RISI 3 (4) (2006)
Revista de Investigación de Sistemas e Informática
Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos
ISSN: 1815-0268 (versión impresa) / ISSN: 1816-3823 (versión electrónica)

Estudio de Metodologías para Desarrollo de Sistemas Multi-Agente

Marcos Rivas Peña, Giovana Valverde Ayala [1]

Resumen

Considerando que en el marco de la inteligencia artificial existen muchos problemas complejos con características distribuidas, los Sistemas Multi-Agente se presentan como posibles soluciones a estos problemas. Al momento de desarrollar Sistemas Multiagente como en el desarrollo de cualquier tipo de software necesitamos usar metodologías que nos permitan obtener el producto final. Numerosas metodologías se han propuesto para el desarrollo de Sistemas Multi-Agente pero todavía no ha alcanzado la madurez adecuada para convertirlos en estándar. Ante la dificultad de revisar todas se ha seleccionado cinco metodologías para estudiar en base a aquellas que tienen mayor documentación y mayor aceptación en la comunidad de agentes.

Palabras clave: metodologías, agentes, Sistemas Multi-Agente, ingeniería de software.

Abstract

Considering the mark of the artificial Intelligence exists many complex problems with distributed characteristics, the Multi agent Systems is presented as possible solutions to these problems. To the moment to develop Multi agent Systems like in the development of any software type we need to use methodologies that allow us to obtain the final product. Numerous methodologies have intended for the development of Multi agent Systems but it has not still reached the appropriate maturity to convert them in standard. Before the difficulty of revising all has been selected five methodologies to study based on those that have bigger documentation and bigger acceptance in the community of agents.

Key words: Methodologies, Agents, Multi-Agent Systems, Engineering of Software

1 Docentes del Departamento Académico de Ciencias de la Computación
E-mail: mrivasp@unmsm.edu.pe, gmvalverde@yahoo.com

1. Introducción

El avance tecnológico en las comunicaciones lleva al planteamiento de nuevos escenarios en los que es necesario compartir y coordinar, por consiguiente es necesario nuevas metodologías, técnicas y entornos de soporte informático para desarrollo de sistemas que incluyan los aspectos de coordinación y distribución.

La Inteligencia Artificial, campo de la informática, no ha sido impasible a este avance y a finales de los años setenta aparecen los primeros trabajos en Inteligencia Artificial Distribuida (aunque la primera reunión temática fue en 1980). Su objetivo es el estudio de modelos y técnicas para resolución de problemas en los que la distribución, sea física o funcional, sea inherente. A principios de los años ochenta los sistemas para resolución distribuida de problemas se caracterizaban por una forma de actuación concurrente en los diferentes nodos de una red, en general con control centralizado, de forma que los diferentes componentes son impasibles ante las actuaciones del resto de los componentes de la red.

Al principio de los años noventa aparecen Sistemas Multi-Agente (SMA) con control descentralizado y con módulos reusables. Los agentes de un SMA se conciben como independientes de un problema en concreto y se dota al sistema de protocolos de comunicación suficientemente genéricos.

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

En los últimos años hay un mercado interesado en la tecnología orientado a agentes [9], los investigadores le han prestado a esta áreas una atención especial, la Ingeniería de Software orientado a agente ha especificado algunas áreas de interés [18], y un interés especial en el desarrollo de Sistemas Multi-Agente (SMA), [16], por otro lado la industria y / o empresa están interesándose en usar esta tecnología para desarrollar sus propios productos, pero su aceptación se llevará a cabo en la medida que se desarrollen las herramientas necesarias que soporten todo el ciclo de vida del procesos de desarrollo de software.

Un SMA es un conjunto de agentes autónomos que trabajan cooperativamente para alcanzar sus objetivos, cada agente puede interactuar con su ambiente o con otro agente usando un lenguaje de alto nivel de comunicación.

