

## REQUISITOS Y PATRONES: RETOS PARA LA INGENIERÍA DE SOFTWARE ORIENTADA A AGENTES

### Resumen / Abstract

En este trabajo se hace una introducción al nuevo paradigma de la orientación a agentes, con especial énfasis en dos aspectos: la ingeniería de requisitos o requerimientos y los patrones. Se comentan brevemente algunas de las novedades que en estos dos aspectos introduce el nuevo paradigma de agentes, y cómo interactúan los procesos de desarrollo actuales con dicho paradigma

*In this work make an introduction to the new paradigm of the oriented to Agents, with special emphasis in two aspects: the engineering of requirements and the patterns. Some of the new features comment briefly that in these two aspects the new paradigm of agents introduces, and how the present development processes with this new paradigm interact.*

### Palabras clave / Key words

Agentes, ingeniería de software, requisitos, patrones

*Agents, software engineering, requirements, patterns*

**Yanely Lores de Feria**, Ingeniera Informática, Departamento de Multimedia y Web, Instituto de Información Científica y Tecnológica (IDICT), Ciudad de La Habana, Cuba  
e-mail: yanelys@idict.cu

**Elieser Norberto Fuentes Jorge**, Ingeniero Informático, Asistente, Grupo Empresarial Emprestur SA, Casa Matriz, Ciudad de La Habana, Cuba  
e-mail: elieser@emprestur.tur.cu

**Julio César Iglesias Bayeux**, Ingeniero Informático, Centro de Ingeniería Genética y Biotecnología (CIGB) Ciudad de La Habana, Cuba  
e-mail: jcesar@cigb.edu.cu

**Alejandro Rosete Suárez**, Ingeniero en Sistemas Automatizados de Dirección, Doctor en Ciencias Técnicas, Profesor Auxiliar, Centro de Estudios de Ingeniería de Sistemas (CEIS), Instituto Superior Politécnico José Antonio Echeverría, Ciudad de La Habana, Cuba  
e-mail: rosete@ceis.cujae.edu.cu

Recibido: mayo del 2006  
Aprobado: julio del 2006

## INTRODUCCIÓN

Con el objetivo de adecuarse a la complejidad creciente del desarrollo de sistemas de software, la ingeniería del software ha avanzado junto al desarrollo de mecanismos de abstracción cada vez más potentes y de más alto nivel. La abstracción modular, los tipos abstractos de datos, los objetos y clases,<sup>1,2</sup> los componentes y los servicios web,<sup>3-5</sup> son claros ejemplos de este tipo de abstracciones.

La orientación a **agentes**,<sup>6</sup> constituye un nuevo paradigma, donde se continúa creciendo en el nivel de abstracción, a la vez que se introduce con él una vía de poner los desarrollos en inteligencia artificial en una línea convergente con el desarrollo en la ingeniería de software. Para poder diseñar y desarrollar sistemas orientados a agentes, es necesario contar no solamente con modelos y tecnologías modernos, sino también con un conjunto de abstracciones de ingeniería de software, que sirvan de referencia en el proceso de desarrollo, y se establezcan como las bases para metodologías que permitan a los desarrolladores crear tales sistemas de una manera robusta y segura.

En este trabajo se analizan primero algunos aspectos generales del paradigma de la orientación a agentes, y luego se tratan con más detalle dos aspectos: La ingeniería de requisitos y los patrones, comentándose cómo estos aspectos cambian con el nuevo paradigma.

