tarea es clara: las tareas de las hojas del modelo de tareas son ejecutadas por uno o varios agentes.

El desarrollo de ambos modelos suele darse en paralelo: tras una identificación inicial de los agentes, y de sus capacidades de razonamiento, se realiza una descomposición de tareas y una posterior asignación de las mismas. La descomposición de tareas puede dar lugar a la identificación de nuevos agentes. El empleo de agentes previamente desarrollados puede acortar la descomposición de las tareas, pues una vez que se llega a una tarea que ya es resoluble por un agente no se necesita realizar una descomposición mayor, como regla general.

Modelo de Comunicación

El modelo de agente debe ser desarrollado con antelación al modelo de comunicación, y proporcionar las capacidades de los agentes involucrados. Una vez que el modelo de comunicación ha sido desarrollado, puede realimentar los modelos de tareas y de agente. Si las transacciones no son compatibles con la asignación de tareas, deben modificarse en ambos modelos.

Modelo de Coordinación

El modelo de agente debe ser desarrollado con antelación al modelo de coordinación, proporcionando un conjunto inicial de agentes con unas determinadas habilidades y objetivos. Como resultado del desarrollo del modelo de coordinación, puede que se considere conveniente añadir, eliminar, dividir o combinar agentes definidos en este modelo, que deberá ser modificado.

Modelo de la Experiencia

El modelo de la experiencia se emplea para describir las capacidades de razonamiento de los agentes software, pudiendo ser empleado también para modelar parte del razonamiento de los agentes humanos.

Modelo de Organización

El modelo de organización describe las relaciones estáticas entre los agentes y recursos de la organización. El modelo de agente debe haberse desarrollado con antelación al modelo de organización. El desarrollo del modelo de organización puede conllevar modificaciones en el modelo de agente.

Modelo de Tareas

En este modelo las tareas se descomponen siguiendo un acercamiento de lo general a lo particular, y descritas (las tareas) en un árbol y/o.

En este modelo se describen las actividades relacionadas para alcanzar un objetivo. Las tareas de conocimiento que se deseen implementar se detallarán en un modelo de la Experiencia, mientras que a las tareas de comunicación humana se les hará un estudio más detallado en el modelo de comunicación y para la comunicación de agentes en el modelo de coordinación.

La descripción de la tarea incluye su nombre, una breve descripción, ingredientes de entrada y salida, estructura de la tarea, frecuencia de aplicación, precondiciones y capacidad requerida de los actores.

El beneficio potencial del desarrollo de este modelo es que proporciona la documentación de las actividades de la organización antes y después de introducir el sistema Multi-Agente.

Este servicio de documentación ayuda el soporte de mantenimiento y administración de cambios en la organización y para apoyar la valoración de viabilidad del proyecto.

Modelo de Experiencia

MAS-CommonKADS desarrolla en el modelo de la experiencia las tareas que requieren conocimiento para ser llevadas a cabo y que permitirán caracterizar al agente como un sistema basado en conocimiento. El principal problema de un sistema basado en conocimiento es la adquisición del mismo, que ha dado lugar a la disciplina denominada ingeniería del conocimiento o adquisición del conocimiento.

Desarrollo del modelo de la experiencia

Se pueden distinguir los siguientes enfoques genéricos de adquisición del conocimiento:

• La adquisición del conocimiento basada en modelos: el modelado se ve como un proceso constructivo descendente, es decir, primero se selecciona o construye un modelo abstracto y después se especifica con conocimiento específico del dominio.

En realidad, estos enfoques pueden verse como los casos extremos de un continuo que va de un desarrollo con soporte débil de modelos hasta un soporte fuerte, ya que siempre se empleará algún modelo abstracto del conocimiento del dominio para poder adquirirlo, aunque dicho modelo sea débil.

La fase de adquisición del conocimiento parte de una primera fase de orientación en la que el ingeniero de conocimiento se familiariza con el dominio, a través del estudio de publicaciones relacionadas con el campo, entrevista con los expertos, etc. En CommonKADS esta fase se realiza en el modelo de organización, y su principal cometido consiste en identificar los requisitos o funcionalidad deseada, delimitar el sistema basado en conocimiento del resto de sistemas y determinar su viabilidad, pudiendo identificar además el léxico básico o términos del dominio.

Se pueden distinguir las siguientes actividades en la adquisición de conocimiento basada en el modelo construcción del esqueleto del modelo (skeletal model), instanciación, compilación y refinamiento del modelo.

La construcción del esqueleto del modelo consiste en la creación o selección de una especificación abstracta del conocimiento requerido para realizar la tarea deseada. Esta especificación abstracta puede definirse con diferente granularidad.

La instanciación consiste en “rellenar” el esqueleto del modelo con conocimiento del mismo para generar una base de conocimiento completa. En esta fase se emplean técnicas de adquisición, que son soportadas por herramientas de adquisición del conocimiento. Esta descripción del conocimiento del dominio suele realizarse en un lenguaje conceptual no ejecutable.

La siguiente fase consiste en compilar el modelo “rellenado”, que transforma el esqueleto del modelo en un modelo ejecutable. Esta fase sólo es necesaria si el modelo “rellenado” se describe en un lenguaje no ejecutable. La compilación o traducción del modelo puede ser manual, automática o semiautomática.

La principal problemática del proceso suele surgir cuando el lenguaje fuente es más expresivo que el lenguaje destino. La traducción totalmente automática suele producir modelos ejecutables poco eficientes, por lo que se suele realizar una traducción semiautomática, que mantiene un diálogo con el ingeniero del conocimiento u ofrece una colección de procedimientos de compilación.

La última fase consiste en el refinamiento del modelo ejecutable. El modelo ejecutable se valida con un conjunto de casos de prueba y si hay errores se busca qué fase de las anteriores debe modificarse. Esta fase puede estar soportada por una herramienta, con facilidades para inspeccionar las ejecuciones con preguntas del tipo por qué, cómo, por qué no o qué pasaría si, o con facilidades de comprobación de completitud y consistencia del conocimiento.

Actividades de construcción del esqueleto del modelo

La construcción del esqueleto de un modelo consta de las siguientes actividades:

1. Descripción informal del dominio y tareas de la aplicación. Esta fase denominada de orientación se realiza en CommonKADS en los modelos de organización y tareas, por lo que no se desarrolla en este modelo.

2. Identificación de tareas genéricas.

3. Construcción del esquema del modelo, que hemos desglosado en identificación y descripción del esquema del modelo.

4. Especificar las correspondencias entre papeles de tareas genéricas diferentes. Una vez adquirido el esquema del modelo puede determinarse si hay alguna correspondencia entre papeles del conocimiento de tareas genéricas diferentes.

5. Especificar las correspondencias entre papeles del esquema del modelo y de las tareas genéricas. Es necesario relacionar cada papel del conocimiento de las estructuras de inferencias de las tareas genéricas con un término del esquema del modelo. Estas reglas se denominan “correspondencias ontológicas”.

6. Validación del esqueleto. Se comprueba la completitud y consistencia del conocimiento.

Modelo de Coordinación

Se presenta el modelo de coordinación para desarrollar y describir las interacciones entre los agentes de un sistema Multi-Agente. Se muestra su relación con el resto de modelos de MAS-CommonKADS y los métodos de construcción del modelo.

Dependencias en la construcción del modelo

Modelo de Agente

La relación entre el modelo de coordinación y el modelo de agente es clara: el modelo de coordinación describe las conversaciones entre agentes. Es necesario, por tanto, haber realizado un modelo de agente inicial para poder describir las interacciones. Estos agentes iniciales pueden haberse identificado en la fase de conceptuación.

Debe darse una relación de coherencia entre ambos modelos: todos los agentes del modelo de coordinación deben estar descritos en el modelo de agente.

Normalmente el desarrollo del modelo de coordinación se realizará en paralelo con el modelo de agente, ya que todas las capacidades identificadas en el modelo de coordinación deben ser reflejadas en el modelo de agente.

Modelo de Tareas

El modelo de tareas describe los ingredientes de las tareas que se comunican los agentes entre sí. Normalmente asumiremos que el modelo de tareas se ha desarrollado previamente al modelo de coordinación, aunque el desarrollo del modelo de coordinación puede conllevar una revisión del modelo de tareas, tanto de su estructura como de los ingredientes intercambiados.

Modelo de Comunicación

El modelo de comunicación depende del modelo de coordinación. Durante el desarrollo del modelo de coordinación hemos supuesto que los usuarios humanos son sustituidos por agentes software que actúan como asistentes personales. Esto permite independizar el resto de modelos de los requisitos de la interfaz

hombre-máquina seleccionado. Se supone, por tanto, que el modelo de comunicación se desarrolla posteriormente al modelo de coordinación. En el caso de que se seleccione un protocolo de negociación puede darse que un agente no tenga conocimiento suficiente para realizar dicha negociación de forma automática, sino que necesite de la asistencia de un humano. En este caso, durante el desarrollo del modelo de coordinación deberá reflejarse este hecho al describir las capacidades de un agente que actúa como asistente personal.

Modelo de Organización

El modelo de coordinación puede ser utilizado para representar las interacciones habituales en una organización. En este caso, en vez de fijarnos en agentes concretos representaremos interacciones de papeles (roles) concretos de los agentes de la organización.

Modelo de la Experiencia

Las interacciones de las tareas de transferencia (tareas cuyo objetivo es la comunicación con otro agente) del modelo de la experiencia se describen en este modelo y en el modelo de comunicación. La descripción de las capacidades de un agente para poder mantener una conversación puede hacer relación a conceptos y relaciones descritas en un ejemplar del modelo de la experiencia.

El desarrollo del Modelo de Coordinación combina técnicas de metodologías orientadas a objetos con lenguajes de descripción formal de ingeniería de protocolos.

Modelo de Organización

El modelo de organización de MAS-CommonKADS tiene como objetivo analizar desde una perspectiva de grupo las relaciones entre los agentes (tanto software como humanos) que interactúan con el sistema.

El modelo de organización de MAS-CommonKADS tiene como antecesor al modelo homónimo de CommonKADS.

Estructura del modelo

Constituyentes del modelo.

En el modelo se analiza una organización con varias perspectivas: recursos, funciones, estructura, relaciones de autoridad y procesos en la organización. También se describen los problemas de la organización que pueden resolverse con la introducción de un sistema inteligente. El concepto de constituyente en el modelo de organización difiere de los constituyentes de otros modelos. En el modelo de organización los constituyentes principales son puntos de vista para analizar una organización, y no tienen, por tanto, atributos en sí mismos. Esta es la razón por la cual no tienen tampoco plantillas asociadas, ya que su definición consiste en identificar y describir (o dibujar) las funciones de una empresa, su estructura, los procesos realizados, las relaciones de poder o autoridad y los recursos disponibles. En el caso de que el objeto de análisis sea una sociedad Multi-Agente, estos mismos criterios son aplicables, junto con el estudio de las relaciones de herencia.

Dependencias en la construcción del modelo

Las dependencias del modelo de organización con el resto de modelos deben validarse durante el desarrollo del mismo.

Modelo de Agente

La relación entre el modelo de agente y de organización consiste en que el modelo de organización describe las relaciones estructurales entre los agentes humanos y software tanto de la organización humana como de la sociedad Multi-Agente. El modelo de organización requiere, por tanto, que se hayan identificado y descrito los agentes.

Modelo de Tareas

El modelo de tareas está relacionado con la perspectiva del modelo de organización que analiza las funciones desempeñadas en la organización. Las funciones se relacionan con las tareas de dos formas: una función define principalmente varias tareas requeridas para realizar dicha función y una tarea sirve para realizar una o varias funciones. Las funciones de la organización que vayan a ser realizadas por el sistema Multi-Agente serán el nodo raíz de un ejemplar del modelo de tareas. En caso de que se

desarrolle el modelo de organización, el estudio de la organización humana debe ser anterior al desarrollo de modelo de tareas. El estudio de la organización de agentes será posterior al desarrollo de un modelo de tareas inicial y a la asignación de objetivos (tareas) a los agentes.

Modelo de Comunicación

El modelo de comunicación puede obtener información para realizar el modelado de los usuarios del modelo de organización, que debería ser desarrollado con antelación.

Modelo de Coordinación

El modelo de coordinación depende de las estructuras organizativas definidas en el modelo de organización, en el caso de que se definan. Las estructuras organizativas pueden limitar o definir los protocolos que deben mantenerse entre los componentes de dichas estructuras.

En esta sección se ha presentado el modelo de organización de la metodología MAS-CommonKADS. Los fines de dicho modelo son:

- Comprender la organización en que se va a introducir un sistema Multi-Agente
- Analizar la viabilidad del sistema Multi-Agente.
- Describir las relaciones de la sociedad Multi-Agente.

Modelo de Diseño

En esta sección se presenta el modelo de diseño, cuyo objetivo es la descripción de los componentes que cumplen los requisitos del resto de modelos del análisis y que tienen en cuenta los requisitos no funcionales. Se muestra su fundamento lógico, su estructura, relación con el resto de modelos de MAS-CommonKADS y los métodos de construcción del modelo.

El modelo de diseño de MAS-CommonKADS parte del modelo homónimo de CommonKADS. El modelo original distinguía entre diseño de la aplicación, diseño de la arquitectura y diseño de la plataforma. Esta

estructura se mantiene pero ha sido dotada de un nuevo significado: distinguiremos entre diseño del agente (aplicación), diseño de la red (arquitectura) y diseño de la plataforma.

Dependencias en la construcción del modelo

El modelo de diseño debe estar vinculado al resto de modelos, ya que especifica cómo las funcionalidades definidas en el resto de modelos son implementadas. En CommonKADS los modelos de especificación presentan una especificación detallada del comportamiento del sistema, no sólo de los requisitos de dicho sistema. Por tanto, aportan más información que la contenida en un documento de requisitos software.

Los modelos de análisis de CommonKADS deben expresar todos los requisitos funcionales (que describen la conducta funcional del sistema) y expresan, además, algunos de los requisitos no funcionales (requisitos de restricciones para que una implementación sea aceptable).

Modelo de agente

El modelo de agente recoge las restricciones y capacidades de los agentes para llevar a cabo los objetivos asignados. En el diseño pueden combinarse o subdividirse los agentes del análisis, bien para satisfacer requisitos no funcionales, bien debido a las características de la arquitectura de agente seleccionada (por ejemplo si permite la realización de tareas simultáneas o no).

Modelo de comunicación

El modelo de comunicación es la entrada del diseño de la interfaz de usuario. En dicho modelo se ha recogido la representación de la información intercambiada y la secuencia de interacciones entre el usuario y el sistema.

Modelo de tareas

El modelo de tareas puede ayudar en el diseño, dado que ofrece la descomposición global de tareas del sistema, algunas de las cuales son descompuestas en otros modelos (experiencia, comunicación, coordinación) y asignadas a agentes. Los atributos “funcionales” de la plantilla textual del modelo de

tareas son objetivo, precondición y control-tareas. Los atributos “no-funcionales” de dichas plantillas son frecuencia y tiempo-ejecución. Estos parámetros pueden influir en la implementación de dichas funciones o en el diseño general de la aplicación para satisfacer estas restricciones.

Modelo de la experiencia

La comunidad de adquisición del conocimiento ha estudiado en profundidad cómo se pueden “operacionalizar” las descripciones conceptuales de los modelos de la experiencia.

Modelo de organización

El estudio de la organización restringe las decisiones de diseño. En concreto, en el modelo de organización se enumeran los recursos hardware y software disponibles. También el estudio de la organización puede proporcionar información sobre los requisitos no funcionales, tales como estándares seguidos para la instalación, operación y mantenimiento de aplicaciones, políticas de seguridad, etc.

Modelo de coordinación

El modelo de coordinación describe las interacciones y protocolos entre los agentes y el procesado de dichas interacciones. El diseño de los protocolos entre agentes deberá seguir las especificaciones desarrolladas en el modelo de coordinación.

Conclusiones

En el transcurso de este capítulo, como base para el entendimiento del tema en que se desenvolverá este proyecto, se formularon una serie de conceptos básicos para el posterior entendimiento de lo expuesto.

Se trataron también las principales características de las metodologías a usar que serán utilizados para el desarrollo de este trabajo.

Capítulo 2 Las metodologías MaSE Y MAS-COMMONKADS: un caso de estudio

Introducción

El presente Capítulo propone la aplicación de MaSE y MAS-CommonKADS a un caso de estudio. En este se presentan los artefactos correspondientes a cada fase de nivel de aplicación. Es importante resaltar que el objeto de este trabajo no abarca la implementación real del caso de estudio propuesto; por lo cual el mayor énfasis se realiza sobre el análisis y diseño de ambas metodologías.

