INFORMÁTICA

INGENIERÍA DE SOFTWARE ORIENTADA A AGENTES: ROLES Y METODOLOGÍAS

Mailyn Moreno Espino, *Ingeniera Informática, Instructora, Centro de Estudios de Ingeniería de Sistemas (CEIS).*
e-mail:my@ceis.cujae.edu.cu

Alejandro Rosete Suárez, *Ingeniero en Sistemas Automatizados de Dirección, Doctor en Ciencias Técnicas, Profesor Auxiliar, CEIS.*
e-mail:rosete@ceis.cujae.edu.cu

Alfredo Simón Cuevas, *Ingeniero Informático, Instructor, CEIS.*
e-mail:asimos@ceis.cujae.edu.cu

Reinier Valdés González, *Ingeniero Informática, Instructor CEIS.*
e-mail:rualdes@ceis.cujae.edu.cu

Exiquio Leyva Pérez, *Ingeniero Químico, Doctor en Ciencias Técnicas, Profesor Titular, CEIS.*
e-mail:exiquio@ceis.cujae.edu.cu

Raisa Socorro*, Ingeniera Informática, Máster en Informática Aplicada, Asistente, CEIS.*
e-mail:raisa@ceis.cujae.edu.cu

Joaquín Pina Amargós, *Ingeniero Informático, Doctor en Ciencias Técnicas, Profesor Auxiliar, CEIS.*
e-mail:jpina@ceis.cujae.edu.cu

Alexander García Fernández, *Ingeniero Informático, Instructor, CEIS, Instituto Superior Politécnico José Antonio Echeverría, Cujae, Ciudad de La Habana, Cuba*
e-mail:agarciaf@ceis.cujae.edu.cu

Recibido:mayo del 2006
Aprobado:julio del 2006

Resumen / ***Abstract***

Hoy los estándares de ingeniería de software, RUP como proceso de desarrollo y UML como lenguaje de modelado, enfrentan retos debido a los paradigmas distribuidos: Servicios Web y agentes. Los agentes, paradigma prometedor para la ingeniería de software, tienen un gran reto, la falta de estándares metodológicos para su desarrollo. En este trabajo se realiza un estudio comparativo de nueve metodologías de agentes, se analizan roles de personas en la ingeniería de software orientada a agentes y se proponen algunos roles no presentes en RUP y recomendaciones para otros sí presentes que parecen necesarios con este nuevo paradigma.

Software engineering standards, such as RUP as a development process and UML as a modeling language, face a big challenge now due to new distributed paradigms: Web Services and agents. Agent-orientation is a very promising paradigm for software engineering, but they have the big challenges, is the absence of one standard methodology to use. In this work we present a comparative study of nine agent-oriented methodologies. Also, the roles of persons for agent-oriented software engineering are analyzed. New roles of persons are proposed in order to deliver agent-oriented products and some recommendations for using the roles of RUP in agent process are also presented.

Palabras clave / ***Key words***

Ingeniería de software, agentes, metodologías, roles

Software engineering, agents, methodology, roles

INTRODUCCIÓN

La ingeniería de software ha transitado por diferentes etapas, marcadas por los paradigmas de programación, por ejemplo, el enfoque estructurado y el enfoque orientado a objetos (OO). Este último tiene gran aceptación por las facilidades que brinda y por la estandarización ganada con el lenguaje de modelado UML y el proceso unificado de desarrollo (RUP).[1] Hoy existen grandes retos para estos estándares, provocados por las nuevas tecnologías distribuidas entre los que están los servicios Web y los agentes. Los agentes son entidades computacionales que se diferencian de los objetos en varios aspectos entre los que se destacan, la autonomía, la proactividad y el carácter social. Los agentes encapsulan además de los datos y los métodos, los mecanismos de activación de los métodos y su control interno.

Con esta nueva tecnología se han desarrollado aplicaciones exitosas que han influido en su aceptación. Sin embargo, un reto para su desarrollo es la no existencia de estándares metodológicos. Las metodologías orientadas a objetos (OO) no logran cubrir las necesidades de desarrollo de estos nuevos sistemas,[2] debido a que un objeto no es autónomo, ni proactivo.