## INGENIERÍA DE SOFTWARE ORIENTADA A AGENTES

Aunque no hay total unificación en cuanto a qué es un agente, un intento de unificar los esfuerzos para el desarrollo de esta tecnología puede encontrarse en FIPA (Foundation for Intelligent

Physical Agents)<sup>7</sup> que los define como una entidad de software con un grupo de propiedades entre las que se destacan: el ser capaz de actuar autónomamente en un ambiente, comunicarse directamente con otros agentes, estar condicionado por un conjunto de tendencias u objetivos, manejar recursos propios, ser capaz de percibir su ambiente y tomar de él una representación parcial, ser una entidad que posee habilidad y ofrece servicios, etcétera.<sup>7</sup>

De forma general, varios autores reconocen en los agentes, diversas propiedades, entre las que se destacan el ser autónomos, reactivos, proactivos y tener habilidad social. Con ligeras modificaciones de enfoques, otros autores también reconocen estas propiedades.<sup>6-16</sup>

El modelo BDI (*Belief, Desire, Intention*): creencia, deseo, intención), propuesto por Rao y Goergeff, es el modelo teórico más extendido y conocido para modelar los agentes. Se entiende por creencia, la información disponible para un agente, la cual es tomada del ambiente y puede verse como su entrada o lo que conoce del mundo. Los deseos son las tareas ubicadas en el agente, mientras que las intenciones son los deseos que se ha propuesto o decidido lograr, deseos seleccionados o escogidos.<sup>15</sup> El modelo BDI ve a los agentes como sistemas intencionales, cuyo comportamiento puede predecirse y explicarse de manera similar a cómo se predice el comportamiento de los sistemas inteligentes complejos (es decir, las personas).<sup>15</sup>

Se le asigna a cada agente un conjunto de propiedades que varían su nombre según la metodología que se utilice como son TROPOS (agentes, capacidades, eventos, planes, etcétera),<sup>9,17</sup> GAIA (permisos, responsabilidades, actividades),<sup>14,18</sup> AAI (creencias, metas y planes, servicios, responsabilidades o roles), SODA (permisos, actividades, tareas, reglas de interacción), OperA (objetivos, normativas), etc.<sup>19</sup> Como puede verse, son muchas las metodologías que se han propuesto y aún está lejos el momento de tener una unificación o estandarización de las mismas.

Una de las propiedades más importantes en los agentes es su capacidad social. Los agentes sirven para abstraer en ellos elementos de alto nivel (subsistemas) que involucran las intenciones, similar a como las estructuras organizacionales estructuran una tarea. Por esa razón, puede ser importante desde un inicio, definir las reglas de la organización de los agentes. Existen metodologías que se centran en esto.<sup>14,18,19</sup>

En principio, los agentes independientes tienen una gran utilidad pero lo realmente interesante es el desarrollo de comunidades de agentes. La ventaja de los paradigmas multiagentes está asociada a su capacidad para modelar el desarrollo de sistemas en entornos complejos y distribuidos.

### Ingeniería de requisitos orientada a agentes

Hay dos cambios importantes que impactan en la ingeniería de requisitos o requerimientos a partir de las características propias del paradigma de los agentes:

- La orientación a agentes normalmente lleva a desarrollar un sistema multiagente, más que a un agente solitario.
- Los agentes son proactivos y no solo reactivos.

Estos cambios implican grandes retos para los métodos tradicionales de ingeniería de requisitos, ya que los mismos normalmente están enfocados a modelar un sistema que va a surgir en solitario como **una herramienta de cambio en una empresa**, pero que él en sí mismo es algo cerrado que encapsula toda la funcionalidad a implementar.

Cuando surge un sistema multiagente, esto implica un desacoplamiento mayor que lo que se puede lograr con una división por módulos o paquetes que está más enfocada a aspectos de implementación que a la función que cumplen dentro de la organización donde se va a hacer la automatización. Esto viene modelado a partir de las características de las metas que tengan cada uno de los agentes y la alineación de las mismas con la organización.