2.1 Descripción del Caso de Estudio

El proyecto @LIS Technology Net, se construye usando como base la plataforma experimental AgentCities para utilizar las más modernas tecnologías de agentes autónomos, Web Internet.

Entre los resultados más importantes se espera disponer de un demostrador capaz de crear servicios personalizados de carácter cultural o turístico que sean accesibles en línea desde un PC y/o wirreles (teléfonos móviles, PDAs con servicio de roaming u otros dispositivos) a través de la composición dinámica de servicios instalados por los distintos socios en los diferentes países de los socios del consorcio.

El sistema debe ser capaz de que un usuario se conecte desde una PDA o desde una PC, a una página de Internet, para registrarse y solicitar un plan de viaje, simulando una agencia turoperadora, desde un país cualquiera a los países Cuba, Costa Rica, México y Chile. Además el mismo debe tener a su cargo la automatización de una serie de acciones que le permitirán al usuario obtener lo que desea.

2.2 Aplicación de la Metodología MaSE al caso de estudio propuesto

A continuación se explicaran las dos fases correspondientes a la metodología MaSE según el caso de estudio propuesto.

2.2.1 Fase de Análisis

Captura de Objetivos

La metodología MaSE no proporciona una descripción específica de cómo deben ser capturados los requisitos iniciales del sistema. No obstante a su incapacidad de aludir a lo antes mencionado, se propone realizar en el siguiente paso una captura de requisitos que debe cumplir el sistema para su óptimo desarrollo.

El sistema debe permitir fundamentalmente la gestión del usuario que desea conectarse al turoperador, así como la gestión de todos los itinerarios que dicho turoperador debe mostrar al usuario.

1. Gestión de usuarios:
    - Cada usuario debe darse de alta en el sistema antes que pueda utilizarlo, cada uno se registrará con su nombre y el correspondiente password.
    - Se crea un usuario que tendrá acceso a los diferentes destinos y lugares que desee visitar.
    - Un usuario podrá darse de baja.
    - Un usuario ya registrado debe autenticarse en el sistema para acceder a éste.
    - Se valida la información existente de cada usuario.
    - Se actualiza el perfil de los usuarios.
    - El usuario obtiene el itinerario elegido.
2. Gestor de Itinerario Global
    - Se planifica el itinerario según la disponibilidad que exista en ese momento.
    - Se envía solicitud de itinerario al itinerario local
    - Se planifican los vuelos acorde con el itinerario local
    - Se divide tiempo y dinero con que cuenta el usuario para su reservación

- Se encapsula toda la información y se envía una propuesta de itinerario

3. Planificar Itinerario Local
    - Se gestiona todo lo relacionado con el alojamiento de usuario
    - Se localiza el grupo de actividades que estén disponibles para el usuario
    - Se gestiona lo relacionado con el traslado del usuario.
    - Se envía la información del itinerario local planificado al Gestor de Itinerarios Global
4. Gestión de Información
5. Se recibe la información que el usuario ha solicitado
6. Se valida la información solicitada
7. Se muestra en pantalla el posible itinerario que el usuario debe recorrer.

Figura 4: Diagrama Jerárquico de Objetivos

Relacionados con cada uno de los objetivos, y necesarios para realizar las operaciones especificadas en los requisitos del sistema, aparecen los siguientes objetivos:

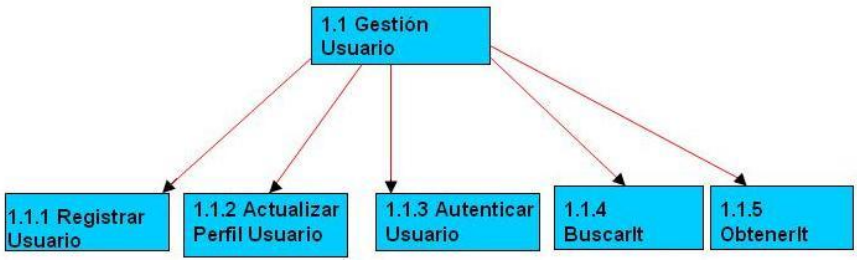

Figura 5: Diagrama de Objetivos: Gestión Usuario

Figura 6: Diagrama de Objetivos: Gestionar Itinerario Global

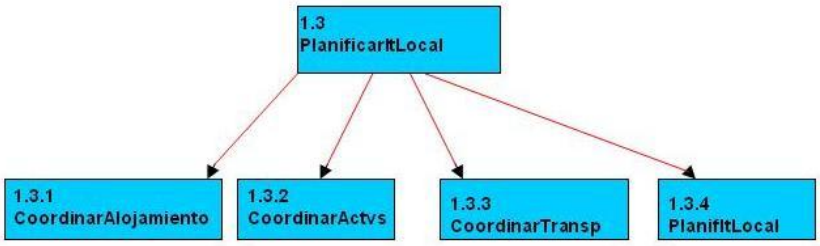

Figura 7: Diagrama de Objetivos: Planificar Itinerario Local

Figura 8: Diagrama de Objetivos: Gestionar Información

Diagrama de Casos de Uso

Este proyecto no tiene en cuenta ningún tipo de jerarquía de usuario por lo que cada individuo que interactúe con el sistema es un actor del mismo.

En lo que se refiere a recursos externos, dígase que este actor es un actor genérico que representa las diversas fuentes de información en las que se sustenta el sistema. Entiéndase por ello las bases de datos (BD en lo adelante), Planificador Personal, que registra los perfiles de los usuarios existentes; la BD Histórico de Itinerarios que almacena los itinerarios que llegan a ser aprobados por cada usuario, y a partir de la cual se procede a la actualización de los diversos perfiles.

También se encuentran las BD de las diversas Aerolíneas, donde se concentra toda la información referente a los vuelos (origen-destino, fecha, hora, costo, etc.). Se tiene además la BD Actividades, la BD Alojamiento, y la BD Transporte, que se encuentran ubicadas en cada uno de los países que se incluyen y almacenan información similar a la descrita en las BD de las Aerolíneas.

A continuación se ofrece los Casos de Uso identificados:

1. CU Recibir Información
2. CU Mostrar Información
3. CU Validar
4. CU Autenticar
5. CU Crear Usuario
6. CU Actualizar Perfil
7. CU Obtener Itinerario
8. CU Planificar Itinerario
9. CU Gestionar Vuelo
10. CU Planificar Itinerario Local
11. CU Gestionar Actividades
12. CU Gestionar Alojamiento
13. CU Gestionar Transporte

Descripción de los casos de uso y diagramas de secuencia

De los casos de uso identificados, por el interés que representan para el desarrollo del sistema, se detallarán los que a continuación se presentan:

MaSE propone una descripción menos detallada de los diagramas de secuencia que las metodologías orientadas a objetos tales como RUP.

Nombre CU: Obtener Itinerario

Objetivo: Conformar una plantilla de itinerario tomando como base la selección realizada por el Usuario.

Actores: Usuario.

1. **Descripción**:
2. El usuario se autentica para realizar la selección de las preferencias que desea incluir en su itinerario.
3. El gestor de información valida la autenticación de usuario. [Excepción: entrada de datos incorrectos].
4. El gestor de información le muestra al usuario la página principal del sistema donde se encuentra toda la información concerniente al mismo.
5. El usuario teclea la solicitud de itinerario.
6. El gestor de información envía los datos al gestor de usuarios para que obtenga el itinerario deseado.
7. El gestor de usuarios envía la información solicitada al Gestor de Itinerario Global para que se seleccione el itinerario que será brindado al usuario de la lista de posibles itinerarios que posteriormente se confeccionará.
8. El Gestor de Itinerario Global tiene que solicitar al Planificador de Itinerario Local una propuesta de itinerario y enviar la misma al Gestor de Usuario.
9. El Planificador de Itinerario gestiona todo lo relacionado con el alojamiento, las actividades y el transporte.
10. El planificador de Itinerario envía reservación al Gestor de Itinerario Global

11. El Gestor de Itinerario Global solicita al Gestor de Vuelos la información concerniente a los vuelos para los datos solicitados previamente.
12. El gestor de vuelos hace un posible itinerario de vuelos y envía la información al gestor de itinerarios
13. El gestor de Itinerario Global obtiene toda la información que le ha sido enviada, la encapsula y se la envía al Gestor de Usuario.
14. El gestor de Usuario confirma el itinerario elegido al Gestor de Itinerario Global
15. El Gestor de Itinerario Global notifica OK al Gestor de Información
16. El Gestor de Información muestra toda la información solicitada al Usuario.
17. Si el usuario no está satisfecho con la propuesta de itinerario brindada, se añade la funcionalidad de deducir cual de los itinerarios restantes es el que más se semeja a la solicitud inicial. [Excepción: no hay más variantes]
18. Presenta como CU incluido el CU Planificar Itinerario.

Excepciones:

1. Entrada de datos Incorrectos: Si el usuario se logea incorrectamente mostrar una ventana con la advertencia de que ha cometido un error a la hora de entrar los datos correspondientes.
2. No hay más variantes: Si se agotan todas las posibles variantes, se procede a la confección de una nueva plantilla.

Escenarios:

1. Obtener Itinerario.

Diagramas de Secuencia:

Obtener Itinerario: Muestra el curso normal que sigue la confección de un itinerario en el sistema, desde que el usuario que lo solicita se autentica, hasta que confirma el itinerario propuesto

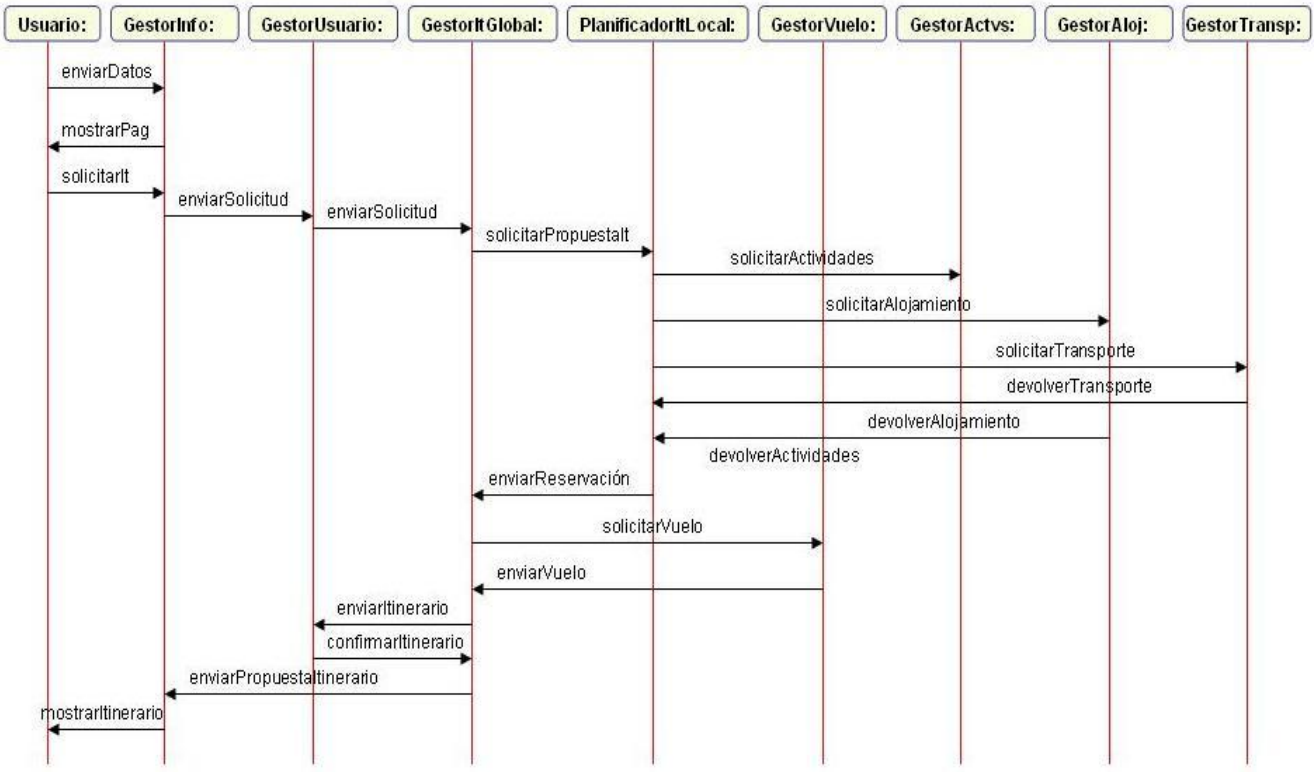

Figura 9: Diagrama de CU Obtener Itinerario

Nombre CU: Registrar Nuevo Usuario

Objetivo: Permitir que el usuario forme parte de la lista de usuarios logeados en el sistema.

Actores: Recursos Externos

Descripción:

1. El Usuario se conecta a la página para acceder a la sección de logeo de la misma y entra los datos con los que desea registrarse (identificación de usuario y password)
2. El Gestor de Información procede a validar los datos entrados por el Usuario. [Excepción: Datos incorrectos]

3. El Gestor de Información solicita el registro de usuario al Gestor de Usuario.
4. El Gestor de Usuario comprueba que el identificador de usuario no esté en la base de datos [Excepción: error identificación].
5. El Gestor de Usuario notifica el registro al Gestor de Información
6. El Gestor de Información le notifica al Usuario que ya ha sido incluido en la lista.

Excepciones:

- Datos incorrectos: Esto significa que el usuario no debe entrar caracteres extraños a la hora de crearse un identificador, tampoco debe hacer uso indiscriminado de las mayúsculas, ni que el id o password sean la repetición de un único carácter. Además el password debe cumplir la restricción de tener como mínimo 7 caracteres.
- Error Identificación: Si el Identificador coincide con uno que está en la base de datos, se le notifica al usuario que debe elegir otra identificación.

Escenarios:

1. Registrar Nuevo Usuario.

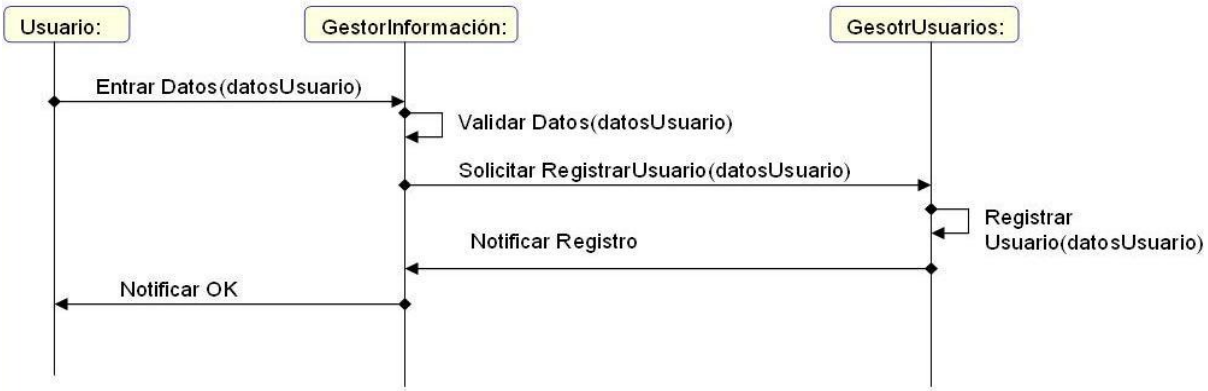

Figura 10: Diagrama de Secuencia Registrar Nuevo Usuario

Refinamiento de Roles

A continuación se proponen los roles que se deben llevar cabo dentro del sistema:

1. Rol de Interfaz
2. Rol Registrar
3. Rol Autenticar
4. Rol Buscar Itinerario
5. Rol Confirmar Itinerario
6. Rol Coordinar Itinerario
7. Rol Gestionar Itinerario
8. Rol Coordinar Vuelo
9. Rol Coordinar Actividades
10. Rol Coordinar Alojamiento
11. Rol Coordinar Transporte

Modelo de Roles

En este modelo se muestran un grupo de roles llevados a cabo por el sistema, a los cuales se les asigna un determinado número de tareas para llevar a cabo.