Para el desarrollo de SMA se necesitan tecnologías que provienen de diferentes áreas de conocimiento, técnicas de ingeniería de software para soportar el proceso de desarrollo, técnicas de inteligencia artificial para desarrollar programas con capacidad de tomar decisiones en situaciones no previstas, y programación concurrente y distribuida para administrar las tareas que se ejecutan en diferentes máquinas.

Las técnicas convencionales de ingeniería de software no contemplan las necesidades que exige el desarrollo de SMA, como por ejemplo un lenguaje de comunicación orientado a agentes, por eso en los últimos años se han planteado diferentes metodologías para el desarrollo de éstos sistemas que cumplen el rol de guiar en todo el ciclo de vida de desarrollo de aplicaciones basada en agentes.

En este artículo realizaremos un análisis de algunas metodologías orientadas a agentes identificando sus bondades y deficiencias para el desarrollo de aplicaciones empresariales.

El resto del artículo tiene la siguiente organización: en el punto 2 se presenta el estado del arte en metodologías orientada a desarrollo de Sistemas Multi-Agente. En el punto 3 se presentan los distintos trabajos relacionados con el estudios de las metodologías. En el punto 4 se realiza un análisis de algunas de éstas metodologías elegidas considerando ciertos criterios. Finalmente, en el punto 5 se presentan las conclusiones.

2. Estado del Arte

Considerando los resultados de investigaciones realizadas en agentes y Sistemas Multi-Agente se han desarrollados diferentes metodologías que guían el desarrollo de SMA, una parte de éstas fueron ordenadas en una genealogía [11].

Los investigadores que han desarrollado las diferentes metodologías han seguido un enfoque de extensión de metodologías existentes para el desarrollo tradicional de software incluyendo aspectos relevantes de los agentes. Estas extensiones se han realizado principalmente desde tres áreas: Orientado a Objetos (OO), Ingeniería de Conocimiento (KE) y Ingeniería de Requisitos (RE). Una nueva revisión de las propuestas actuales se muestra en la Figura 1 con una genealogía actualizada

[26], donde P denota una nueva categoría de fuente de metodologías.

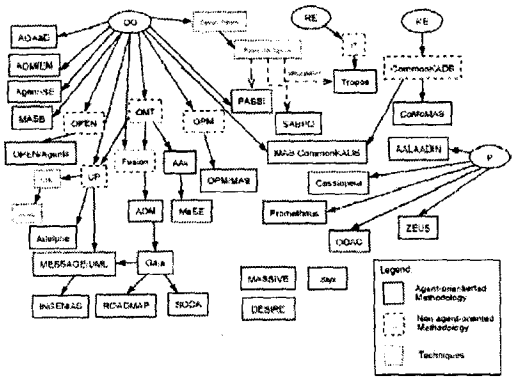

Figura 1: Genealogía de Metodologías Propuestas

Como se puede apreciar muchas metodologías para ingeniería de software orientado a agente se han propuesto pero uno de los principales obstáculos encontrado es la falta de metodologías maduras para de desarrollo de SMA [17], es decir pocas han logrado una madurez que permita su uso real, principalmente debido a la falta de una descripción detallada de éstas.

Según [4], las metodologías que se han descrito con mayor detalle y que han recibido mayor aceptación en la comunidad de agentes son: Gaia [31], MESSAGE [5], [6], MaSE [29], Prometheus [19], y Tropos [23], [24], realizaremos una breve descripción del propósito de cada una de las metodologías mencionadas.

GAIA

La metodología GAIA se centra en el concepto de construcción de un Sistema Multi-Agente como una organización computacional con diferente roles que interactúan entre ellos, pretende guiar al analista desde la declaración de requisitos iniciales a un modelo de diseño lo suficientemente detallado para ser implementado directamente.
Los principales conceptos que aparecen en la metodología se dividen en dos: abstractos y concretos, las entidades abstractas son aquellas que son empleadas durante el análisis para la conceptualización del sistema y las entidades concretas son empleadas en el proceso de diseño.