Por otra parte, al ser los agentes proactivos, no puede pensarse en un sistema inerte esperando por órdenes de los actores como se modela en los sistemas tradicionales partiendo de los casos de uso<sup>1,2,20</sup>, que es la herramienta esencial que guía el proceso. Para enfrentar estos retos, han surgido un grupo de métodos adaptados a las nuevas características como son los métodos de la **ingeniería de requisitos tempranos**, y especialmente los **métodos orientados a las metas** como es el caso de "*i\**".<sup>9,21-23</sup>

Actualmente existe consenso en que la especificación de los requerimientos debería incluir alguna información que describa el contexto en el cual el sistema debe desarrollarse, estos se obtienen en una fase temprana de la ingeniería de requerimiento. Es por esta razón que se le llama **ingeniería de requisitos tempranos**, a las nuevas técnicas, enfocadas al **por qué**, es decir a **por qué** cada actor o agente debe realizar algo, y cómo la organización cumple sus objetivos a partir de eso.<sup>22</sup>

Dentro de los métodos de **ingeniería de requisitos tempranos**, el más conocido es *i\**,<sup>22</sup> el cual es un marco de trabajo que se centra en la modelación de las relaciones entre los actores del negocio. Esto permite:

- Obtener una mejor comprensión de las relaciones que se establecen entre varios agentes en una organización.
- Entender los fundamentos de las decisiones tomadas por los agentes.
- Documentar varias características encontradas en las fases tempranas de especificación de requerimientos.

De acuerdo con esta técnica, los participantes en una organización son actores con motivaciones (metas a cumplir, creencias, habilidades y compromisos). Cada actor depende de otros para alcanzar sus objetivos. Para ellos la técnica *i\** provee dos modelos: **modelo de dependencia estratégica** y el **modelo de razonamiento estratégico**, que permiten, fundamentalmente, representar la dependencia entre los actores y describir los intereses, preocupaciones y motivaciones de los participantes en el proceso, respectivamente.

*i\** permite entender los contextos en los cuales se obtienen los requisitos del software, de manera que resulta más fácil entender cómo los cambios en los objetivos de la organización afectan las funciones de los componentes de software que las soportan.

## PATRONES EN LA ORIENTACIÓN A AGENTES

Uno de los principios de ingeniería que más ha llamado la atención en la ingeniería de software en los últimos años ha sido sin dudas el auge de los **patrones**.

Los **patrones** son soluciones a problemas típicos. Especifican las buenas prácticas que han evolucionado para resolver un problema, y mantienen una definición concisa de elementos comunes, contexto, y los requisitos esenciales para una solución. Se han destacado por ser una poderosa herramienta para la descripción y reutilización del conocimiento empírico empleado en las distintas fases que componen el ciclo de vida del software, permitiendo un ahorro de tiempo y costo del proceso de desarrollo.

En el caso de la orientación a agentes, hay tres dimensiones en las que los patrones pueden implicar un salto cualitativo:

- Los agentes al ser un componente de nivel más alto que los objetos, pueden permitir el encapsulamiento y tipificación de elementos de solución más complejos. Esto implica que, debido a que los agentes son componentes menos acoplados, debe ser más fácil su extrapolación a nuevas situaciones donde un agente con similares metas y posibilidades puede dar solución a un problema similar.

- El hecho de que el agente permita establecer una comunicación menos desacoplada, puede posibilitar una mayor reutilización de los componentes de solución, y extender el uso de los patrones más allá de mostrar **cómo** puede resolverse un problema, sino contando con el componente de software que lo resuelve.

- El patrón puede no solo ser un ente inerte que espera ser llamado o identificado para cumplir su función, sino que puede tener entre sus intenciones la capacidad de buscar los espacios donde su inteligencia sea aplicable y llegar a ese lugar y ofrecer sus servicios. En este sentido, el patrón podría llevar una tarea de promoción de sus servicios.

Debido a lo anterior, a la hora de describir un patrón de agentes, es importante notar que los cambios anteriores sean tenidos en cuenta. Algunos ejemplos del uso de los patrones en la orientación a agentes se explican a continuación:<sup>24, 25</sup>

La utilización de patrones en el desarrollo de sistemas de agentes puede tener diversos alcances: la colaboración entre dos agentes se puede lograr mediante un patrón de servicio, un agente con un propósito específico se puede modelar aplicando un patrón de componente (proponen una solución integrada por la estructura de un agente junto con sus tareas), sus características específicas con un patrón de comportamiento y las acciones simples se pueden hacer con los patrones de acción (funcionalidades del sistema).