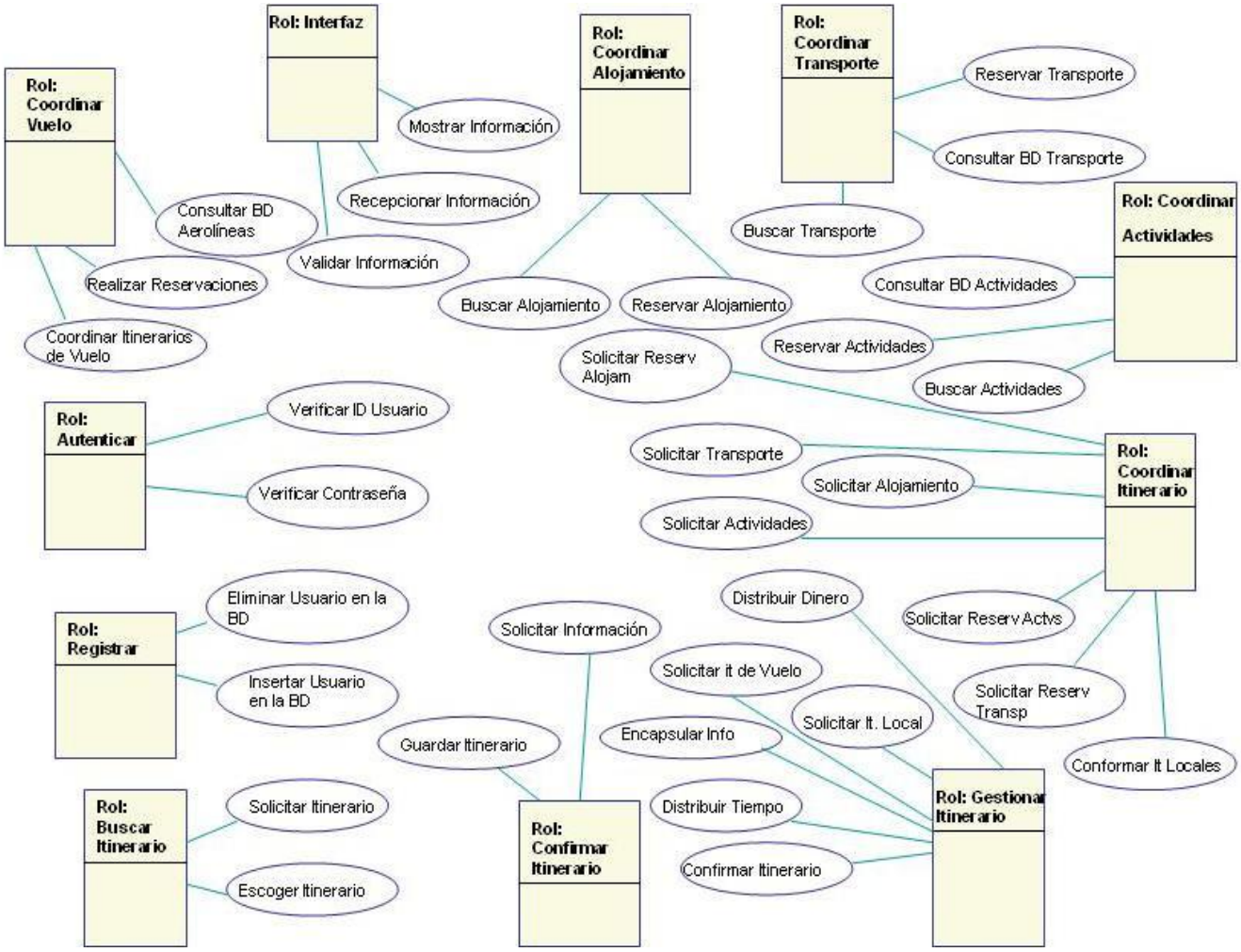

Figura 11: Modelo de Roles de la Metodología MaSE

MaSE realiza una descripción de comunicación de roles en el mismo modelo que propone, pero dada la complejidad estructural del que se presentó anteriormente, resulta conveniente realizar dicha descripción de roles mediante texto plano, lo cual contribuye a que el modelo de roles no se presente cargado en imágenes que puedan provocar que el mismo resulte ininteligible.

A continuación se muestran las comunicaciones entre roles:

El rol interfaz interactúa con el rol buscar itinerario ya que el primero es el encargado de solicitarle el itinerario que el usuario ha elegido, ambos intercambian información de envío y respuesta. Este rol además se comunica con los respectivos roles autenticar, confirmar itinerario y gestionar itinerario.

Por su parte, el rol buscar itinerario se comunica con el rol autenticar y gestionar itinerario ya que una vez que el usuario se autentique en el sistema tendrá acceso a solicitar un itinerario y escoger entre los disponibles el de mayor preferencia.

El rol autenticar se comunica con el rol buscar itinerario y con el rol confirmar itinerario, mientras que este último se comunica con el rol interfaz y rol gestionar itinerario

El rol gestionar itinerario se comunica con el rol interfaz, autenticar, confirmar itinerario, coordinar itinerario local y coordinar vuelo.

El rol coordinar itinerario local se comunica con el rol gestionar itinerario, coordinar alojamiento, coordinar actividades y coordinar transporte.

Diagrama de Tareas Concurrentes

Rol: Registrar.

Tarea: Registrar Usuarios

En esta tarea se registran y se eliminan los usuarios dependiendo del tipo de petición que envíe el gestor de usuarios. Si la petición es la de registrar un usuario, primero comprueba que no existe, y si no existe, lo añade a la lista de usuarios y se almacena en un archivo.

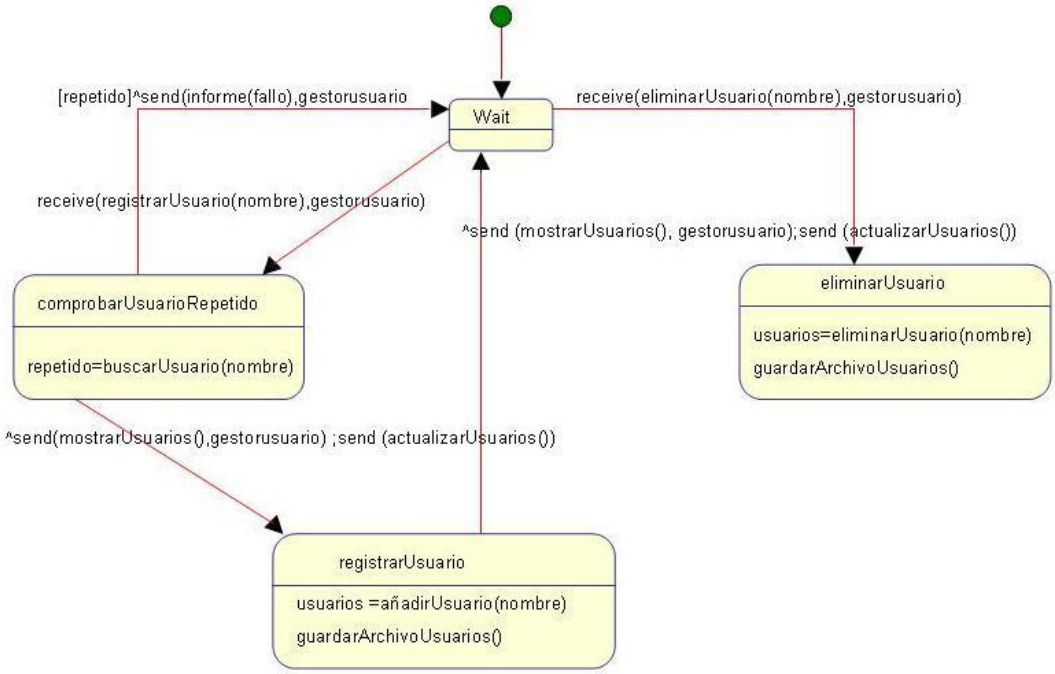

Figura 12: Diagrama de Tareas Registrar

2.2.2 Fase de Diseño

Crear las Clases de Agentes

Los roles del modelo de rol se van a agrupar en ocho agentes:

- Agente Interfaz: El criterio seguido para la creación de este agente fue atender principalmente a los casos de usos a través de los cuales el usuario interactúa con el sistema. Por ello se propone un agente que modere esta interacción, y que responda por las validaciones a nivel de interfaz de toda la información que circule entre el Sistema y el Usuario.
- Agente Personal: Se propone este agente como moderador de la comunicación entre el sistema y los recursos externos en que se encuentra almacenada toda la información. El mismo interactúa con la Base de Datos Planificador Personal y la Base de Datos Histórico de Itinerarios.

- Agente Global: Es responsable de dividir el tiempo y el dinero disponible por el usuario. Divide en diversas secciones las actividades y lugares de interés, atendiendo a reglas que organizan dichas secciones, asignándole a cada país las actividades correspondientes. Tiene la responsabilidad de obtener los posibles vuelos teniendo en cuenta los pares origen-destino. Una vez que se obtengan las secciones de itinerarios satisfechas por cada país, el Agente Global encapsula estas propuestas junto a las opciones de vuelos hacia el país en cuestión; conformando así una lista de posibles itinerarios, la cual será suministrada al Agente Personal descrito con anterioridad. Este agente se encarga de que, en caso de ser aceptado algún itinerario por el Usuario, iniciar el proceso de reservaciones, tanto en los diversos países involucrados, como en las Aerolíneas incluidas.
- Agente Local: Es el encargado de confeccionar los diversos itinerarios que le sean solicitados. El itinerario local se confecciona en términos de actividades, alojamiento y servicio, las cuales se asignan atendiendo al tiempo, costo, calificación y clasificación de las mismas indistintamente. Una vez recibida la solicitud por parte del Agente Global, se procede a la debida reservación de los diversos elementos del itinerario aprobado por el usuario, y la notificación de la culminación de este proceso.
- Agente Vuelo: Constituye una interfaz de comunicación entre el Sistema y las Bases de Datos de las diferentes Aerolíneas. A partir de una solicitud del Agente Global, el Agente Vuelo devuelve un número finito de posibles conexiones entre el país de origen y el destino, así como la categoría de cada viaje, y el costo del mismo. Una vez recibida la solicitud, este agente responde por la reservación de vuelos solicitados.
- Agente Actividades: Constituye una interfaz de comunicación entre el Sistema y las Bases de Datos de las diversas entidades que brinden actividades y servicios en el país. A partir de una solicitud, el Agente Actividades devuelve un número de posibles actividades a realizar en el país, así como la categoría de cada actividad o servicio, y el costo del mismo. Este agente responde por la debida reservación de las actividades y/o los servicios solicitados.
- Agente Alojamiento: Constituye una interfaz de comunicación entre el Sistema y las Bases de Datos de las diversas entidades que brinden alojamiento en el país. El Agente Alojamiento devuelve un número finito de posibles hospedajes en un país, así como la categoría y el costo de los mismos. Responde por la debida reservación de los hospedajes solicitados.

- Agente Transporte: Constituye una interfaz de comunicación entre el Sistema y las Bases de Datos de las diversas entidades que brinden transporte en el país. Devuelve un número finito de posibles conexiones de origen-destino, así como la categoría y el costo de los mismos. Responde por la debida reservación de los medios de transporte solicitados.

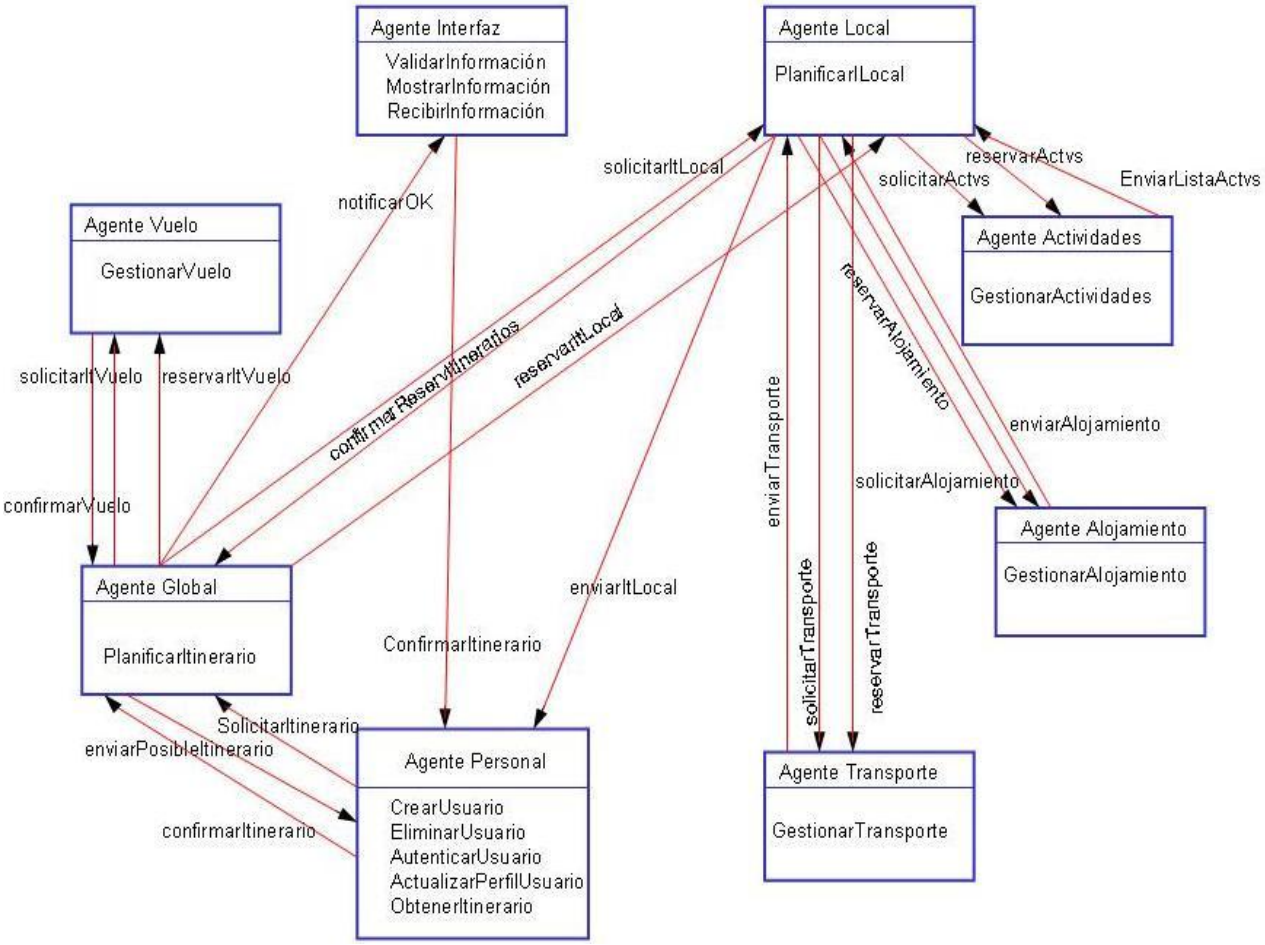

Figura 13: Diagrama de Clases de Agentes

En este trabajo se identificaron dos escenarios fundamentales: crear usuario y obtener itinerario. En el diagrama anterior se muestran las comunicaciones entre los agentes atendiendo al escenario obtener itinerario.

Construir Conversaciones

Para la construcción de conversaciones se propone describir una de las conversaciones que se lleva a cabo entre el Agente Interfaz y el Agente Personal, que en este caso es actualizar perfil de usuario.

ActualizarPerfilUsuario: El agente Interfaz cada vez que modifique algún dato sobre los usuarios, deberá informar al agente Personal para que actualice esta información y los usuarios no accedan a datos incorrectos.

El agente Interfaz inicia la conversación solicitando la petición de que se actualicen los datos de los usuarios utilizando la acción actualizarPerfilUsuario, y espera hasta que el agente Personal responda si se actualizó la información o no.

El agente Personal está esperando a que le llegue el mensaje de actualizar a los usuarios y cuando lo recibe, realiza la operación de modificación, y envía el informe del resultado de la modificación al agente Interfaz para que este la visualice.