En GAIA el objetivo del análisis es conseguir comprender el sistema y su estructura sin referenciar ningún aspecto de implementación a través de la idea de organización, entendiendo una organización como una colección de roles, los cuales mantienen ciertas relaciones con otros. Los roles agrupan cuatro aspectos: responsabilidades del agente, los recursos que se le permite utilizar, las tareas asociadas e interacciones.
En la fase de análisis se usan dos modelos, el modelo de roles para identificar los roles clave en el sistema junto con sus propiedades definitorias y el modelo de interacciones que define las interacciones mediante una referencia a un modelo institucionalizado de intercambio de mensajes. Se debe destacar que para explicar la funcionalidad de los roles usa formulas lógicas para conseguir mayor precisión.

El objetivo de la fase de diseño es generar tres modelos: el modelo de agentes que define los tipos de agentes que existen, cuántas instancias de cada tipo y qué papel juega cada agente, el modelo de servicio que identifica los servicios (funciones del agente) asociados a cada rol, y un Modelo de conocidos, que define los enlaces de comunicaciones que existen entre los agentes.

El dominio de aplicación es principalmente en un ambiente donde los agentes son heterogéneos y estos podrían ser implementados usando diferentes lenguajes de programación, arquitectura y técnicas. La metodología no asume una arquitectura definida ni una plataforma especifica para su desarrollo y ejecución.

MESSAGE (Methodology for Engineering Systems of Software Agentes)

La metodología MESSAGE extiende el metamodelo de The Unified Modelling Language (UML) [14] adicionando nuevos metaconceptos como agente, objetivos y tareas, con la finalidad de proveer a los desarrolladores un alto grado de expresión. Una notación apropiada es también definida para representar gráficamente instancias de los metas conceptos. Además proporciona un proceso y ciertas reglas heurísticas que guían al diseñador como llevar las fases de análisis y diseño de un sistema en desarrollo.

MESSAGE realiza inicialmente una definición de conceptos presentando una notación basada en UML siempre que es posible. Los diagramas son extensiones sobre diagramas de clase y de actividad e incorpora las extensiones AUML [12] para diagramas de interacción.

El propósito de la fase de análisis es producir un conjunto de modelos (Modelo de organización, Modelo de Objetivos / tareas, Modelo de agente, Modelo de Interacción y Modelo de Dominio) en los que estén de acuerdo el analista y el usuario, estos modelos se realizan por refinamientos, en un nivel 0

se define el sistema con respecto a su entorno y el sistema es visto como un conjunto de organizaciones que interactúan con recursos, actores u otras organizaciones, en un nivel 1 se define la estructura y conducta de entidades como organización, agentes, tareas, objetivos, Niveles adicionales pueden ser definidos para analizar aspectos específicos del sistema.

La fase de diseño no presentan un único proceso sino que plantean dos posibles aproximaciones, una primera aproximación donde el diseño es dirigido por la organización del SMA y la arquitectura de agente, y la segunda aproximación está orientada por una plataforma de agente concreta.

En MESSAGE la arquitectura de agente es según diseño, los autores emplean una plataforma FIPA [7] [8] y en concreto la implementación JADE [1], basada en un modelo de programación OO y empleando JAVA como lenguaje de programación, esta metodología fue diseñada para desarrollar aplicaciones basados en agente en dominio de telecomunicaciones.

MaSE (Multiagent Systems Engineering)

El enfoque principal de MaSE es guiar a un diseñador a través de todo el ciclo de desarrollo de software, desde una especificación inicial del sistema hasta una especificación de implementación de un SMA que se lleve a cabo. Es independiente de una plataforma, de una arquitectura de agente, de un lenguaje de programación o de sistema de intercambio de mensajes.

El análisis en MaSE consta de tres pasos: (1) capturar los objetivos, donde se identifica el objetivo global del sistema y se descompone en subobjetivos donde cada uno define qué debe hacerse para cumplir con el objetivo padre, éstos se estructuran en diagrama jerárquico de objetivos, (2) capturar los casos de uso, se debe producir un conjunto de casos de uso y diagrama de secuencias para ayudar a identificar un conjunto inicial de roles del sistema y (3) refinar roles, se debe transformar el diagrama jerárquico de objetivos y los diagramas de secuencias en roles y sus tareas asociadas.