Como vía para enfrentar las limitaciones de las metodologías OO se han hecho varias propuestas de metodologías orientadas a agentes: TROPOS,[3] MaSE,[4,5] MESSAGE,[6,7] Ingenias,[8] Zeus,[9]

Prometheus,[10, 11] GAIA,[12] MAS-CommonsKADS,[13], Vowel Engineering,[14] entre otras.

En este trabajo se comparan las nueve metodologías anteriormente mencionadas. Además, como la orientación a agentes introduce nuevos conceptos y tareas a desarrollar que no están incluidas en RUP, se proponen nuevos roles del equipo de proyecto para el desarrollo de sistemas multiagentes y nuevas habilidades en otros roles.

COMPARACIÓN DE METODOLOGÍAS

Para comparar las nueve metodologías anteriormente mencionadas se seleccionó un grupo de aspectos que se dividieron en tres partes: la primera que tiene que ver con las etapas del ciclo de vida que cubren; la segunda, relacionada con los antecedentes, lenguaje de modelado y las herramientas CASE, y la tercera parte relacionada con lo que se modela, es decir, si modelan el ambiente, la inteligencia y las relaciones entre los agentes. Los resultados se muestran en la tabla 1.

En la primera parte de la tabla, se presentan las etapas del ciclo de vida que cubren. La palabra **Sí** significa que sí cubre la etapa, **Sí-** que propone la etapa pero no la desarrolla y **No,** no la cubre. En la segunda parte de la tabla, referida al leguaje de modelado y antecedentes, tiene la nomenclatura siguiente. En el lenguaje de modelado se pone el nombre de lenguaje que utiliza. Si la metodología no propone un lenguaje determinado se pone un **0** que debe entenderse como neutral. Para el aspecto de herramienta case: **Sí** si tiene, y **No** lo contrario a esto. En el aspecto de antecedentes: **OO** se usa cuando tiene antecedentes en la orientación a objetos, **IR** en la ingeniería de requisitos, **IC** en la ingeniería del conocimiento. Se usa ***** (asterisco) cuando sus antecedentes son una mezclas de paradigmas y **No** para cuando no tiene antecedente tan directos de los paradigmas anteriormente planteados. En la tercera parte de la tabla que tiene que ver con los aspectos que modela cada metodología se utiliza como nomenclatura: **Sí**, que significa que tiene en cuenta el modelado de este aspecto, **No** para representar que no lo tiene en cuenta y **?** (signo de interrogación) cuando modela parcialmente este aspecto pero no del todo.

TABLA 1
Comparación de metodologías

Metodología / Aspectos	Tropos	GAIA	MaSE	MESSAGE/-UML	INGE-NIAS	Vowel Enginee-ring	MAS CommonK-ADS	ZEUS	Prome-theus
Etapas del ciclo de vida									
Requisitos	Sí	No	Sí	No	No	No	No	No	Sí
Análisis	Sí	Sí	Sí	Sí	Sí	No	Sí	Sí	Sí
Diseño	Sí	Sí	Sí	Sí	Sí	No	Sí	Sí	Sí
Implementación	Sí	No	Sí-	No	No	No	Sí-	Sí-	No
Prueba	No	No	No	No	No	No	No	No	No
Leguajes de modelado, herramienta CASE y antecedentes (origen)									
Lenguaje de modelado	UML/-AUML	0	UML + OCL	UML/AUML	AUML	0	UML	UML	UML/-AUML
CASE	No	No	Sí	No	Sí	No	Sí	Sí	Sí
Origen	IR	OO*	OO	OO	OO	No	IC	*	*
Modelado de aspectos importantes									
Ambiente	No	Sí	Sí	No	Sí	Sí	No	No	Sí
Inteligencia	Sí	Sí	?	?	?	Sí	Sí	?	Sí
Interacción	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí

Como se puede observar en la tabla 1, solo tres metodologías proponen cómo tratar la etapa de requisitos: TROPOS, MaSE y Prometheus. Las dos últimas incluyeron esta etapa en extensiones de ellas. La mayoría de las metodologías cubren las etapas de análisis y diseño, con la excepción de Vowel Engineering que no tiene definida ninguna etapa. El resto de las etapas no son consideradas en detalle. TROPOS sí incluye la implementación y MaSE, MAS-Common KADS y ZEUS la proponen pero no la desarrollan.[15] MAS-Common KADS propone pequeñas pinceladas del proceso de implantación.