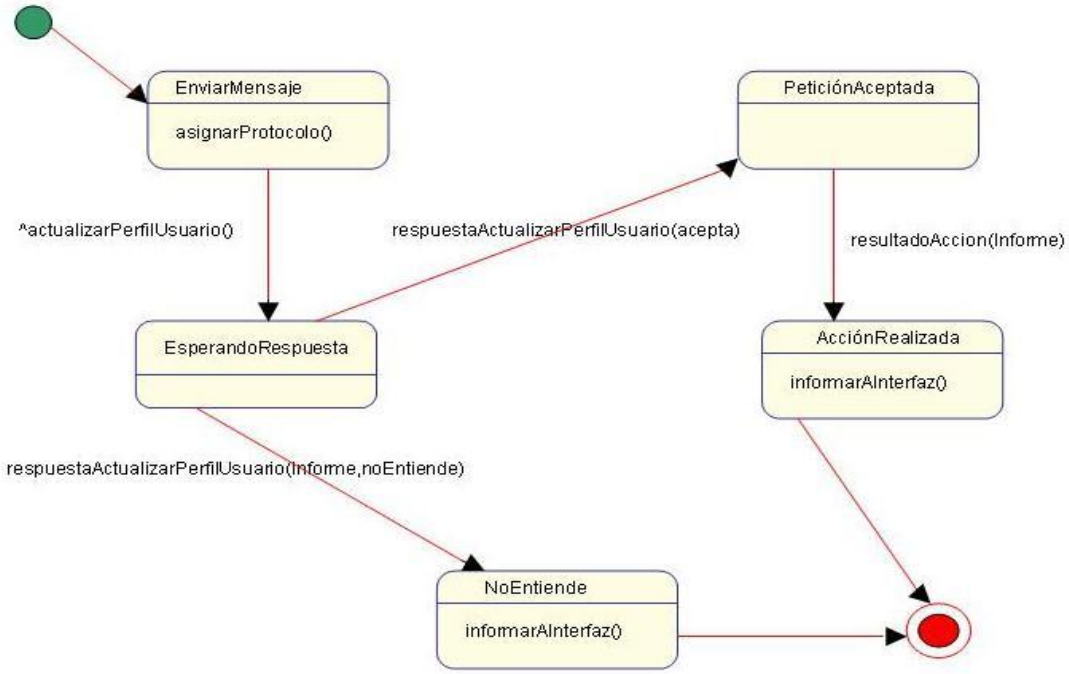

Figura 14: Actualizar Perfil Usuario: Emisor

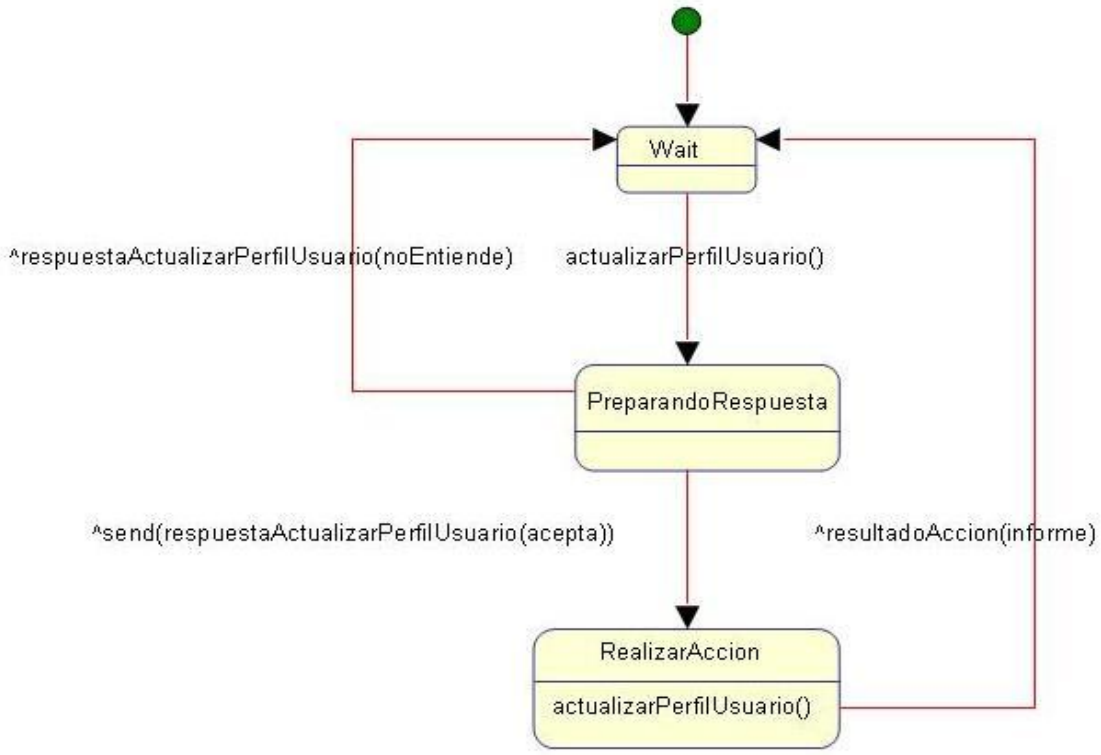

Figura 15: Actualizar Perfil Usuario: Receptor

Se propone aplicar el siguiente artefacto al Agente Personal, por ser el mismo uno de los agentes más importantes que realiza funciones complejas dentro del sistema.

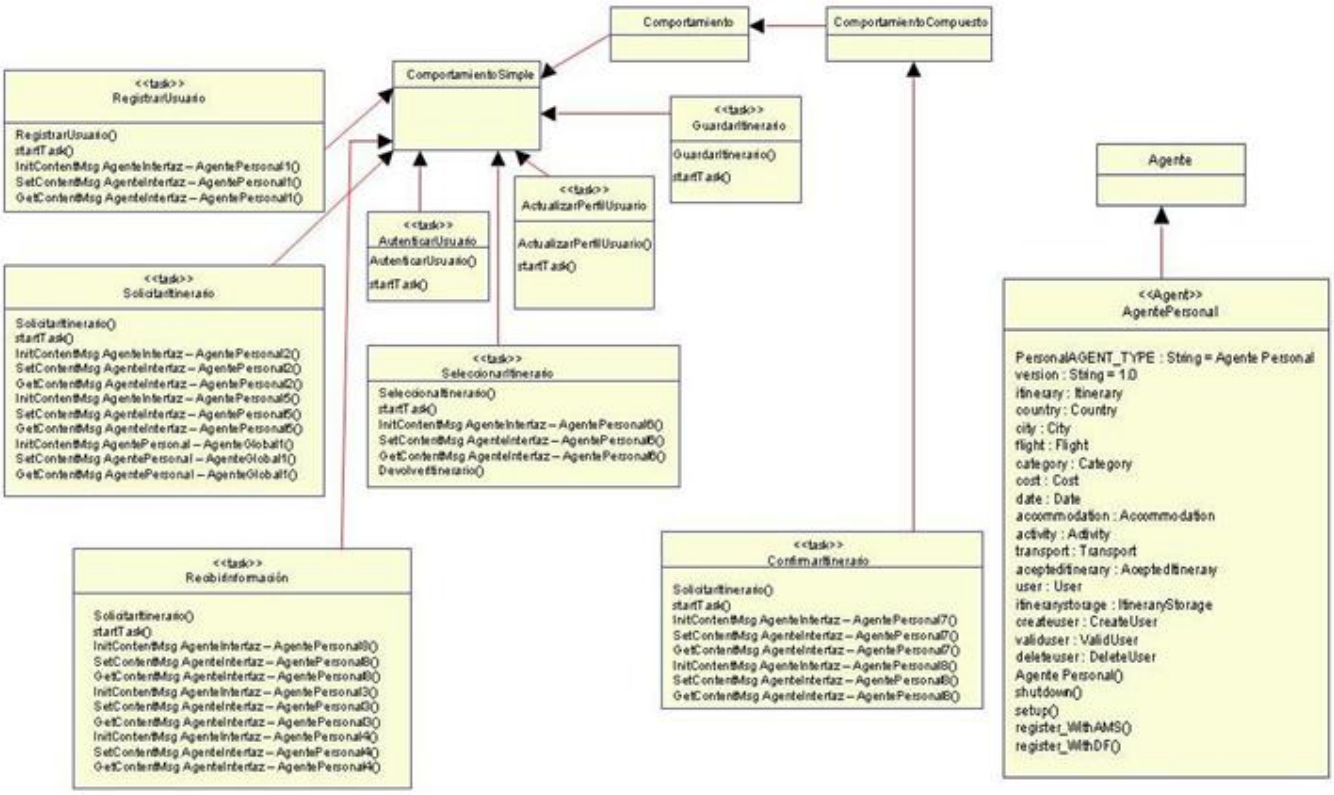

Figura 16: Diagrama de Clases de Agentes

Desarrollar el diseño

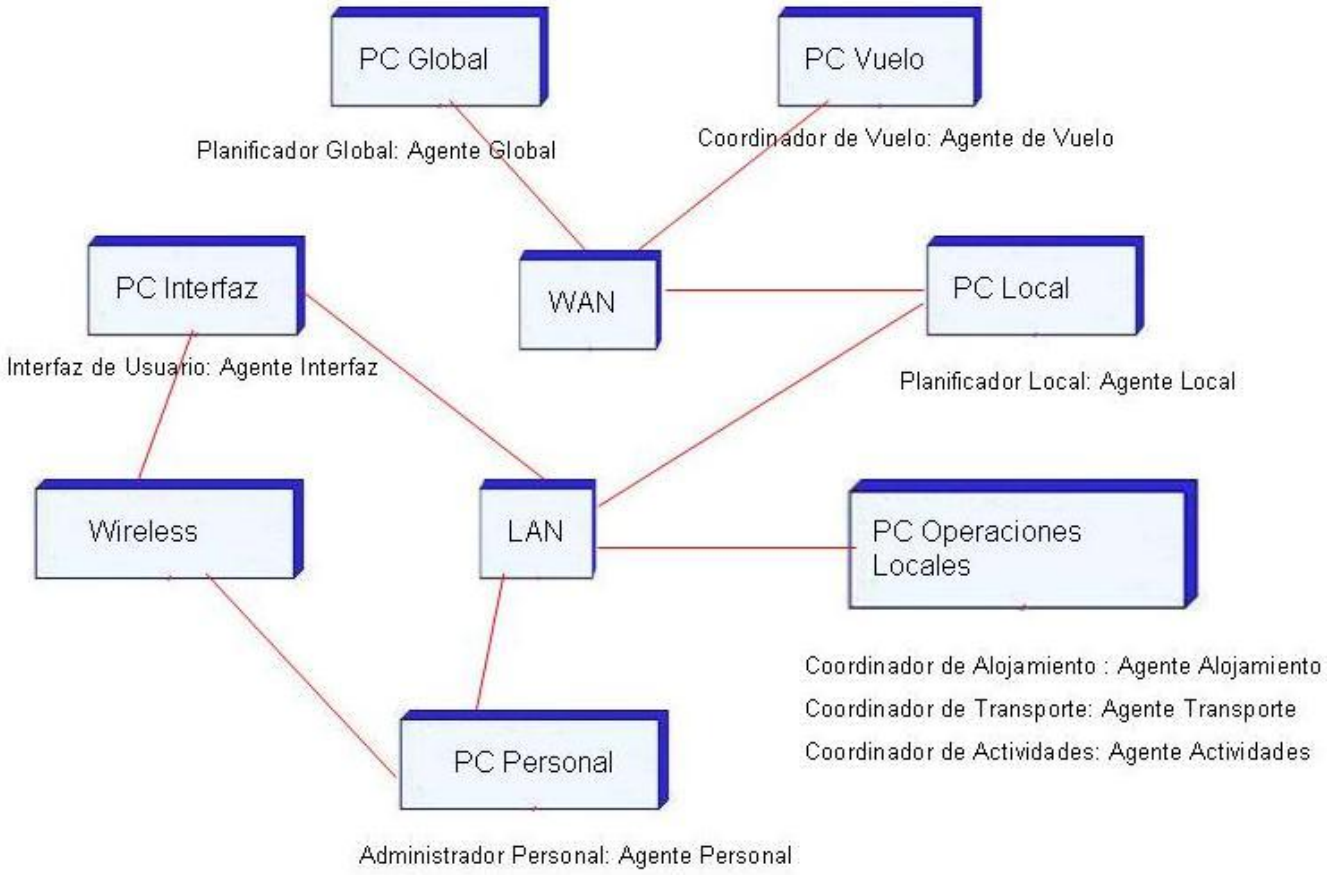

Figura 17: Diagrama de Despliegue

2.3 Aplicación de MAS-CommonKADS al Caso de Estudio propuesto

El uso de la metodología consiste en el desarrollo de los diversos modelos. Cada modelo consiste en los componentes (las entidades que se modelarán) y las relaciones entre los componentes. Una plantilla textual se define para cada constitutivo para describirlo.

2.3.1 Conceptualización

En esta fase se obtendrá una descripción preliminar del problema .Se determinan algunos casos de uso (escenarios) los cuales pueden ayudar a la comprensión de los requerimientos informales y las pruebas del sistema.

Se puede identificar como principal actor al usuario: persona que solicita un plan de viaje, es decir, un itinerario, los operarios (agentes software) y los sistemas de bases de datos.

. La información del usuario debe estar provista de: id _ usuario, día (d), presupuesto (p) y preferencias (prf).

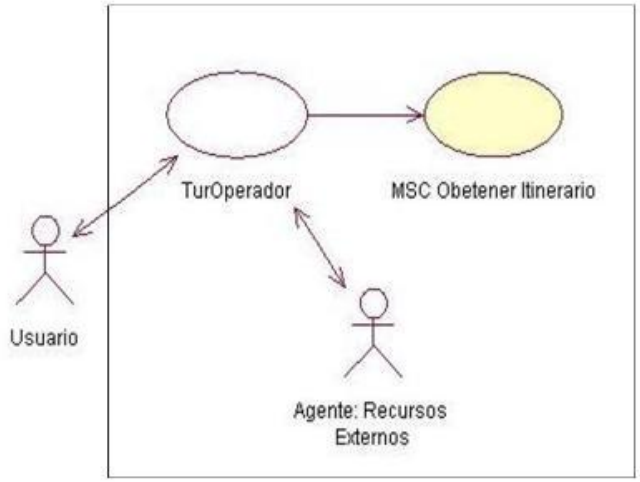

Figura 18: Diagrama de Caso de Uso

Las interacciones de los casos del uso se formalizan usando el MSC (Diagrama de Secuencia de Mensajes) como notación. En la siguiente figura se muestra como se realiza la elaboración del itinerario en el sistema, desde que el usuario que lo solicita se autentica, hasta que confirma el itinerario propuesto.. El propósito en esta fase es conseguir una idea de las interacciones, pero serán refinados más adelante en el modelo de la coordinación.

Ejemplo aplicado al caso de estudio:

Los Casos de Uso identificados, así como los operarios (agentes software) del sistema y los sistemas de bases de datos se encuentran reflejados al inicio de este Capítulo en la sección dedicada a MaSE.

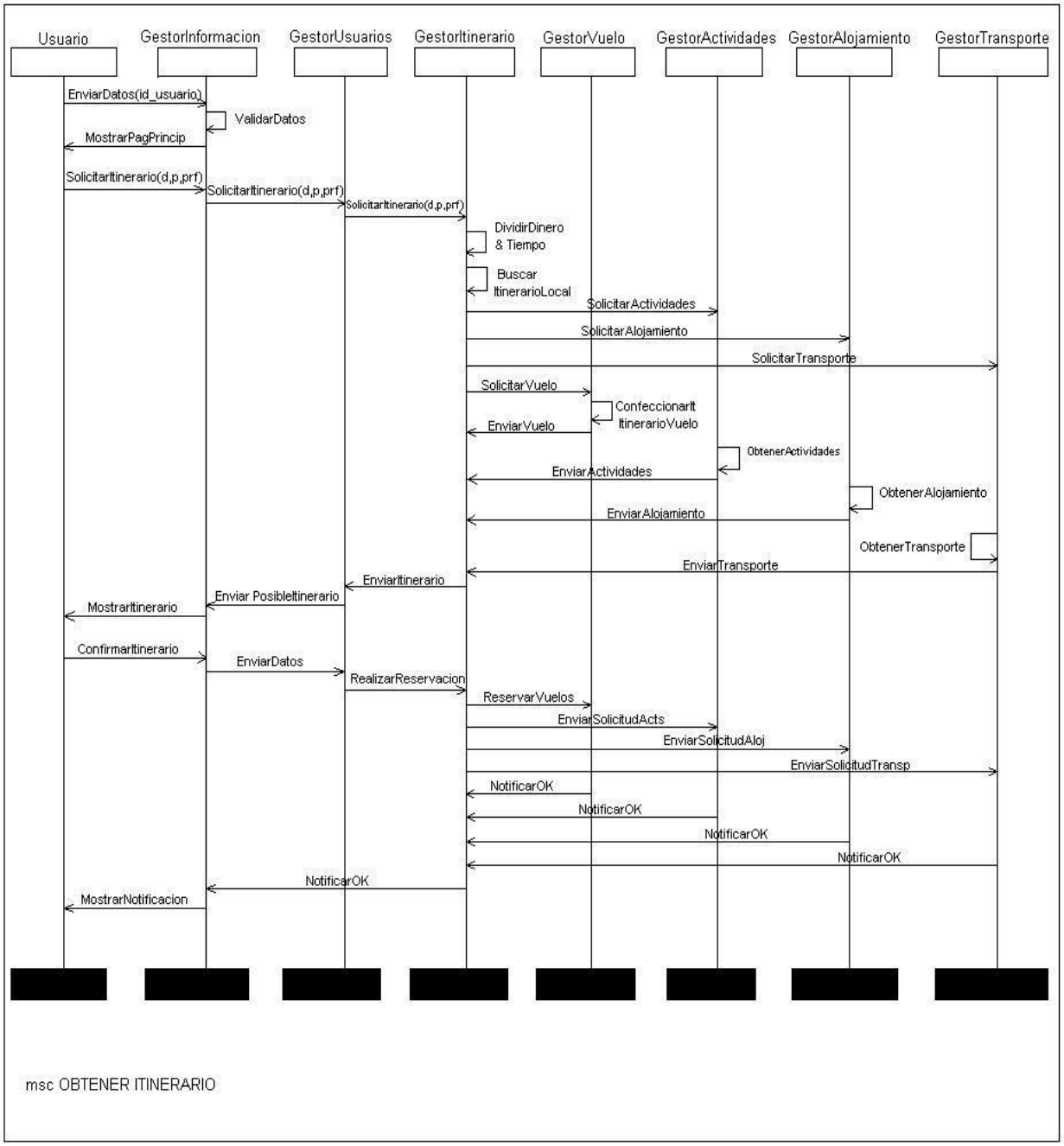

Figura 19: Diagrama del caso de uso MSC Obtener Itinerario.