La modelación de estructuras sociales está siendo utilizada en la representación de **sistemas multiagentes**. En estos modelos se hace mayor énfasis en la arquitectura del sistema, viendo a los agentes como elementos sociales dentro de una organización compleja.<sup>14</sup>

GAIA<sup>14,25</sup> introduce la metáfora organizacional en su proceso de desarrollo y plantea que una organización es más que una

simple colección de roles, e introduce otras abstracciones organizacionales como son las normas.

Los agentes pertenecerán, en general, a una o más organizaciones, donde pueden desarrollar uno o más roles, para lo cual necesitan interactuar entre sí, para intercambiar conocimiento y coordinar sus actividades. Estas interacciones ocurren de acuerdo con patrones y protocolos que son determinados por la naturaleza del rol involucrado, definiendo la arquitectura de la organización del **sistema multiagente**.

Estas estructuras pueden ser de diferentes tipos, desde completamente planas (los agentes se conocen entre sí y están completamente conectados) hasta jerárquicas de varios niveles. Es probable que solamente un pequeño subconjunto de las estructuras que podrían ser concebidas sea utilizado continuamente, introduciendo la idea de patrones organizacionales orientados a agentes.

Resulta importante notar que solo unas pocas metodologías de agentes introducen los patrones. Cuando se avance en la investigación y definición de los mismos, se contará con una biblioteca de patrones cada vez más poderosa que permita enfrentar problemas futuros de una manera más eficiente.

## CONCLUSIONES


En este trabajo se ha realizado una introducción al nuevo paradigma de la orientación a agentes, tratándose fundamentalmente dos aspectos de la ingeniería de software: la **ingeniería de requisitos** y los **patrones**.

Se evidencia que junto con la evolución de los agentes debe cambiarse la concepción tradicional de la ingeniería de software, para adaptarse a las necesidades que demanda este tipo de tecnología. Especialmente en las dos dimensiones anteriores, se ve cómo el paradigma de los agentes ofrece una abstracción que permite tratar de mejor manera dos aspectos importantes:

- Enfrentar mejor la inserción y evolución de un sistema de software en un ambiente real, y prepararlo para ser más útil y flexible a los cambios de este.

- Crear componentes de software más flexibles en su aplicación y en la identificación de los problemas que pueden resolver.

La tecnología basada en agentes, parece prometedora para desarrollar sistemas de software, que puedan solucionar algunos problemas, que hasta el momento no han sido resueltos de forma satisfactoria, sin embargo, aún le falta madurez para lograr una propuesta estable como la que hoy se tiene con los procesos de desarrollo orientados a objetos.

Existen muchas cuestiones sin resolver, pero a juzgar por la evolución observada en los últimos años, parece un campo de investigación prometedor. 

## REFERENCIAS

1. JACOBSON, I. ; GRADY BOOCH Y J. RUMBAUGH: *El proceso unificado de desarrollo de software*, p. 438, Pearson Educación, Madrid, 2000,