El diseño consta de cuatro pasos: (1) crear clases de agentes, que produce un diagramas de clases de agentes, que enumeran los agentes del sistema, roles jugados e identifican conversaciones entre los mismos, (2) construir conversaciones donde una conversación define un protocolo de coordinación entre dos agente, se representa a través de diagramas de clases de comunicación (iniciador + respondedor), (3) ensamblar clases de agentes donde se realiza el diseño interno de cada agente definiendo la arquitectura del agente y (4) diseño del sistema usando diagramas de despliegue para mostrar número, tipo y localización de las instancias de agentes en el sistema.

Un sistema diseñado en MaSE podría implementarse de diferentes maneras desde el mismo diseño. La metodología tiene un dominio de aplicación independiente, por lo tanto su aplicación se da en ambiente donde los agentes son heterogéneos y además dispone de una herramienta de desarrollo llamada AgentTool [28].

PROMETHEUS

El objetivo en el desarrollo de Promethus es tener un proceso con entregables definidos que puedan enseñar a los desarrolladores en las industrias y a los estudiantes de pregrado a desarrollar SMA sin haber tenido experiencia en agentes.

La metodología soporta todas las fases del ciclo de desarrollo de software y se diferencia de las otras metodologías por su nivel de detalle que cubre todas las actividades requeridas par el desarrollo de SMA, su evolución se ha dado con experiencias académicas y aplicaciones en la industria apoyando al desarrollo de agentes BDI (Belief - Desire - Intention), pero las fases iniciales de la metodología puede aplicarse a cualquier tipo de SMA.

Promethus consiste de tres fases: (1) Especificación del Sistema, esta fase involucra dos actividades que consiste en determinar el ambiente, el objetivo y las funcionalidades del sistema, el ambiente es definido en términos de percepción de la información que proviene del medio ambiente y de las acciones de los agentes que afectan su medio ambiente, se define los objetivos y las funcionalidades para alcanzar estos objetivos, las funcionalidades son descritas mediante escenarios usando casos de uso. (2) Diseño de Arquitectura, esta fase involucra tres actividades, se debe definir los tipos de agentes, diseñar una estructura global del sistema y definir las interacciones entre agentes. Cada tipo de agente se describe usando un descriptor de agente que describe su ciclo de vida, sus funcionalidades, los datos que utiliza y los datos que produce, sus objetivos, los eventos a la que debe responder, sus acciones y las interacciones que realiza con otros tipos de agentes. La estructura del sistema es plasmado en un diagrama de vista general que provee a los diseñadores y a los que implementan el sistema una descripción de cómo el sistema funciona en forma global, mostrando los tipos de agentes, los enlaces de comunicación entre ellos, los datos, las restricciones del sistema y su ambiente. (3) Diseño Detallado, en esta fase se

detallan los aspectos internos de los agentes y cómo ejecutarán sus tareas dentro del sistema global.

Se centra en definir las capacidades, los eventos internos, los planes y en detallar la estructura de datos para cada tipo de agente que se ha identificado en los pasos anteriores. Las capacidades de los agentes son descritos mediante un descriptor de capacidades que contiene información tales como qué eventos serán generados y qué eventos serán recibidos. Existen otros descriptores de menor detalle como el descriptor de planes individuales, descriptor de eventos y descriptor de datos. Esta fase involucra también la construcción de diagramas de vista general de agentes que junto con el descriptor de capacidades presentan un buen nivel de detalle de los componentes de la arquitectura interna de los agentes y de sus interacciones.

Prometheus es soportado por dos herramientas, JACK Development Environment (JDE) [15] y Prometheus Desing Toll (PDT), pero ninguna soporta la fase de especificación.

TROPOS

La metodología ha sido proyectada a cubrir todos los flujos de trabajos del Proceso Unificado (PU) [13], la idea principal que se aplica en el proceso de análisis y modelado es el actor, así como sus objetivos y posibles dependencias con otros actores.