2.3.2 Análisis

Los resultados de esta fase serán la especificación de requerimientos de la metodología a través del desarrollo de sus modelos previamente descritos en el Capítulo anterior.

Modelo de Agente

Identificación de Agentes:

En esta tesis se han identificado ocho agentes los cuales son de vital importancia para el funcionamiento de este sistema.

A continuación se nombran dichos agentes. La correspondiente explicación de las tareas que involucra cada uno de ellos, constituye la misma que anteriormente se trató en MaSE.

- Agente de Interfaz
- Agente Personal
- Agente Global
- Agente Local
- Agente Vuelo
- Agente Actividades
- Agente Transporte
- Agente Alojamiento

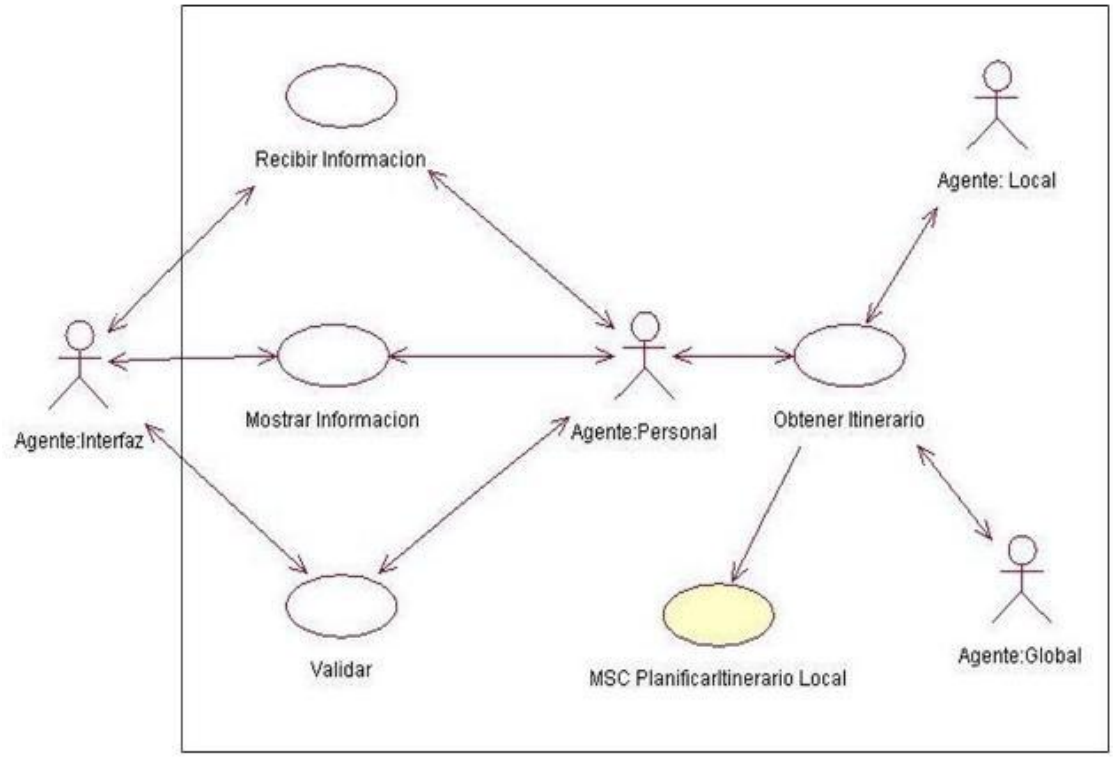

Figura 20: Diagrama Interno del Caso de Uso Planificar Itinerario Local

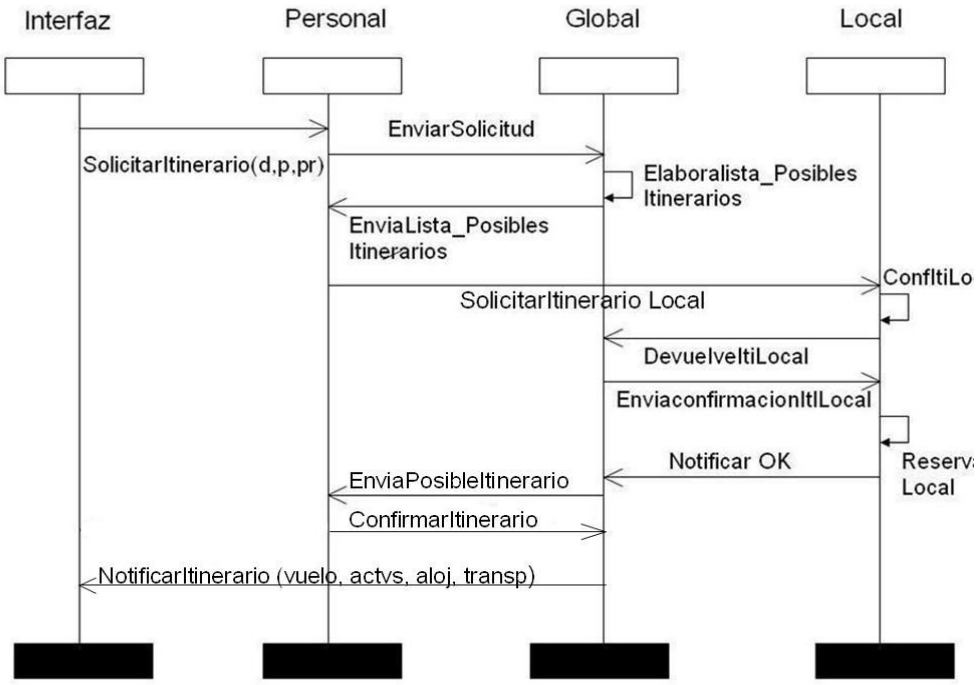

Figura 21: Diagrama del caso de uso MSC Obtener Itinerario Local

Es importante destacar que dicho modelo no asume ninguna arquitectura de agente concreto puesto que esta decisión será tomada posteriormente en el Modelo de Diseño.

A continuación se muestra una plantilla de agente aplicado a este caso de estudio (para el agente personal):

Agente: Personal

Tipo: Agente software inteligente.

Experiencia: Conocimiento del domino de itinerarios y gestión de los datos del usuario.

Modelo de Tareas

En este modelo las tareas son descompuestas jerárquicamente y/o descritas en un árbol.

La descripción de una tarea incluye su nombre, una corta descripción de la misma, elementos de entrada y salida, estructura de la tarea, su control, frecuencia de aplicación, precondiciones y capacidad requerida de los actores.

El beneficio potencial al desarrollar este modelo es la documentación de las actividades y la organización antes y después de la introducción del sistema Multi-Agente. Esta documentación ayuda a soportar el mantenimiento y administración de los cambios en la organización y como apoyo en la valoración de viabilidad del proyecto.

Se propone aplicar este modelo:

Tarea: Planificar Itinerario

Objetivo: Planifica los posibles Itinerarios para una solicitud.

Descripción: En esta tarea se determinan los itinerarios posibles a realizar. Se ofrecen todas las alternativas clasificadas según un criterio determinado por el usuario.

Entrada: Solicitud Itinerario (día, presupuesto, preferencias)

Salida: Posibles itinerarios

Una vez determinadas las tareas del sistema, deben asignárseles a los agentes.

Modelo de Coordinación

El Modelo de Coordinación tiene 2 hitos:

- Definición de los canales de comunicación y construcción de un estándar.
- Análisis de las interacciones y determinación de las interacciones complejas (con los protocolos de coordinación).

A continuación se exponen una serie de pasos relacionados con esta primera fase:

1. Se describen los escenarios estándares entre agentes utilizando la notación M SC.

Las conversaciones son identificadas tomando como entradas el resultado de las técnicas utilizadas para identificar agentes. Durante esta primera etapa se considerará que cada conversación consiste en una simple interacción y una posible respuesta.

2. Se representan los eventos (se intercambian mensajes) entre agentes en el diagrama de flujo de los eventos. Este diagrama almacena las relaciones entre agentes mediante servicios.

3. Modela el intercambio de información en cada interacción. El Modelo de Experiencia puede ayudar a definir el intercambio de las estructuras de conocimiento. Estos intercambios de información son mostrados en el diagrama de flujo de los eventos en cuadrados entre paréntesis.

4. Modela cada interacción mediante el diagrama de transición. De SLD (Specification and Description Language) especificando los actos de habla como mensajes eventuales de entrada/salida. Estos diagramas pueden ser confirmados por los diagramas MSC.

5. Cada estado puede ser analizado en más detalle en el modelo de Tarea o en el Modelo de Experiencia.

6. Se analiza cada interacción y se determina su tipo de sincronización: Sincrónico, asincrónico o futuro.

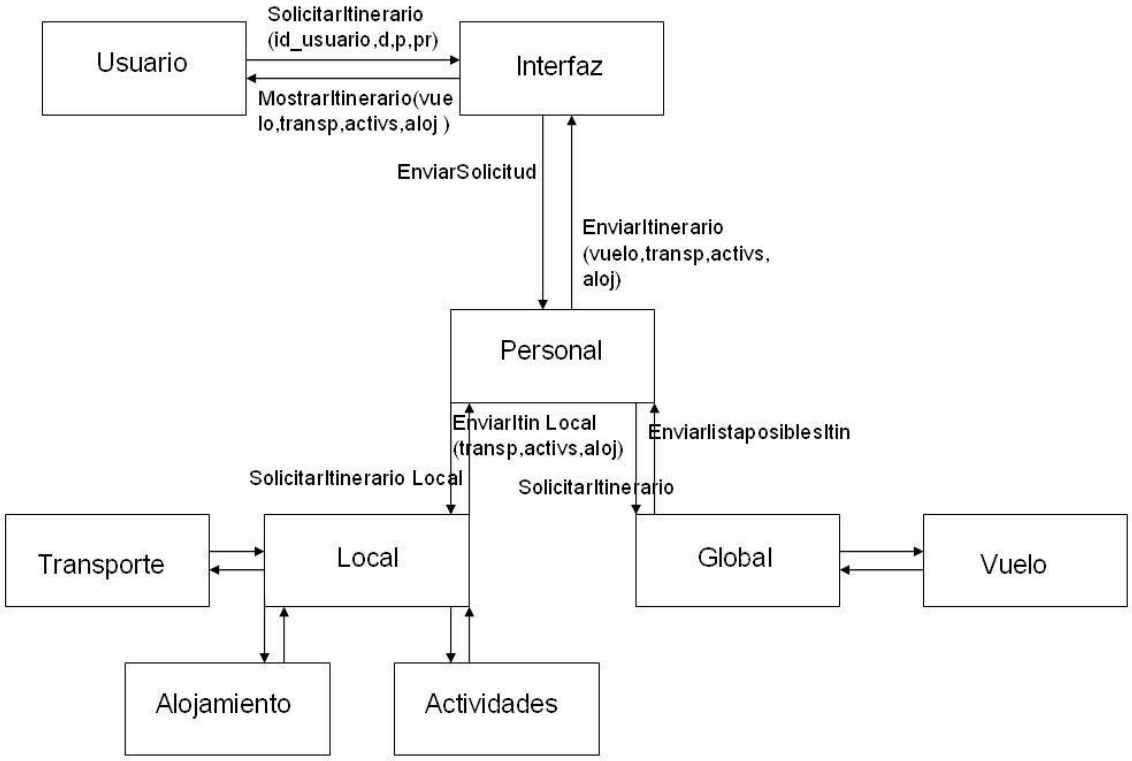

Figura 22: Diagrama de Flujo de los Eventos

Se puede usar HMSC (Diagrama de secuencia de alto nivel de mensajes), el cual es muy útil para este propósito. Estos diagramas muestran la ruta (fases) del protocolo y como las diferentes fases (especificadas con msc) son combinadas. Una fase puede ser un simple MSC u otro HMSC.

El procesamiento de las interacciones es descrito usando el diagrama estático SLD. Dicho diagrama es necesario para llenar la plantilla de protocolo textual especificando la capacidad de razonamiento requerida en los participantes del protocolo.

El diagrama estático considera 3 tipos de eventos: eventos de mensajes, eventos desde otros agentes usando intercambio de mensajes, eventos externos, eventos de entorno observados a través de sensores, y eventos internos, eventos que surgen en un agente debido a su actitud proactiva.

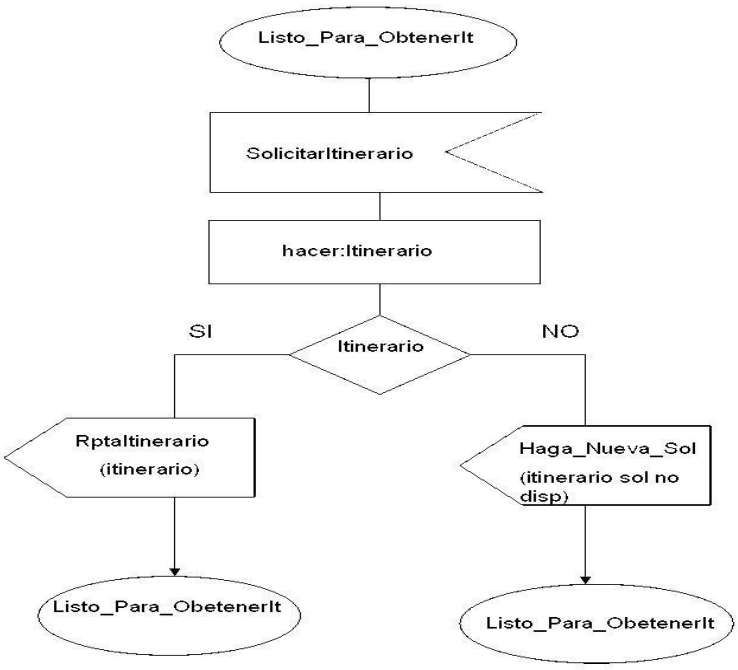

Figura 23: Diagrama Estático SLD

Desarrollar este Modelo trae consigo potenciales beneficios:

El desarrollo del Modelo de Coordinación es un medio para especificar las interacciones de los protocolos entre los agentes trabajando en la solución del problema, junto con las interacciones del medio. Este Modelo es utilizado para almacenar la estructura de las comunicaciones y los protocolos asociados con estas comunicaciones. El uso de estas descripciones es doble: el diseñador puede reutilizar los protocolos y escenarios y el agente inteligente puede seleccionarlos al tiempo que este corriendo

MSC y SLD son técnicas de descripción formal. El uso de estos lenguajes para especificar interacciones en Sistemas Multi-Agentes han sido alcanzadas por:

1-Definición de un tipo de señal para cada acto de habla posible.

2-Asociación de una expresión lógica para cada nombre estado (usando comentarios).

3-Considera eventos internos (similares para transiciones espontáneas) para cambiar en el estado mental del agente motivado debido a su actitud proactiva.