En cuanto al lenguaje de modelado algunas utilizan UML, tal es el caso de MESSAGE, MAS-Common KADS y ZEUS. TROPOS, Prometheus y MaSE también emplean UML, pero las dos primeras utilizan AUML para complementar algunos de sus artefactos y la segunda emplea una combinación con OCL.[9] Los caso de GAIA y Vowel Engineering, no proponen el uso de un lenguaje de modelado determinado. No obstante, en la mayoría se incluyen diagramas propios. En cuanto a la existencia de herramientas de desarrollo asociadas debe decirse que no es un aspecto generalizado. MaSE incorpora la herramienta agentTool para facilitar el desarrollo, aunque no cubre todas las etapas. Prometheus también tiene herramientas pero tampoco abarcan todo el proceso, estas son JACK Development Environment y Prometheus Design Tool. ZEUS e INGENIAS, son otras que tienen herramienta de desarrollo, en primer caso se llama ZEUS y en el segundo IDE. En MAS-Common KADS existe una herramienta conocida como AgentEditor, mientras que en TROPOS se comenta que se está en fase de desarrollo de una herramienta que sustente todo el proceso. Finalmente MESSAGE propone una guía de recomendaciones de herramientas útiles para MESSAGE, básicamente se propone hacer uso de herramientas para UML como Rational Rose o la herramienta metacase MetaEdit+.

En cuanto a los orígenes de la propuesta, la mayoría toma elementos de la orientación a objetos (MESSAGE, MaSE e INGENIAS).

En el caso de GAIA los autores toman muchos elementos de la metodología FUSION, aunque hacen cambios radicales. En el caso de TROPOS toma elementos de la ingeniería de requisitos (método i*). MAS-Common KADS deviene de la ingeniería de conocimiento (metodología Common KADS). ZEUS y Prometheus vienen de una mezcla de varios de paradigmas.[16] Vowel Engineering no tiene una ascendencia clara.

En cuanto al modelado, es de destacar que todas las metodologías estudiadas modelan la interacción entre los agentes de una forma u otra. Todos modelan la inteligencia aunque hay casos en que lo asumen con una inteligencia limitada (MESSAGE, MaSE, ZEUS e INGENIAS). El aspecto que no todas modelan es el ambiente alrededor del sistema, lo cual es importante ya que sin esto se supone que los sistemas no son dinámicos y aparentemente serán cerrados. Dentro de las metodologías que sí modelan el ambiente están GAIA, MaSE, INGENIAS, Vowel Engineering y Prometheus.

ROLES EN PROCESO DE DESARROLLO DE UN SISTEMA MULTIAGENTES

Los roles en el proceso de desarrollo de software son importantes para la especialización y para mejorar la eficacia en la distribución de las tareas entre los miembro del equipo. Como se pudo observar anteriormente, las metodologías orientadas a agentes tienen varias tareas que no están presentes en RUP,[1,17] tal es el caso de modelar la inteligencia, modelar la interacción entre los agentes, entre otros. Esto trae como consecuencia que dentro de las etapas del proceso de desarrollo aparezcan nuevas tareas y esto a su vez nuevos roles dentro del equipo de proyecto o nuevas habilidades que deben tener los roles propuesto por RUP. Aquí se presentan algunas propuestas para ajustar los roles de las personas para el desarrollo de sistemas multiagentes. Se optó, siempre que fuera posible, por modificar las responsabilidades de los roles de RUP y adquirir nuevas tareas necesarias. Cuando esto no es posible (debido a la naturaleza diferente de las tareas en los roles de RUP y las nuevas necesidades) se proponen nuevos roles. Los roles de RUP considerados que no cambian mucho, no se discutirán. Se ha preferido usar el término **papeles** en este trabajo para llamar a lo que es llamado también **roles** en las metodologías de agentes, para evitar confusiones.