El proceso de desarrollo con Tropos consiste de cinco fases: (1) Requisitos Iniciales, el análisis consiste en identificar los Stakeholders y sus intenciones, estos Stakeholders son modelados como actores sociales en un diagrama de actores donde se representan como los actores dependen de otros para alcanzar sus objetivos, sus planes a desarrollar y los recursos que necesite. En un segundo diagrama de objetivos se muestra un análisis de objetivos y planes con interés a un actor específico quien tiene la responsabilidad de alcanzarlos. El análisis de objetivos y planes se realiza en base a técnicas de razonamiento propuesto por la metodología. (2) Requisitos Posteriores, esta fase involucra una extensión del modelo creado en la fase anterior, se incluye un nuevo actor que representa al sistema y varias dependencias con otros actores del ambiente, estas dependencias definen los requisitos funcionales y no funcionales del sistema a crearse. (3) Diseño de Arquitectura, en esta fase se define la arquitectura global del sistema en términos de subsistemas interconectados a través de datos y flujos de control. Los subsistemas son representados en el modelo como actores y los datos y controles son representados como dependencias. (4) Diseño Detallado, en esta fase de define las especificaciones de agentes a un nivel detallado, Tropos usa diagrama de actividades de UML para representar capacidades y planes en detalle. La interacción entre agente es representado por diagrama de interacción de agentes AUML. (5) Implementación, Tropos para la implementación de agentes escoge una arquitectura BDI, con plataforma específica JACK, y provee pautas y heurísticas para graficar conceptos tropos a conceptos BDI y concepto BDI a estructuras JACK.

Una lista completa de metodologías para desarrollo de SMA puede encontrarse en [27].

3. Trabajos Relacionados

Existe pocos trabajos relacionados con la evaluación de metodologías para desarrollo de SMA comparadas con las evaluaciones hechas para las metodologías orientadas a objeto donde se pueden encontrar bastantes referencias, los trabajos encontrados son los siguientes.

O'Malley y DeLoach [22], presenta un framework que guía a la organización en la decisión, de adoptar una Metodología de Ingeniería de Software orientada a agente o una orientada a objeto para un proyecto en particular, selecciona un conjunto de criterios que los agrupa en dos categoría (1) administración de problemas y (2) requerimientos del proyectos, estos criterios fueron validados con profesionales de Ingeniería de Software y la aplicación de los criterios los realiza mediante un proceso de toma de decisiones aplicando la técnica de análisis de decisiones multiobjetivo, la investigación evaluó una metodología orientada a objeto versus un metodología orientada a agente, cuyos resultados son presentados en [22].

Cernuzzi and Rossi [2], propone un análisis cualitativo seguido de un calculo cuantitativo.

Proponen la construcción de un árbol de atributos que se organiza de acuerdo a criterios propuestos que se evalúa en las metodologías asignandoles un peso a cada atributo que es representado por ramas del árbol, si existen atributos que se subdividen, estas serán las hojas, entonces se asignan pesos a las hojas y luego se realiza el calculo para las ramas o atributos, realizado el calculo para cada atributo es posible obtener el valor de la raíz, así se obtiene un valor para cada metodología evaluada y finalmente se podrá tomar una decisión sobre la metodología a escoger.

Los autores identificaron tres tipos de criterio. Los Atributos internos que caracterizan las estructuras interiores de los agentes, los Atributos de la Interacción que describen cómo las interacciones dentro del sistema pueden ser modelados.

Finalmente, los Otros Requisitos del Proceso que evalúan el diseño y el proceso de desarrollo, propuestos por las metodologías.