4-El desarrollo de este modelo puede ayudar en el mantenimiento y la prueba de un sistema Multi-Agente.

Es importante destacar que este modelo propuesto por la metodología MAS-CommonKADS presenta deficiencias en su notación gráfica MSC.

Modelo de Conocimiento (Experiencia)

El Modelo de Experiencia es usado para la modelación de las capacidades de los agentes para llevar a cabo sus tareas y conseguir sus objetivos. Normalmente, varias instancias del Modelo de Experiencia deben ser desarrolladas: modelando inferencias en el dominio (Ej. Como predecir que cierto itinerario no está disponible para ese usuario.); modelando el razonamiento de los agentes (problema de solución de métodos para conseguir una tarea, carácter del agente, etc.) y modelando las inferencias del ambiente (como el agente puede interpretar el evento que recibe desde otros agentes o desde el mundo). Cuando se tiene que desarrollar las capacidades de razonamiento de un agente, se volverán a emplear las instancias previamente elaboradas en el Modelo de Experiencia y adaptándolas a las nuevas características del problema.

El Modelo de Experiencia desarrolla el conocimiento de la aplicación (consistencia del conocimiento de domino, conocimiento de inferencia y conocimiento de tarea) y el conocimiento de la solución del problema.

Conocimiento de Dominio: Representa el conocimiento del problema, modelado como conceptos, propiedades, expresiones y relaciones usando el Modelo Conceptual de Lenguaje (CML: Conceptual Modelling Language) o la notación gráfica del Modelo Objeto.

En este problema, si se enfoca en el dominio, se pueden identificar conceptos como: itinerario, itinerario local. etc. y propiedades como: p(precio), pr (preferencias). Estos conceptos son colocados en el Modelo de Dominio.

Conocimiento de Inferencia: Representa los caminos de inferencia realizados para solucionar una tarea.

Existe una librería de estructuras de inferencias genéricas seleccionadas por una tarea (diagnóstico, valoración). Estas estructuras de inferencias genéricas deben ser adaptadas al problema.

Consultando la librería para modelar como predecir cuando un itinerario irá a estar disponible. La tarea Itinerario no es apropiado porque ésta es usada para sugerir qué pasará en el próximo sistema. Se puede decir que la tarea más apta sería *Valoración,* cuya estructura de inferencia (suponiendo que no es una norma disponible). Conocimiento de Tarea: representa el orden de las estructuras de inferencia. Las anotaciones están compuestas por estructuras de inferencia y estructuras de descomposición de inferencia del método de tarea.

Método de solución del problema: durante el diseño se debe especificar el Método de Solución del Problema para cada tipo de inferencia: como la inferencia es llevada a cabo. Los métodos de solución del problema son arreglados en las librerías para su reutilización.

El potencial beneficio con el desarrollo de este Modelo es el empleo del bien conocido nivel de modelación de framework y la provisión de una librería de componentes genéricos, especificación de lenguajes y herramientas software.

Es importante destacar que este modelo tiene como uno de sus principales problemas es que presenta un vocabulario un poco complejo para su compresión.

Modelo de Organización

CommonKADS define este modelo para modelar la organización en el cual el sistema basado en el conocimiento será introducido. Aquí el modelo es extendido en el mismo sentido que el modelo agente, modelando la organización de los mismos.

Con este modelo se muestran las relaciones dinámicas.

Las notaciones gráficas de este modelo están basadas en la notación del Modelo de Objeto, agregando un símbolo especial para distinguir entre agentes y objetos.

El símbolo de agregación es usado para expresar los grupos de agentes.

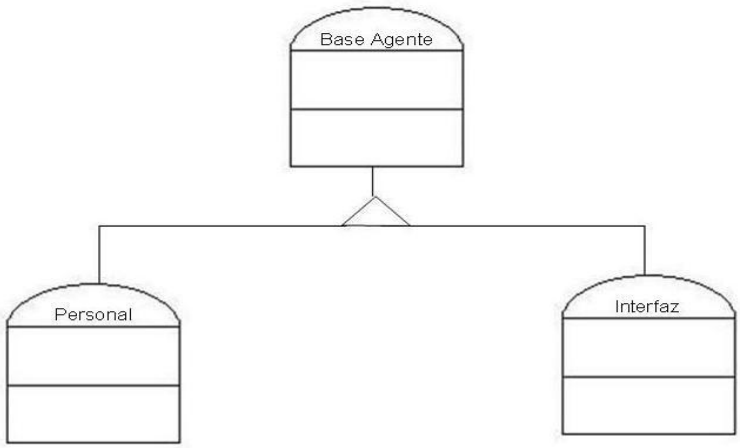

Figura 24: Diagrama de Clase de Agente

El símbolo de agente es muy semejante al símbolo de clases propuesto en el Modelo de Objeto pero tiene un significado diferente. El cuadro superior almacena los atributos definidos como en el modelo de objeto pero el estado mental y los atributos externos de los agentes, tales como sus objetivos, creencias, planes, etc. El cuadro inferior almacena los atributos externos de los agentes: servicios, sensores y efectores.

La relación de herencia entre los agentes es definida como la unión de los valores de las clases precedentes para cada atributo. Por ejemplo: una clase de agente tiene sus objetivos y los objetivos de las clases de agentes precedentes. Si un agente define un atributo como exclusivo, los valores se sobrescriben.

Este modelo tiene beneficios potenciales puesto que especifica la relación estructural entre humanos y/o agentes software, y la relación con el ambiente.

El estudio de la organización es una herramienta para la identificación de posibles impactos en Sistemas Muti-Agentes cuando sean instalados. Del mismo modo este modelo puede proveer información sobre las funciones, flujo de trabajo, procesos y estructura de la organización que permite el estudio de la viabilidad de las soluciones propuestas.

Este Modelo representa ambos diagramas de clases de agente, mostrando la particular relación con el medio ambiente.

De acuerdo con otro paradigma (orientado a objeto), los diagramas de instancias de agente son frecuentemente más relevantes que los diagramas de clases de agentes.

2.4 Diseño

Como resultado de la fase de análisis, un conjunto inicial de agentes ha sido identificado. Durante la fase de diseño el modelo de diseño es desarrollado. Esta fase consiste en:

Diseño de la red de Agente: La infraestructura del sistema-MAS (también llamado modelo de red) es determinado, y consiste en la red , conocimientos y facilidades de coordinación. Los agentes (también llamados agentes de red) que mantiene esta infraestructura también definida, dependiendo de las facilidades requeridas. Algunas de estas facilidades requeridas pueden ser:

- Facilidades de red: servicio nombre de agente, servicio de páginas amarillo/blancas, nivel de seguridad, encriptación y autenticación, transporte, etc.
- Facilidades de conocimiento: lenguaje de traducción de la representación de conocimiento, etc.

Facilidades de Coordinación: Protocolos disponibles de coordinación, protocolo del servidor, facilidades del grupo de administradores, facilidades para la colaboración y coordinación de los objetivos compartidos.

El resultado de compartir las facilidades comunes por los agentes permite la comunicación eficiente entre ellos.

Diseño de Agente: La arquitectura más adecuada es determinada por cada agente y donde alguno de ellos pueden ser introducidos o subdivididos de acuerdo con el criterio práctico. Cada agente es subdividido en módulos para la comunicación de agentes (del modelo de coordinación), deliberación y reacción (del modelo de experiencia, agente y modelos de organización), habilidades externas y servicios (del modelo de agente, modelo de experiencia y modelo de tareas).

El diseño de agente planea las funciones definidas en estos módulos sobre la arquitectura de agente seleccionada.

El tema del diseño de la arquitectura de un agente no está planteado en esta metodología, desde que la arquitectura de agente está alimentada por la plataforma de desarrollo de agente.

Diseño de Plataforma: Selección del software y hardware que es necesitado (o disponible) para el sistema.

Los beneficios potenciales de desarrollar este Modelo son:

- Las decisiones en la selección de la plataforma Multi-Agente y la arquitectura de un agente por cada agente documentado.
- El Modelo de diseño archiva la información de los modelos desarrollados previamente y detalla como estos requerimientos pueden ser cumplidos.
- El Modelo de Diseño para sistemas Multi-Agentes determina los recursos comunes y las necesidades de los agentes .También diseña la administración de una infraestructura común por los agentes de red.

Esto facilita modularidad en el diseño.

Aplicado a este caso de estudio propuesto:

Diseño de RED-Red-TurOperador

Tipo-red de MAST (Multiagent System Tool)

Servicios-de-red: Registrarse, eliminarse, actualizar

Agente de red: ofrece la mayor parte de las funciones del nivel de red (páginas blancas/amarillas).

Diseño de plataforma: Se toman decisiones de software y hardware.

Plataforma: Sistema Multi-Agente.

Descripción: La plataforma seleccionada ha sido MAST por ser la única disponible. Esta plataforma dispone del lenguaje ADL (Lenguaje de Descripción de Agentes) y permite programar los agentes en C++ o en JAVA. El sistema está disponible para Unix o Linux (versión C++) y Windows (versión JAVA).

Hardware requerido: Computadoras cuyo sistema operativo sea Linux o Windows (depende en el lenguaje que se vaya a programar el agente).

Software requerido: La plataforma MAST, compilador de C++ o de JAVA .

La fase de implementación y prueba no son cubiertos en esta metodología porque son muy dependientes de la plataforma empleada.

Conclusiones

MAS-CommonKADS proviene de una metodología orientada al conocimiento, lo cual implica que presente una buena base para modelar sistemas Multi-Agentes debido a que se ocupa del desarrollo de sistemas basados en el conocimiento de agentes. No obstante a ello presenta demasiada teoría en sus modelos y genera pocos artefactos en cada uno de ellos, lo cual hace que sea menos comprensible y aplicable que otras metodologías.

Es fundamental resaltar que esta metodología concibe un sistema basado en conocimiento como un sistema centralizado y presenta limitaciones para modelar aspectos distribuidos o sociales de los agentes.

MAS-CommonKADS no es una metodología de gran eficiencia, puesto que el Modelo de Coordinación que ella propone presenta problemas. Además la herramienta de trabajo no está disponible en Internet de forma gratuita.

Se ha podido comprobar que MaSE constituye una extensión de las metodologías orientadas a objetos y presenta similitudes con éstas, la misma provee una sólida estructuración de cada una de las etapas del proceso de desarrollo, teniendo sus objetivos bien definidos en cada fase y generando artefactos que viabilizan la solución del problema.

Se considera a MaSE una metodología bastante eficiente que lleva a cabo con precisión sistemas prácticos y en menor tiempo de desarrollo. La herramienta de soporte presenta problemas a la hora de su instalación y se requiere licencias para instalarla.

El Caso de Estudio presentado constituye un problema que engloba una serie de características idóneas para ser tratadas con metodologías orientadas a agentes, lo que permite aplicar a MaSE y MAS-CommonKADS de forma satisfactoria al mismo, llevándose a cabo un ciclo de desarrollo de ambas metodologías que posteriormente será el basamento para el establecimiento de comparativas entre ellas.

Capítulo 3 Evaluación de las Metodologías MaSE y MAS-CommonKADS

Introducción

El presente capítulo se centra en un análisis evaluativo de ambas metodologías: MaSE y MAS-CommonKADS, atendiendo a un determinado número de criterios que serán expuestos en el desarrollo del mismo, donde se hará alusión esencialmente a sus debilidades y fortalezas. Este análisis tiene su basamento en comparativas que se establecen entre MaSE y MAS-CommonKADS, teniendo en cuenta los aspectos más significativos de ellas. En el mismo se expone un framework de comparación para las dos metodologías.

3.1 Comparación entre MaSE y MAS-CommonKADS.

En los últimos años se han propuesto numerosas metodologías para facilitar y soportar el desarrollo de sistemas distribuidos complejos. Por tanto, cuando se plantea la construcción de una aplicación de agentes es importante decidir qué metodología utilizar. Es por ello que con el objetivo de ayudar en esta decisión, se propone un framework de evaluación entre las metodologías: MaSE y MAS-CommonKADS, estableciendo un conjunto de criterios que tienen en cuenta aspectos de ingeniería de software, así como de la tecnología de agentes.

Para la comparación entre ambas metodologías se abordarán criterios tales como: dominio de aplicación, ciclo de vida, Modelos, etapas de desarrollo, roles del equipo de desarrollo, lenguaje de modelado, arquitectura interna, herramienta de soporte, captura de requisitos, notación, autonomía, comunicación, protocoles, usabilidad y pragmatismo.

MaSE es una metodología diseñada fundamentalmente para ingeniar Sistemas Multi-Agentes prácticos, mientras que MAS-CommonKADS tiene como enfoque principal la construcción de sistemas expertos.

Por otra parte MaSE tiene gran acercamiento a la orientación a objetos ya que ésta considera al agente como un objeto que puede o no tener inteligencia, por lo que los componentes inteligentes como no inteligentes son gestionados en un mismo proceso. La misma es independiente de dominio. En cambio MAS-CommonKADS define el ciclo de desarrollo de una aplicación empleando una perspectiva de agente. Esta metodología clasifica a los agentes en: agentes software y agentes humanos realizando una especificación de los mismos para capturar sus habilidades y restricciones.

En MaSE se plantea que la metodología cubre todo el ciclo de desarrollo de un software, siendo esto totalmente incierto ya que existen fases como la de prueba y captura de requisitos en las cuales no se proporciona información suficiente de como deben ser llevadas a cabo. Por su parte MAS-CommonKADS no cubre la fase de implementación y prueba.

MAS-CommonKADS es una metodología que ayuda a la construcción de Sistemas Multi-Agentes completos y robustos. Dentro de su definición destaca que se ha incluido una fase de conceptualización, considerada como una fase informal de toma de contacto del problema puesto que el análisis del mismo se realiza desde el punto de vista del usuario. MaSE también presenta problemas en la captura de requisitos iniciales del sistema, los cuales son concebidos, pero en cambio no se aporta información de cómo deben ser capturados.

MaSE presenta dos etapas fundamentales para su desarrollo, las cuales son: análisis y diseño. Cada una de estas etapas contiene un conjunto de pasos que generan artefactos, los cuales presentan dependencias entre sí. Por otro lado MAS-CommonKADS cuenta con las siguientes etapas de desarrollo: conceptualización, análisis, diseño, codificación y prueba de cada agente, integración (donde el sistema completo es probado), operación y mantenimiento, por lo que se puede apreciar que MAS-CommonKADS cuenta con más etapas que le permiten abarcar de manera factible todo el proceso.

En teoría se puede considerar como una metodología muy eficiente pero en la práctica resulta ser todo lo contrario porque como se menciona anteriormente no cubre la etapa de implementación y prueba, la fase de conceptualizaron es un tanto informal y presenta deficiencias en la notación gráfica. .

En el modelo de implementación MaSE propone interfaces y dependencias entre las mismas. En la etapa de pruebas, MaSE no proporciona documentación de cómo debe ser tratada la misma, por lo que deja

esta etapa a iniciativa de los desarrolladores. No sucede así con MAS-CommonKADS quien presenta una etapa de integración en la cual se puede comprobar parcialmente la corrección de la conducta global del sistema utilizando los escenarios típicos que tratan los posibles conflictos y los métodos de resolución de los mismos. Pero, dado que la conducta global del sistema no puede ser determinada durante la fase de análisis o diseño, porque depende de los acuerdos y compromisos concretos realizados entre los agentes, normalmente se necesitará recurrir a la simulación [5].

MaSE y MAS-CommonKADS por su parte no identifican un amplio conjunto de roles a ser interpretados por el equipo de desarrollo, por lo que son el analista y el diseñador quienes llevan a cabo la mayor parte del trabajo en ambas metodologías. Esto constituye un punto débil, ya que no le atribuyen ninguna importancia a la organización y la especialización del equipo de desarrollo.

Como se pudo observar en los capítulos anteriores MaSE presenta un lenguaje de especificación basado en UML+OCL. UML es una especificación de notación orientada a objetos. Se basa en las anteriores especificaciones BOOCH, RUMBAUGH y COAD-YOURDON. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del sistema, e introduce diagramas que representan una visión dinámica del mismo. UML intenta solucionar el problema de propiedad de código de los desarrolladores al implementar un lenguaje de modelado común para todos los desarrollos [8].

UML constituye un estándar. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, pues UML ha sido diseñado para modelar cualquier tipo de sistemas, tanto informáticos como de cualquier otra rama [4].

Por su parte OCL (Object Constraint Language) es un lenguaje para la descripción formal de expresiones en los modelos UML. Sus expresiones pueden representar invariantes, precondiciones, poscondiciones, inicializaciones, guardias, reglas de derivación, así como consultas a objetos para determinar sus condiciones de estado.