Analista de sistema: Este rol debe extender sus tareas para identificar los papeles preliminares y asociarlos con los casos del uso. Esta tarea incluye la identificación preliminar del grado de autonomía y la actitud reactiva/proactiva. También implica que el diagrama de caso de uso pueda modificarse de alguna manera para identificar claramente qué papel es responsable de cada caso del uso.

Arquitecto: Este rol en la etapa de análisis de requisitos puede adquirir la responsabilidad de poner la importancia relativa de cada papel, y proponer el orden de las iteraciones de desarrollo según esta decisión anterior. En la etapa de Análisis, puede adquirir la responsabilidad de definir la estructura organizacional o la comunidad del agente. Esta decisión está muy estrechamente relacionada con la decisión de organización de los subsistemas (paquetes, clase más importantes, etc.) pero también necesita una calificación adicional en dirección y patrones organizacionales humanos (como se sugiere en la referencia 12). Debe notarse que en la estructura organizacional de la comunidad de agentes influyen aspectos de diseño y también la organización real. Esta relación entre agentes y subsistemas está muy cerca del enfoque de Tropos.[3] Este rol también debe encargarse de asignar los papeles a los agentes.

Ingeniero de componentes: Este rol adquiere la tarea de especificación de los papeles, cuando el análisis se dirige al nivel más alto.

Ingeniero de la interacción: Este es un rol nuevo, no está en RUP. Aparece como consecuencia de la complejidad de la interacción entre los agentes. Este rol debe encargarse de definir la conversación, ontologías a usar, normas a ser respetadas, etc. La complejidad de esta tarea se explica en la referencia18.

Ingeniero de conocimiento: Este rol también es nuevo. Aparece como consecuencia de la complejidad del control interno

en cada agente y la necesidad de operar con flexibilidad. Es responsable de definir las habilidades de inferencia de cada agente e implementar el mecanismo de razonamiento acorde con el nivel de reactividad/proactividad definido. Debe cooperar estrechamente con el ingeniero de la interacción para definir las ontologías a ser usadas en el razonamiento, acorde con la comunicación.

Integrador de sistema: Puede desaparecer, ya que la naturaleza de su tarea es menos necesaria, por la flexible integración impuesta por la orientación a agentes. Podría usarse si la implementación interna de los agentes implica todavía complejos y distribuidos subsistemas que utilizan el paradigma orientado a objetos u otro (debido a la mezcla natural de paradigmas).

Ingeniero de prueba: Puede asumir la preparación de una fase de prueba más compleja, incluyendo una prueba posdespliegue (prueba tardía) que garantice que el sistema no se va de control.

Ingeniero de prueba de integración: Este papel puede adquirir la responsabilidad de hacer pruebas de la capacidad de colaboración de agentes y la prueba de cambios organizacionales.

Ingeniero de prueba de agentes: Este rol es nuevo. Aparece como consecuencia de la naturaleza más compleja de la conducta del agente. Es responsable de hacer pruebas de agentes por separado para probar la autonomía, reactividad y conducta proactiva.

CONCLUSIONES

La orientación a agente es muy reciente y debe revolucionar el desarrollo de software.[2] Dicha revolución ha generado un nuevo campo de investigación, como es la ingeniería de software orientada a agentes. Hay una gran cantidad de metodologías orientadas a agentes y no existe una estandarización en este sentido ni en los lenguajes de modelado, aunque es de destacar que recientemente OMG ha comenzado a dar pasos hacia una unificación.[19] Casi todas las metodologías proponen y desarrollan las etapas de análisis y diseño, y dejan las demás etapas sin definir una forma clara para su desarrollo.

Es importante destacar que ninguna de las metodologías estudiadas cubren la etapa de prueba, siendo esta una etapa muy importante en el ciclo de vida de desarrollo de cualquier software y en especial de los sistemas orientados a agente, debido a su complejidad, dinamismo y carácter distribuido.