2. **MARTIN FOWLER, K. S.:** *A Brief Guide to the Standard Object Modeling Language, UML Distilled*, Second Edition, p. 224, Addison Wesley, 1999.
3. **BLOOMBERG, J.:** "Web Services: A New Paradigm for Distributed Computing", 2001, www.theRationaleEdge.com.
4. ———.: Testing Web Services Today and Tomorrow. 2002, www.theRationaleEdge.com.
5. **CONALLEN, J.:** *Developing Applications for the Web Service Era*, S. Engineer Editor, 2003,
6. **JENNINGS, N. R.:** *On Agent-Based Software Engineering, Artificial Intelligence*, 117: pp. 277-296, 2000.
7. FIPA Methodology: Glossary of Terms, 2003, Foundation for Intelligent Physical Agents (FIPA), www.fipa.org.
8. **DELOACH, S.:** "Multiagent Engineering: A Methodology and Language for Designing Agent Systems", in *Agent-Oriented Information Systems*, 1999.
9. **BRESCIANI, P. et al.:** *Tropos: An Agent-Oriented Software Development Methodology*, Centro Per la Ricerca Scientifica e Tecnologica: Italia, 2002.
10. **PIERMARCO BURRAFATO, M. C.:** *Designing a Multi-agent Solution for a Bookstore with the PASSI Methodology*, 2002.
11. **JENNINGS, N. R.:** "An Agent-based Approach for Building Complex Software Systems", *Communications of the ACM*, No.4, pp. 35-41, April, 2001.
12. **BOOCH, GRADY; R. C. M. AND JAMES NEWKIRK:** "Object Oriented Analysis and Design with Applications", Second Edition, Addison Wesley Longman, Inc. 1998.
13. **O'MALLEY, SCOTT A. AND S. A. D.:** *Determining when to Use an Agent-Oriented Software Engineering Paradigm*, in Second International Workshop on Agent-Oriented Software Engineering, Montreal, Canada, AOSE'2001, 2001.
14. **ZAMBONELLI, F.; N. R. JENNINGS AND M. WOOLDRIDGE:** "Developing Multiagent System: the Gaia Methodology", *ACM Transactions on Software Engineering and Methodology*, 2003.
15. **WOOLDRIDGE, M.:** *Agent-Based Software Engineering*", in *IEE Software Engineering*, 1997.
16. **STAN FRANKLIN, A. G.:** "It is an Agent, or Just a Program?: A Taxonomy for Autonomous Agents, In Third International Workshop on Agent Theories," *Architectures and Languages*, Springer-Verlag, 1996.
17. **BRESCIANI, P. et al.:** *An Agent-Oriented Software Development Methodology*, Netherlands: Kluwer Academic Publisher, 2004.
18. **PAVLOS MORAÏTIS, E.; P. NIKOLAOS AND I. SPANOUDAKIS:** "Engineering JADE Agent with the Gaia Methodology", in R. Kowalszyk, J. Miller, H. Tianfield, and R. Unland editors, *Agent Technologies, Infrastructures, Tools, and Applications for e-Services*, Vol. 2592 of Lecture Notes in Computer Science, pp. 77- 91, Springer-Verlag, 2003.
19. **MEHDI DASTANI, J. H. et al.:** "Issues in Multiagent System Development", in *AAMAS'04*, 2004.
20. **BAUER, BERHARD; M. J. P. AND JAMES ODELL:** *Agent UML: A Formalism for Specifying Multiagent Interaction*, Paolo Ciancarini and Michael Wooldridge eds., pp. 91-103, Springer-Verlag, Berlin, 2001 (Held at the 22nd International Conference on Software Engineering (ISCE)).
21. **GIORGINI, PAOLO et al.:** *Agent-Oriented Software Development: A Case Study*, in 13th International Conference on Software Engineering Knowledge Engineering, Buenos Aires, Argentina, 2001.
22. **YU, E.:** *Agent-Oriented Modelling: Software Versus the World*, In: Agent-Oriented Software Engineering AOSE-2001 Workshop Proceedings. LNCS 2222, pp. 206-225, Springer Verlag, 2001.
23. **ALENCAR, F. M. et al.:** *From Early Requirements Modeled by the i\* Technique to Later Requirements Modeled in Precise UML*, in: III Workshop em Engenharia de Requisitos, 2000, Rio de Janeiro. Anais do III Workshop em Engenharia de Requisitos. Rio de Janeiro: Puc-Rio, Vol. 1. pp. 92-108, 2000.
24. **COSENTINO, M.; L. SABATUCCI AND A. CHELLA:** *Patterns Reuse in the PASSI Methodology*, pp. 294-310, ESAW, 2003.
25. **GONZÁLEZ-PALACIOS, J. AND M. LUCK:** *A Framework for Patterns in Gaia: A Case-Study with Organisations*, AOSE, pp. 174-188, 2004.



<http://aprendist.cujae.edu.cu/home/index.htm>