Se puede mencionar que MAS-CommonKADS utiliza un lenguaje declarativo denominado ADL (Lenguaje de Definición de Agentes) para definir las clases de agentes y sus habilidades básicas de agentes. Para la transmisión de conocimiento utiliza el lenguaje de representación CKRL (lenguaje común de representación del conocimiento).

Una de las mayores desventajas que presenta MaSE es la no utilización de los conceptos BDI (Believe Desire Intention) a través de todo su ciclo de desarrollo. Las arquitecturas BDI constituyen una forma de guiar los agentes como procesos autónomos independientes, estableciendo un paralelismo evidente con el modo de funcionamiento de un trabajador humano. Los agentes utilizan un modelo del mundo, una representación de cómo se les muestra el entorno. El agente recibe estímulos a través de sensores ubicados en el mundo. Estos estímulos modifican el modelo que tiene el agente el cual está representado por un grupo de creencias.

El proceso de desarrollo de MaSE es llevado a cabo por un conjunto de pasos que se ejecutan dentro de una herramienta denominada Agenttool. Dentro de esta misma herramienta, MaSE incorpora utilidades para verificar la corrección de los protocolos que utilicen los agentes, siendo ésta mucho más efectiva que la que proporciona MAS-CommonKADS, que se denomina MAST (Multiagent System Tool) que ofrece funcionalidades básicas de comunicación entre agentes.

MaSE propone un proceso más simple que el de MAS-CommonKADS, por lo que ésta más bien responde al hecho de tomar la herramienta y llevar a cabo en ella los pasos pertinentes para diseñar el sistema. La ventaja que ofrece este enfoque es que se aprende rápido ya que sólo se tiene que aprender a utilizar la herramienta de soporte. No obstante esta forma de modelar proporciona desventajas ya que se obvia que existen dependencias entre los diagramas propuestos (como entre los diagramas de secuencia y las conversaciones) y que no es tan sencillo el saber qué máquinas de estados definen la ejecución de una tarea en el contexto de una interacción.

Es válido resaltar que MaSE usa máquinas de estados que le permiten especificar el comportamiento de distintos elementos y aunque la propia metodología no hace mención a ello, esta idea no es original, ya que la misma es aplicada en UML y lleva dos décadas de utilización en la industria.

Entre una de sus principales debilidades, se puede resaltar el hecho de que MaSE no considera elementos característicos de la tecnología de agentes tales como: razonamiento, organización de los agentes y caracterización de su estado mental. En cambio, MAS-CommonKADS especifica las características de los agentes tales como: clase de agente, capacidad de razonamiento, habilidades, sensores, efectores, servicios y grupo de agentes a los que pertenece. En esta metodología un agente

puede ser un agente humano, software o cualquier entidad capaz de empelar un lenguaje de comunicación de agentes. Los agentes constituyen un elemento esencial en esta metodología. El Modelo de Agente constituye el modelo más importante sirviendo de enlace con el resto de los modelos del análisis.

Por otra parte en ambas metodologías se plantea que son iterativas, por lo que le permiten al diseñador moverse libremente entre pasos y etapas, donde a cada etapa pasada se le añaden nuevos detalles y finalmente se produce un completo y consistente diseño del sistema, pero todo esto no se cumple en MAS-CommonKADS puesto que su modelo de coordinación puede desarrollarse de forma independiente o con otra metodología.

En cuanto a los requisitos iniciales del sistema, MaSE los captura a través de un conjunto de roles que plasman los requisitos críticos del sistema y capturan lo que se intenta conseguir, pero no describe cómo deben ser capturados. No sucede así con MAS-CommonKADS, quien desde un inicio captura los requisitos del sistema en la fase conceptuación, en la cual se lleva a cabo un análisis centrado en el usuario, siendo su principal objetivo la comprensión de los requisitos de los usuarios para construir el sistema que se ajusta a sus necesidades.

MaSE presenta un modelado basado en roles. Los roles son usados para identificar los diferentes tipos de agentes por los que el sistema debe estar compuesto. Estos roles se le asignan a los objetivos iniciales del sistema que anteriormente han sido trazados. A diferencia de MaSE, MAS-CommonKADS propone 7 modelos donde cada uno de ellos está basado en una teoría diferente. Los modelos son: Modelo de Agente, Modelo de Organización, Modelo de Tareas, Modelo de Experiencia, Modelo de Coordinación, Modelo de Coordinación, Modelo de Diseño

Ambas metodologías por su parte, presentan deficiencias a la hora de la utilización de ontologías, ya que ninguna posee información sobre ello y en sus respectivas documentaciones no tratan este importante tema.

En cuanto a la autonomía de los agentes MAS-CommonKADS describe ésta a través de tareas, lo que le permite a los agentes ejecutarse por sí mismos, mientras que MaSE no considera esta característica de los agentes.

MaSE provee una notación sencilla que le permite establecer comunicación entre los agentes y los diversos artefactos generados en cada fase de desarrollo. MaSE fundamentalmente usa dos eventos especiales para indicar los mensajes que se envían entre agentes, los cuales son envío y recepción, que se llevan a cabo en los diagramas de tareas concurrentes.

Los protocolos en MasE describen el intercambio de mensajes en relación al procesamiento dentro de agentes. En los protocolos son de vital importancia para la capacitación de los agentes para su comunicación e interacción. Se utiliza el protocolo de contratos, siendo este un protocolo de alto nivel muy conocido en los Sistemas Multi-Agentes puesto que facilita el control distribuido de la ejecución de tareas.

MaSE posee documentación disponible en Internet, aunque la mayor parte de dicha información es presentada en inglés. También se puede acceder a descargar de forma gratuita la herramienta que la soporta. El agenttool se encuentra gratis en diferentes versiones, pero no obstante a ello presenta problemas a la hora de su instalación ya que requiere de componentes que se usan en Java y que son versiones muy viejas y difíciles de encontrar, tal es el caso del JDK 1.3.1 de Java. MAS-CommonKADS por el contrario no tiene una documentación enriquecida y la herramienta de soporte no es pública en Internet por lo que resulta muy difícil su obtención y para acceder a ella es necesario comprarla.

Los aspectos que se han tratado en el presente Capítulo, son esenciales para evaluar cualquier metodología orientada a agentes, por lo que éstos dan una idea de las fortalezas y debilidades de las metodologías en las que se evalúen.

3.2 Framework: MaSE

Áreas de aplicación

Comentario general: En sentido general no existen muchas publicaciones sobre proyectos específicos que hayan sido modelados con MaSE.

Áreas concretas: Robótica X Se menciona __ Hay documentación __Se menciona __Hay documentación

Sistemas abiertos

Comentario: MaSE no brinda mecanismos que permitan la creación o duplicación de Agentes en tiempo de ejecución.

Evaluación: __Sí X No __No se sabe

FASE 1:

Nombre: Análisis

Breve explicación: Este nivel cubre las fases relacionadas a captura de objetivos, aplicación de los casos de uso y refinamiento de roles

El mismo puede ser homologado funcionalmente con el flujo de trabajo Captura de Requisitos de MAS-CommonKADS.

Requisitos: __Centrado X Centrado parcialmente __No centrado

Análisis: X Centrado __Centrado parcialmente __No centrado

Diseño: _ Centrado X Centrado parcialmente _ No centrado

Implementación: __Centrado __Centrado parcialmente X No centrado

Pruebas: __Centrado __Centrado parcialmente X No centrado

Despliegue: __Centrado __Centrado parcialmente X No centrado

Mantenimiento: __Centrado __Centrado parcialmente X No centrado

Implicación del usuario: __Centrado __Centrado parcialmente _ No centrado X Desconocido

FASE 2:

Nombre: Diseño

Breve explicación: Este nivel proporciona una modelación de la arquitectura de la solución en términos de clases y métodos. Se crean clases de agentes a partir de los roles identificados en el nivel anterior, y se llevan a cabo las conversaciones entre éstos y el diseño del sistema

Requisitos: __Centrado _ Centrado parcialmente X No centrado

Análisis: _ Centrado __Centrado parcialmente X No centrado

Diseño: X Centrado __Centrado parcialmente __No centrado

Implementación: __Centrado __Centrado parcialmente X No centrado

Pruebas: __Centrado X Centrado parcialmente __ No centrado

Despliegue: __Centrado __Centrado parcialmente X No centrado

Mantenimiento: __Centrado __Centrado parcialmente X No centrado

Implicación del usuario: __Centrado __Centrado parcialmente _ No centrado X Desconocido.

1. Vista de modelos

Conceptos y representaciones por:

Objetivos	Diagrama Jerárquico de Objetivos
Casos de Uso	Casos de usos
Roles	Diagrama de Secuencias, Modelo de Roles
Eventos	Diagramas de Secuencias
Tareas	Modelo de Rol, Diagrama de Tareas Concurrentes
Ejecución de Mensajes	Diagrama de Tareas Concurrentes
Actividades	Diagrama de Tareas Concurrentes
Clases de agentes	Diagrama de Clases de Agentes, Diagrama de Despliegue
Conversaciones	Diagrama de Clases de Agentes, Diagrama de Comunicación de Clases
Arquitectura de Agentes	Diagrama de Arquitectura de Agentes
Instancias de Agentes	Diagrama de despliegue

Tabla 2: Características MaSE

Análisis:

Objetivo	Diagrama Jerárquico de Objetivos
Casos de Uso (entidad, camino de comunicación)	Casos de usos
Rol, Evento	Diagrama de Secuencias
Rol, Tarea	Modelo de Roles
Tarea, mensaje (ejecución), Actividad	Diagramas de Tareas Concurrentes

Tabla 3: Análisis MaSE

Diseño:

Clases de Agentes, conversación	Diagrama de Clases de Agentes
Conversaciones (coordinación de protocolos)	Diagrama de Comunicación de Clase
Arquitectura de Agentes	Diagrama de Despliegue
Agentes (instancia), Clase de Agente	

Tabla 4: Diseño MaSE

Atributos del agente:

Autonomía: __Sí X No __No se sabe

Sociabilidad: __Sí X No __No se sabe

Reactividad: __ Sí X No __No se sabe

Proactividad: X Sí __No _ No se sabe

Inteligencia: _ Sí X No _ No se sabe

Veracidad: _ Sí X No __No se sabe

Benevolencia: _Sí X No __No se sabe

Racionalidad: __Sí __No X No se sabe

Cooperación: X Sí __No __No se sabe

Otras:

Conocimiento X Sí __No __No se sabe

Tipos de comunicación:

Comentario: Se modelan las comunicaciones entre Agentes solamente.

Tipos: X A-A __A-H __A-O __Cualquiera

Protocolos de comunicación:

Comentario: Se proponen los definidos por FIPA. En caso de no ser adecuados, se referencia consultar la documentación propuesta por la Fundación para la confección de Protocolos de Interacción.

Protocolos: _ Predefinido X Definido por el usuario __Definida por metodología __No se sabe

Cooperación:

Cooperación: _Cualquiera X No se sabe

Organización:

Comentario: La organización no se conoce. Esto constituye una deficiencia que la sitúa, en este aspecto, por debajo de otras metodologías que sí le dan tratamiento.

Organización: __Predefinida __Definida por el usuario __Definida por metodología X No se sabe

3. Modelado de características adicionales

Ontologías:

Evaluación: _ Sí X No __ No se considera

Movilidad:

Comentario: En el Modelo de Despliegue se tiene en cuenta la movilidad de los Agentes entre las diversas unidades de procesamiento a través de los caminos físicos dispuestos para ello.

Evaluación: X Sí __No __No se considera

Otras:

Comentario:

Evaluación: __Sí __No __No se considera

4. Documentación

Documentación disponible:

Explicación sobre el tipo de documentos: La documentación referente a la metodología es netamente funcional. Evaluación: __Buena _ Suficiente X Pobre

Casos de estudios presentados:

Evaluación: __Trivial X Parcial __Completo

Aspectos adicionales a evaluar

1. Proceso de desarrollo

Plataforma de desarrollo:

Comentario: Se considera que MaSE fue concebida para ser soportada por una herramienta denominada Agenttool. No obstante, al aplicarla manualmente, se considera independiente de plataforma alguna.

Completitud de los flujos de trabajo:

Comentario: MaSE no refleja la captura de Requerimientos Iniciales ni las etapas de Prueba. De igual forma, sería provechoso incluir tratamiento al mantenimiento de los sistemas desarrollados.

Evaluación: __Bueno _ Regular X Mal

A que fase se le da más importancia:

Comentario: Los desarrolladores de la metodología no evidencian superioridad en alguna etapa respecto a las demás.

Evaluación: __Sí __No XNo se sabe

A que flujo de trabajo se le da más importancia:

Comentario: No se evidencia jerarquía alguna entre los flujos de trabajo.

Evaluación: __Sí __No X No se sabe

El ciclo de vida permite llegar a la implementación:

Comentario: MaSE propone el desarrollo completo de un SMA, desde Requerimientos hasta Código (no obstante no definir cabalmente la captura de Requerimientos Iniciales)

Evaluación: X Sí __No __No se sabe

2. Vista de modelos

Lenguaje de modelado:

Lenguaje: UML y OCL

Claridad de la notación de los estereotipos:

Comentario: Esta evaluación se tiene en cuenta considerando experiencia en desarrollo con UML.

Evaluación: X Bueno __Regular __Mal

Facilidad de uso de los estereotipos:

Comentario: Esta evaluación se tiene en cuenta considerando experiencia en desarrollo con UML.

Evaluación: X Bueno __Regular __Mal

Entendimiento de los estereotipos:

Comentario: Esta evaluación se tiene en cuenta considerando experiencia en desarrollo con UML.

Evaluación: X Bueno __Regular __Mal

Facilidad de uso del lenguaje de modelado:

Comentario: Esta evaluación se tiene en cuenta considerando experiencia en desarrollo con UML.

Evaluación: X Bueno __Regular __Mal

Entendimiento del lenguaje de modelado:

Comentario: Esta evaluación se tiene en cuenta considerando experiencia en desarrollo con UML.

Evaluación: X Bueno __Regular __Mal

3. Agentes

Modelado de la inteligencia o conocimiento del agente:

Comentario: Para MaSE los agentes pueden ser o no inteligentes.

Evaluación: __Si X No __Parcialmente __No se sabe

Tiene pasos claros el modelado de la inteligencia:

Evaluación: __Si X No __Parcialmente __No se sabe

Completitud del modelado de la inteligencia:

Evaluación: __Bueno __Regular X Mal

Facilidad del modelado de la interacción:

Comentario: MaSE no propone un diagrama responsable únicamente de definir las interacciones entre los Agentes en términos de Lenguaje de Comunicación, Ontología y Protocolo de Interacción.

Evaluación: __ Bueno __Regular X Mal

4. Documentación

Aplicaciones que han utilizado la metodología:

Comentario: No existe referencia sobre proyectos reales de aplicación de MaSE. Los estudios actuales sobre MaSE y agentTool se concentran en dar soporte a agentes móviles.

5. Otros.

Facilidad de entendimiento de la metodología:

Comentario: La metodología es de fácil entendimiento para desarrolladores ya que la misma se centra en la utilización de la herramienta y está basada en la creación de sistemas Multi-Agentes prácticos.

Evaluación: X Bueno __Regular __Mal

Aspecto más cómodo de la metodología:

Comentario: El aspecto más cómodo de MaSE está dado en la adopción de UML como lenguaje de modelado, pues es bastante difundido. Cabe señalar además que el Agenttool sustenta la aplicación de todas las etapas, y que esta puede ser accedida de forma gratis en Internet aunque se requiere de licencia para su instalación.

Aspecto más incómodo de la metodología:

Comentario: El punto más incómodo de MaSE es la no consideración de inteligencia en los agentes. También otro aspecto es la no utilización de arquitecturas BDI.

Principales ventajas:

- Incorpora como lenguaje de modelado UML + OCL.
- Posee una herramienta que sustenta la mayoría de las etapas de aplicación.
- Existe documentación en inglés que posibilita su aplicación.
- Define con claridad cada etapa de la metodología.
- Permite la generación automática de modelos y código a partir de estos.
- Define el proceso de manera iterativa
- Es independiente de dominio de aplicación y de plataformas de desarrollo.

Principales desventajas:

- No modela Requerimientos Iniciales, Prueba, ni Mantenimiento.
- No incorpora tratamiento a la inteligencia y aprendizaje de los Agentes.
- No sustenta sistemas abiertos.