Un aspecto relevante es que varias metodologías obvian la modelación del ambiente que rodea al sistema. En este trabajo se argumenta la necesidad de cambios en los roles de las personas dentro del desarrollo del software, con respecto a los roles de RUP que incluyen la introducción de nuevos roles (como el ingeniero de conocimiento y de interacción), y la modificación de las responsabilidades de otros (por ejemplo: el arquitecto y el ingeniero de pruebas).

REFERENCIAS

1. **JACOBSO, IVAR; GRADY BOOCH Y JAMES RUMBAUGH:** *El proceso unificado de desarrollo de software*, Pearson Educación, Madrid, 2000.
2. **ZAMBONELLI, F. AND A. OMICINI:** *Challenges and Research Directions in Agent-Oriented Software Engineering*, Autonomous Agents and Multi-Agent Systems, Vol. 9, 2004.
3. **BRESCIANI, P.;** *et al.*: *An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systemas*, Vol. 8, 2004.
4. **DELOACH, S.:** *Multiagent Systems Engineering of Organization-Based Multiagent Systems*, in 4th Intenacional Workshop on Software Engineering for Large-Scale Multi-Agents Systems SELMAS´05. St. Louis, 2005.
5. **DELOACH, S.; M. WOOD AND C. SPARKMAN:** "Multiagent Systems Engineering", *Internacional Journal of Engineering and Knowledge Engineering*, 11(3), 231-258, 2001.
6. **EVANS, R.** *et al.*: *MESSAGE: Methodology for Engineering Systems of Software Agents*, EURESCOM Participants in P907, 2001.
7. **CAIRE, G.,** *et al.*: *Agent Oriented Analysis Using MESSAGE/UML*, in Agent-Oriented Software Engineering AOSE´01, 2001.
8. **GÓMEZ, J. AND R. FUENTES:** *The INGENIAS Methodology*. in Fourth Iberoamerican Workshop on Multi-Agent Systems Iberagent, 2002.
9. **JULIÁN, V. Y V. BOTTI:** "Estudio de métodos de desarrollo de sistemas multiagente", *Revista Iberoamericana de Inteligencia Artificial*, Vol. 18, 2003.
10. **PADGHAM, L. AND M.WINIKOFF:** *Prometheus: A Methodology for Developing Inteligent Agents*, in Firts Internacional Joint Conference on Autonomous and Multi-Agent Systems AAMAS, Bologna, Italy, 2002.
11. **CYSNEIROS, G. AND A. ZISMAN:** *Refining Prometheus Methodology with i**, in 3rd International Workshop on Agent-Oriented Methodologies, OOPSLA 2004, Canada, 2004.
12. **ZAMBONELLI, F.; N. JENNINGS AND M.WOOLDRIDGE:** "Developing Multiagent System: The Gaia Methodology", *ACM Transactions on Software Engineering and Methodology*, Vol. 2, 2003.
13. **IGLESIAS, C.** *et al.*: Analysis and Design of Multiagent Systems Using MAS-CommonKADS, Intelligent Agents IV LNA1, 1365, 1998.
14. *MAGMA Research Group, Vowel Engineering*, 2005.
15. **DAM, K. H. AND M.WINIKOFF:** *Comparing Agent-Oriented Methodologies. en Agent-Oriented Information Systems AOIS´03*, 2003.
16. **SUDEIKAT, J.; L. BRAUBACH.; A. POKAHR AND W. LAMERSDORF:** "Evaluation of Agent-Oriented Software Methodologies- Examination of the Gap Betweem Modeling and Platform", in International Workshop on Agent-Oriented Software Engineering, AOSE´04. 2004.
17. **CRAIN, A.:** *Understanding RUP Roles*, Rational Software: The Rational Edge, April, 2005.
18. **PARUNAK, H. V. D.** *et al.*: *A preliminary taxonomy of Multi-Agent Interaction*, in Agent-Oriented Software Engineering (AOSE IV), Springer, 2004.
19. *Request for Information on Modeling Agent-Based Systems*. Analysis and Design Task Force, Object Management Group (OMG), August 25, 2004.