Cuesta, Gómez, Gonzáles y Rodríguez [3] presentan un framework para evaluar metodologías orientadas a desarrollo de SMA, los criterios utilizados han sido clasificados en cinco grupos dependiendo del áreas de interés. El primer grupo de evaluación de criterios es llamado Proceso de Desarrollo y incorpora aspectos generales de la metodología como lo relacionado con las fases de construcción del sistema. El segundo grupo es llamado Modelo de Vista donde trata de reflejar el concepto de la metodología y su representación. El tercer grupo llamado Grupo de Agentes esta dirigido a la característica individual de los agentes tomado en cuenta por la metodología que se esta evaluando. El cuarto grupo es un modelo de características adicionales, dirigido a las extensiones que las metodologías proponen para tratar con aspecto importante de los SMA como ontologías, movilidad entre otras. Finalmente tenemos el grupo llamado Documentación, dirigido a evaluar la documentación disponible y los casos de estudios presentados. En [3] se presentan evaluaciones con el framework propuesto de las metodologías de ingenierías de software orientadas a agentes (AOSE, Agent Oriented Software Enginieering) más relevantes.

Julían y Botti [30] presenta una visión generalizada de varios métodos de desarrollo de sistema multiagentes analizándolas fundamentalmente a tres niveles: a nivel conceptual, a nivel general del método y a nivel de las fases de desarrollo consideradas. De este estudio resulta que estas metodologías no se podrían aplicar en entorno de tiempo real debido a la imposibilidad de modelar determinados conceptos, por ello se plantea una extensión de la metodología MESSAGE para su aplicación en entorno de tiempo real.

Gómez Sanz [10] realiza un estudio de siete metodologías que fueron seleccionadas con los siguientes criterios (1) Utilización de diferentes vistas para la especificación del sistema, (2) Incorporación de la ideas de procesos de desarrollo, (3) Integración de técnicas de ingeniería y teoría de agentes.

Dam and Winikof [4], presentan una evaluación elaborada en función de las información obtenida de los creadores de las metodologías que seleccionaron para el estudio y de los diseñadores que han realizado casos de estudios. Los criterios encontrados producto de una investigación fueron clasificados en cuatro grupos : Conceptos , Modelado Procesamiento y Pragmatismo. El grupo de conceptos examina que conceptos específicos soporta la metodología, el grupo de Proceso verifica las fases del ciclo de desarrollo que cubre la metodología, los grupos de Modelado y Pragmatismo son orientado a evaluar las características técnicas de las metodologías.

Jan Sudeikat, Lars Braubach, Alexander Pokahr, and Winfried Lamersdorf [24], presentan un framework para evaluar metodologías incluyendo criterios de plataforma de implantación de SMA. Los grupos de criterios considerados son: Conceptos, Notación, Procesos y Pragmatismo. Entre otro criterios principalmente, los Conceptos evalúa cuales son soportado por las metodologías, la Notación evalúa las vistas de los aspectos más importantes en el desarrollo de sistemas, el Proceso evalúa si las metodologías cubren todos los flujos de trabajos considerados por el Proceso Unificado,[13], y el Pragmatismo esta destinado a evaluar las herramientas case disponible que soportan a la metodologías.

4. Evaluación de Metodologías

Para evaluar las metodologías se ha considerado criterios que están relacionados con su aplicación a construir SMA para solucionar problemas empresariales, así es necesario primero considerar la parte conceptual que debe ir de la mano con los conceptos que se consideran en un SMA y que es limitante para poder aplicar metodología orientada a objetos, un segundo criterio es la cobertura que se realiza a nivel de los flujos de trabajos en el ciclo de vida del proceso de desarrollo de software teniendo en cuenta el PU, un tercer criterio es el uso de notación UML, esto debido a que hoy los desarrolladores están familiarizados con esta herramienta, un cuarto criterio es el tipo de arquitectura que se usa debido a que los problemas no podrían estar enmarcados a una específica y, un quinto criterio, es lo referente a las herramientas que soportan el proceso de desarrollo y su implementación.

A nivel conceptual la mayoría de las metodologías usan términos similares, tales como objetivo, tarea, agente, rol, interacción, que constituyen aspectos fundamentales en un SMA. En cuanto a la cobertura de los flujo de trabajos del PU, se observa que Gaia solo cubre la etapa de requerimiento, análisis y parcialmente la de diseño y no usa notación UML, MESSAGE, MaSE, Prometheus y Tropos cubren casi la totalidad de los flujos de trabajos. MESSAGE MaSE y Tropos usan notación UML y para modelar las interacciones Tropos y Prometheus usan notación AUML [20] [21].