Implementación:

Generación del código X Sí __No __Soportada parcialmente __No se sabe

Orientada a una herramienta __Sí X No __Soportada parcialmente __No se sabe

Pruebas: __Sí X No __Soportada parcialmente __No se sabe

Despliegue: X Sí __No __Soportada parcialmente __No se sabe

3.3 Framework: MAS-CommonKADS

Áreas de aplicación:

Comentario general: En sentido general no existen muchas publicaciones sobre proyectos específicos que hayan sido modelados con MAS-CommonKADS

Áreas concretas: __Se menciona X Hay documentación __

Sistemas abiertos:

Comentario: MAS-CommonKADS no brinda mecanismos que permitan la creación o duplicación de Agentes en tiempo de ejecución.

Evaluación: Sí X No __No se sabe

FASE 1:

Nombre: Conceptualización

Breve explicación: Este nivel cubre las fases relacionadas a captura de requisitos e identificación de los casos

El mismo puede ser homologado funcionalmente con el flujo de trabajo Captura de Requisitos de MaSE.

Requisitos: __Centrado X Centrado parcialmente __No centrado

Análisis: X Centrado __Centrado parcialmente __No centrado

Diseño: _ Centrado X Centrado parcialmente _ No centrado

Implementación: __Centrado __Centrado parcialmente X No centrado

Pruebas: __Centrado __Centrado parcialmente X No centrado

Despliegue: __Centrado __Centrado parcialmente X No centrado

Mantenimiento: __Centrado __Centrado parcialmente X No centrado

Implicación del usuario: __Centrado __Centrado parcialmente _ No centrado X Desconocido

FASE 2:

Nombre: Análisis

Breve explicación: En este nivel se desarrollan los modelos de la metodología.

Requisitos: __Centrado __ Centrado parcialmente X No centrado

Análisis: X Centrado __Centrado parcialmente X No centrado

Diseño: __ Centrado X Centrado parcialmente __No centrado

Implementación: __Centrado __Centrado parcialmente X No centrado

Pruebas: __Centrado __ Centrado parcialmente X No centrado

Despliegue: __Centrado __Centrado parcialmente X No centrado

Mantenimiento: __Centrado __Centrado parcialmente X No centrado

Implicación del usuario: __Centrado __Centrado parcialmente _ No centrado X Desconocido.

Vista de modelos

Conceptos y representaciones por:

Objetivos	
Casos de Uso	Casos de usos
Roles	Los roles se identifican en la fase de coneptualización
Eventos	
Tareas	Se determinan y describen las tareas que se le asignarán a los agentes en el Modelo de Tareas
Ejecución de Mensajes	
Actividades	
Clases de agentes	Comunicación de Clases
Conversaciones	Diagrama de Caso de Uso de Agentes

Arquitectura de Agentes Instancias de Agentes	

Tabla 5: Características MAS-CommanKADS

Análisis:

Objetivo Casos de Uso (entidad, camino de comunicación) Rol, Evento Rol, Tarea Tarea, mensaje (ejecución), Actividad	Casos de usos

Tabla 6: Análisis MAS-CommanKADS

Diseño:

Clases de Agentes, conversación Conversaciones (coordinación de protocolos) Arquitectura de Agentes Agentes (instancia), Clase de Agente	

Tabla 7: Diseño MAS-CommanKADS

Atributos del agente:

Autonomía: X Sí __No __No se sabe

Sociabilidad: X Sí __No __No se sabe

Reactividad: X Sí __No __No se sabe

Proactividad: X Sí __No _ No se sabe

Inteligencia: X Sí _ No _ No se sabe

Veracidad: X Sí _ No __No se sabe

Benevolencia: X Sí _ No __No se sabe

Racionalidad: __Sí __No X No se sabe

Cooperación: X Sí __No __No se sabe

Otras:

Conocimiento X Sí __No __No se sabe

Tipos de comunicación:

Comentario: Se modelan las comunicaciones entre Agentes solamente.

Tipos: X A-A __A-H __A-O __Cualquiera

Protocolos de comunicación:

Comentario: Se proponen los definidos por FIPA. En caso de no ser adecuados, se referencia consultar la documentación propuesta por la Fundación para la confección de Protocolos de Interacción.

Protocolos: _ Predefinido X Definido por el usuario __Definida por metodología __No se sabe

Cooperación:

Cooperación: X_Cualquiera __ No se sabe

Organización:

Comentario: La organización se desarrolla a través del Modelo de Organización en el cual se describe la organización humana en la que el sistema Multi-Agente se introduce y la organización de agentes software.

Organización: __Predefinida X Definida por el usuario __Definida por metodología __No se sabe

Modelado de características adicionales

Movilidad:

Comentario: No se han elaborado ejemplos con la metodología donde se muestre la movilidad de agentes

Evaluación: __ Sí X No __No se considera

Documentación

Documentación disponible:

Explicación sobre el tipo de documentos: La documentación referente a la metodología es netamente funcional. Evaluación: __Buena _ Suficiente X Pobre

Casos de estudios presentados:

Evaluación: __Trivial X Parcial __Completo

Aspectos adicionales a evaluar

1. Proceso de desarrollo

Plataforma de desarrollo:

Comentario: La metodología trabaja con la plataforma MAST (Multi Agent System Tool). La misma muestra las funcionalidades básicas de comunicación entre los agentes.

Completitud de los flujos de trabajo:

Comentario: MAS-CommonKADS no refleja las etapas de Prueba e implementación. De igual forma, sería provechoso incluir tratamiento al mantenimiento de los sistemas desarrollados.

Evaluación: __Bueno _ Regular X Mal

A que fase se le da más importancia:

Comentario: Los desarrolladores de la metodología no evidencian superioridad en alguna etapa respecto a las demás.

Evaluación: __Sí __No X No se sabe

A que flujo de trabajo se le da más importancia:

Comentario: No se evidencia jerarquía alguna entre los flujos de trabajo.

Evaluación: __Sí __No X No se sabe

El ciclo de vida permite llegar a la implementación:

Comentario: MAS-CommonKADs propone el desarrollo completo de un SMA.

Evaluación: X Sí __No __No se sabe

2. Vista de modelos

Lenguaje de modelado:

Lenguaje: ADL y CKRL

Claridad de la notación de los estereotipos:

Comentario: Esta evaluación se tiene en cuanta considerando experiencia en desarrollo con UML.

Evaluación: X Bueno __Regular __Mal

Facilidad de uso de los estereotipos:

Comentario: Esta evaluación se tiene en cuenta considerando experiencia en desarrollo con UML.

Evaluación: X Bueno __Regular __Mal

Entendimiento de los estereotipos:

Comentario: Esta evaluación se tiene en cuenta considerando experiencia en desarrollo con UML.

Evaluación: X Bueno __Regular __Mal

3. Agentes

Modelado de la inteligencia o conocimiento del agente:

Comentario: Para MAS-CommonKADS los agentes son inteligentes.

Evaluación: X Si No __Parcialmente __No se sabe

Tiene pasos claros el modelado de la inteligencia:

Evaluación: __Si No X Parcialmente __No se sabe

Completitud del modelado de la inteligencia:

Evaluación: __Bueno X Regular __ Mal

Facilidad del modelado de la interacción: ESTO DEPENDE DE LA ONTOLOGIA Y MODELO DE ROLES

Evaluación: Bueno _X_Regular __Mal

4. Modelado de características adicionales

Modelado del ambiente o entorno:

Evaluación: __Si __No X Parcialmente __No se sabe

5. Otros

Facilidad de entendimiento de la metodología:

Comentario: La metodología a modo general es de fácil entendimiento puesto que se emplean notaciones conocidas o similares a las conocidas.

Evaluación: -- Bueno X Regular __Mal

Aspecto al que la metodología le da más importancia:

Comentario: Como se expresó anteriormente, no existe una diferenciación relevante entre alguna etapa o aspecto en particular y el resto.

Aspecto más cómodo de la metodología:

Comentario: La identificación y descripción de los agentes y de los casos de uso . Esto se trata en la fase de conceptualización.

Aspecto más incomodo de la metodología:

Comentario: El desarrollo de los modelos, principalmente el modelo de la experiencia y el modelo de organización

Principales ventajas: Su principal fortaleza es el hecho de ser la primera en plantear la integración de un Sistema Multi-Agente con un modelo de ciclo de vida de software.

- Incorpora como lenguaje de modelado ADL y CKRL
- Posee una herramienta MAST la cual es de difícil alcance porque no está disponible de forma gratuita.
- Existe documentación en inglés que posibilita su aplicación.
- No se define con claridad cada etapa de la metodología.
- Cubre la gran mayoría de las etapas de desarrollo de un SMA.

Sustenta la movilidad de los Agentes identificados.

Principales desventajas:

- Comentario: Esta metodología tiene algunas limitaciones, por ejemplo: no ofrece ningún entorno integrado para para editar y procesar sus 7 modelos.
- No se han realizado ejemplos de la metodología donde se manifieste la movilidad entre agentes.
- No se aborda como combinar un sistema de aprendizaje con el diseño de un agente.
- En el modelo de coordinación se han mostrado deficiencias con la notación gráfica de los Diagramas de Secuencia de Mensajes (MSCs) para tratar con grupos de agentes.

Conclusiones

Después de una extensa y profunda comparación realizada durante este capítulo, entre ambas metodologías, se ha llegado a la conclusión que MaSE es más completa y eficiente que MAS-CommonKADS teniendo en cuenta, fundamentalmente, los aspectos que se muestran a continuación en la tabla.

Campos	MAS-CommonKADS	MaSE
Etapas de Desarrollo	3 etapas que no cumplen los objetivos de manera eficiente	2 etapas que cumplen todos sus objetivos
Lenguaje de Especificación	Sí	Sí (más completo)
Herramienta de Soporte	Si (tiene pero ésta no está de forma gratuita en Internet)	Si (hay que obtener la licencia)
Documentación Disponible	Muy pobre	Regular
Implementación	No	Sí
Proceso de Desarrollo	Complejo	Fácil

Tabla 8: MaSE vs. MAS-CommanKADS

Conclusiones

Los objetivos que fueron trazados al inicio de la presente investigación se han cumplido cabalmente. Durante el desarrollo del trabajo se realizó una compilación de las principales publicaciones que abordan las metodologías MaSE y MAS-CommonKADS. Ambas fueron aplicadas al Caso de Estudio propuesto, arrojando resultados que permitieron llevar a cabo un eficiente desarrollo de estas dos metodologías. De igual forma se presentó un framework de evaluación de MaSE y MAS-CommonKADS sobre la base de las experiencias alcanzadas en sus respectivas aplicaciones al Caso de Estudio propuesto.

Todo esto conlleva a las siguientes conclusiones:

- MaSE es un proceso de especificación e implementación de SMA que presenta un ciclo de vida iterativo el cual es además independiente de dominio.
- MaSE presenta dos fases fundamentales que son el análisis y diseño, las cuales poseen un conjunto de etapas que generan artefactos en cada una de ellas.
- MaSE propone un tratamiento a los agentes que se corresponde al concepto que define este tipo de entidades adoptado por los desarrolladores de la metodología.
- MaSE presenta deficiencias en cuanto al tratamiento de algunas etapas de la metodología, tal es el caso de la etapa de prueba a la cual no se hace alusión en la documentación, dejándose de esta forma a creatividad de los desarrolladores, sucede llo mismo con la etapa de mantenimiento y con la captura de requisitos iniciales del sistema

Entre una de las principales debilidades de MaSE se le atribuye la no utilización de los conceptos BDI (Believe Desire Intention).

MAS-CommonKADS es una extensión de CommonKADS, siendo una metodología orientada a objetos y con gran experiencia.

Desarrolla 7 modelos en los que no se observa una forma razonable en la obtención de los mismos y el proceso no parece funcionar de forma iterativa, por lo que no resulta lógico.

Se puede destacar que MAS-CommonKADS presenta varias limitaciones, por ejemplo en cuanto a su herramienta de desarrollo: la metodología no presenta un entorno integrado para editar y procesar los modelos [1], para lo cual se propone el desarrollo de una herramienta que guíe en la aplicación de la metodología. También se necesita trabajar la movilidad de los agentes así como deficiencias en la notación gráfica de los Diagramas de Secuencia de Mensajes para tratar grupos de agentes.

Recomendaciones

Se recomienda a todos los interesados en esta rama de la Inteligencia Artificial que pretendan adentrarse en el mundo de los Sistemas Multi-Agentes y deseen implementar dichos sistemas, el estudio y análisis minucioso de este trabajo, el cual presenta una descripción detallada y muestra el funcionamiento de dos metodologías que se emplean en la creación de Sistemas Multi Agentes.

Se recomienda la incorporación en MaSE de algún mecanismo para la captura de requerimientos ya que la misma carece de un proceso para ser realizada. También es recomendable llevar a cabo la utilización de arquitecturas BDI (Believe Desire Intention) dentro de esta metodología.

Se propone además llevar a cabo la realización de proyectos en dominios de aplicación diferentes al propuesto en este trabajo, e intentar extender las metodologías a agentes con características diferentes a los modelados.

Referencias bibliográficas

Libros

[1]. Wooldridge M., N. J., D. Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. International Journal of Autonomous Agents and Multi-Agent Systems. 2000.

[2]. Julián V., B. V. Agentes Inteligentes: el siguiente paso en la inteligencia artificial. Especial 25 Aniversario. 2000.

Tesis

[3]. Moreno.
Estudio de Metodologías Orientadas a Agentes. Ciudad de la Habana, 2005.

[4]. Pinto, R. P.
Evaluación de la Metodología PASSI en un Caso de Estudio. Ciudad de la Habana, CUJAE, 2006.

[5]. Carlos A. Iglesias, M. G., José C. González, Juan R. Velasco.
Analysis and Design of Multiagent Systems using MAS-CommonKADS Madrid, Universidad Plitécnica de Madrid, 1998.

[6]. DeLoach, S. A.
Analysis and Design using MaSE and agentTool, Miami University, Oxford, Ohio, 2001.

[7]. Gómez Sanz Jorge J., R. F.
The INGENIAS Methodology. Fourth Iberoamerican Workshop on Multi-Agent Systems Iberagents, 2002.

[8]. J.Rumbaugh.
OMT:The Develoment Model. Journal of Object Oriented Programming, 1995.

[9]. B. Regnell, M. Andersson, and J. Bergstrand.
A hierachical use case model with graphical representation.1996

[10]. J. Huang, N. R. Jennings, and J. Fox.
Agent-based approach to health care management. Applied Artificial Ingelligence, 9(4):401–420.1995.

Índice de figuras, ecuaciones y tablas

Índice de figuras

Índice de tablas

Glosario de Abreviaturas

ADL: Lenguaje de definición de agentes (Agent Definition Language).

AM: Agent Model

BD: Base de Datos.

BDI: Believe Desire Intention.

CKRL: Lenguaje común de representación del conocimiento (Common Knowledge Representation Language).

CM: Communication Model.

CML: Conceptual Modelling Language.

CoM: Coordination Model.

CommonKADS: También llamado KADS-II. Continuación de la metodología.

DM: Design Model

EM: Experience Model

HMSC: Diagrama de secuencia de mensajes de alto nivel (High-Level Message Sequence Chart)

IA: Inteligencia Artificial.

IAD: Inteligencia Artificial Distribuida.

IADz: Inteligencia Artificial Descentralizada

ISOA: Ingeniería del software Orientada a Agentes.

JDK: Java Development Kit.

KADS: definiendo un nuevo conjunto de modelos.

MAS-CommonKADS: CommonKADS para Sistemas Multi-Agentes. Versión de CommonKADS propuesta en esta tesis para modelar Sistemas Multi-Agente.

MaSE: Multi Agent System Engineering.

MAST: Multiagent System Tool

MSC: Diagrama de secuencia de mensajes (Message Sequence Chart).

OCL: Object Constraint Language

OM: Object Model

SDL: Lenguaje de Descripción y Especificación (Specification and Description Language).

SMA: Sistemas Multi-Agentes

SPD: Solución de Problemas Distribuidos.

TM: Task Model.

UCI: Universidad de las Ciencias Informáticas

UML: Lenguaje de Modelado Unificado (Unified Modelling Language).

Glosario de Términos

***H*:**

Heurística: Una heurística es un algoritmo que ofrece como objetivos fundamentales buenos tiempos de ejecución y buenas soluciones, usualmente las óptimas. Por ejemplo: normalmente encuentran buenas soluciones, aunque en ocasiones no hay pruebas de que la solución no pueda ser arbitrariamente errónea; o se ejecuta razonablemente rápido, aunque no existe tampoco prueba de que deba ser así.

*****