En cuanto al tipo de arquitectura para la implantación de SMA, en este caso las metodologías Gaia y MaSE optan por cierta independencia de arquitectura dando la posibilidad de escoger entre varias opciones, MESSAGE usa una arquitectura según el diseño, Tropos y Prometheus usan una arquitectura específica BDI.

Considerando las herramientas de soporte y la plataforma de ejecución, para Gaia no se ha encontrado una herramienta que la soporte en el proceso de desarrollo, pero su implementación se podría realizar en JADE [1]. MESSAGE recomienda uso de algunas herramientas como Rational Rose para UML, como se desarrolla el diseño sobre una arquitectura genérica se puede orientar el diseño sobre una arquitectura específica siguiendo la propuesta de la plataforma JADE. MaSE incorpora una herramienta de desarrollo AgenTool [28] para facilitar el proceso de desarrollo. Tropos para la implementación, usa la plataforma JACK, que considera una arquitectura específica BDI y Prometheus es soportado por dos herramientas JACK y PDT.

5. Conclusiones

En este artículo se presenta un estudio de las metodologías con más documentación desarrolladas y poniendo énfasis en los criterios seleccionados que son necesarios para aplicarlos en desarrollo de SMA empresariales donde los desarrolladores no tienen el carácter de investigador sino de constructor de software para soportar las administración empresarial.

Las metodologías estudiadas provienen de diferentes líneas de investigación y han sido desarrolladas en base a información proporcionada por desarrolladores de otras metodologías.

La aplicación de cualquiera de estas metodologías requiere una formación en el paradigma de agente, además en la actualidad no existen ejemplos bien detallados de aplicaciones realizadas.

La elección de una metodología podría ser en función a cómo vienen trabajando los sistemas, si lo hace desde el punto de vista orientado a objeto entonces trabaja con sistemas basados en conocimientos, entonces la elección podría ser Gaia, MaSE o MESSAGE, sin embargo, si se desea una arquitectura especifica, por ejemplo BDI, entonces debería usar Tropos o Prometheus.

Una desventaja de las metodologías es que no se pueden aplicar en entorno de tiempo real dado a que su carácter de uso está orientado a sistemas cerrados no dinámicos.

Aquellos desarrolladores que opten por la metodología Gaia deberán ser mas especializados para comprender y definir los predicados que se emplean para describir las funcionalidades de los roles identificados.

Podemos concluir que en el camino por alcanzar una madurez en las metodologías de desarrollo de SMA se irán desarrollando nuevas notaciones y herramientas, pero se debe tener en cuenta que estos dos elementos juegan un papel importante en el desarrollo de software como se puede apreciar cuando se ha usado el paradigma estructurado y el orientado a objeto.

Referencias

[1]. Bellifemine, F., Caire, G., Trucco, T., Rimassa, G.: Jade Programmer's Guide. JADE 3.3 http://sharon.cselt.it/projects/jade/

[2]. Cernuzzi L. und Rossi G. "On the evaluation of agent oriented modeling methods", In Proc. of Agent Oriented Methodology Workshop, Seattle, 2002.

[3]. Cuesta P., Gómez A., Gonzáles J., and Rodríguez F. "A Framework for Evaluation of Agente Oriented Methodologies, Research supported by the spanish national project TIC 2002-04516-C03.01, 2003

[4]. Dam K. H. and Winikoff M. "Comparing Agent-Oriented Methodologies", In Proc. of the Fifth Int. Bi-Conference Workshop on Agent-Oriented Information Systems (at AAMAS03), 2003.

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

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

[7]. Foundation for Intelligent Physical Agents. FIPA Abstract Architecture Specification, SC00001L, 2002. http://www.fipa.org/specs/fipa00001/

[8]. Foundation for Intelligent Physical Agents. FIPA ACL Message Structure Specification, SC00061G, 2002. http://www.fipa.org/specs/fipa00061/

[9]. Green, Brenda, et al, Software Agents : a Review, May 1997, on-line at http://www.cs.tcd.ie/Brenda.Nangle/iag.html

[10]. Gómez J. "Metodologías para el desarrollo de sistemas multia-gente", Revista

Iberoamericana de Inteligencia Artificial Nro. 18, (2003). Pp 51-63.

[11]. HendersonSellers B. and Gorton I. "Agent-based Software Development Methodologies", White Paper, Summary of Workshop at the OOPSLA 2002, 2003.

[12]. J. Odell, H. Parunak, and B. Bauer. Extending UML for agents. In Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence., 2000.

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

[14]. Jacobson, I., Booch, G., Rumbaugh, J., 1999. "The Unified Modelling Language User Guide", Addison Wesley Longman.

[15]. JACK http://www.agent-software.com

[16]. K. P. Sycara. Multiagent systems. The AI Magazine, 19(2):79-92, 1998.

[17]. Luck M., McBurney P. und Preist C. Agent Technology: Enabling Next Generation Computing: A roadmap for agentbased computing. AgentLink report, ISBN 0854 327886, 2003.

[18]. Luck M., McBurney P. und Preist C. "A manifesto for Agent Technology: Towards Next Generation Computing", Autonomous Agents and Multi-Agent Systems, 9, 203 252, 2004.

[19]. Lin Padgham and Michael Winikof. Prometheus: A pragmatic methodology for engineering intelligent agents. In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, pages 97-108, Seattle, November 2002.

[20]. Odell, J., Parunak, H., and Bauer, B. (2000a). Extending UML for agents. In Proceedings of the Agent-Oriented Information Systems Workshop, pages 317.

[21]. Odell, J., Parunak, H., and Bauer, B. (2000b). Representing agent interaction protocols in UML. In Proceedings of the AGENTS'2000, Barcelona, Spain. Software Engineering, volume 1957. LNAI, Springer-Verlag.

[22]. O'Malley S. A. and DeLoach S. A. "Determining When to Use an Agent-Oriented Software Engineering Paradigm", In Proc. of the AOSE-2001, 2001.

[23]. Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, and Anna Perini. Tropos: An agent-oriented software development methodology. Technical Report DIT-02-0015, University of Trento, Department of Information and Communication Technology, 2002.

[24]. Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, and Anna Perini. "Tropos: An Agent-Oriented Software Development Methodology", Autonomous Agents and Multi-Agent Sytems, 8, 203 236, 2004.

[25]. Lin Padgham and Michael Winikof. Prometheus: A methodology for developing intelligent agents. In Third International Workshop on Agent-Oriented Software Engineering, July 2002.

[26]. Sudeika J., Braubach L., Pokahr A. and Lamersdorf W. "Evaluation of agent-Oriented Software Methodologies Examination of the Gap Between Modeling and Platform". Agent-Oriented Software Engineering V, Fifth International Workshop AOSE 2004. Springer Verlag, 126-141, 7/2004.

[27]. Sudeikat J. "Betrachtung und Auswahl der Methoden zur Entwicklung von Agentensystemen", diploma thesis, in German, HAW Hamburg, 2004.

[28]. Scott A. DeLoach. Analysis and design using MaSE and agentTool. In Proceedings of the 12th Midwest Arti_cial Intelligence and Cognitive Science Conference (MAICS 2001), 2001.

[29]. Scott A. DeLoach, Mark F. Wood and Clint H. Sparkman, Multiagent Systems Engineering, The International Journal of Software Engineering and Knowledge Engineering, Volume 11 no. 3, June 2001.

[30]. Vicente J. Julián and Vicente J. Botti, "Estudios de métodos de desarrollo de sistemas multiagente", Revista Iberoamericana de Inteligencia artificial Nro. 18, 2003, pp. 65-80.

Wooldridge M. J., Jennings N. R. und Kinny D. "The Gaia methodology for agent-oriented analysis and design". Autonomous Agents and Multi-Agent Systems, 3(3):285312, 2000.