Agentes de Interfaz y Desarrollo de Software Orientado a Aspectos: Una Propuesta

L. Arredondo, N. Montaño, H. Nuñez

Universidad Central de Venezuela, Venezuela
Centro de Ingeniería de Software y Sistemas - ISYS

RESUMEN

Los agentes de interfaz se han convertido en una opción para cubrir un requerimiento transversal en todo producto de software; es decir, ofrecer soporte al usuario. Para ello se han desarrollado múltiples arquitecturas para agentes que buscan cumplir con tres tareas fundamentales: Conocer al usuario, interactuar con el usuario y ser competente en la asistencia al usuario. De igual manera, el desarrollo de software orientado a aspectos conforma una tendencia en la actualidad y su ejecución conlleva un conjunto de características que pueden ser aprovechadas dentro del contexto de agentes de interfaz y que conforman una parte importante de una propuesta de agente de interfaz inteligente desarrollada en el presente documento.

ABSTRACT

The development of interface agents has emerged as an option to meet a crosscutting requirement for all software: provide user support. In order to achieve that goal, multiple architectures have been developed for agents looking to fulfill three main tasks: Meet the user, interact with the user and be competent in user support. Similarly, the development of aspect-oriented software is a trend in software engineering and its execution leads to a set of features that can be exploited within the context of interface agents that form an important part of a proposed intelligent interface agent developed on this paper.

Keywords*:* Agentes de interfaz, Arquitectura, Aspectos.

1. Introducción

El desarrollo de agentes de interfaz es una disciplina que ha mostrado un gran auge en los últimos años. De manera paralela, el desarrollo de software apunta hacia la resolución de problemas cada día más complejos, por lo cual los mecanismos de soporte al usuario tradicionales no han resultado suficientes para mantener un nivel de rendimiento esperado del usuario con el software.

Una primera definición de agente de interfaz es aportada por [Mae94], la cual señala a los agentes de interfaz como entidades de software diseñados para asistir a usuarios humanos en las tareas que realiza en una computadora. Esta primera definición les confiere a los agentes de interfaz la etiqueta de asistentes al usuario. [AA09] continúan describiendo a este tipo de interacción como un proceso cooperativo en el que tanto el usuario y el agente inician comunicación, monitorean eventos y realizan tareas.

Como consecuencia de la etiqueta de asistentes al usuario, [Mid01] advierte que existen tres (3) aspectos que deben abordarse para lograr una exitosa colaboración con el usuario:

- Conocer al usuario
- Interactuar con el usuario
- Ser competente en la asistencia al usuario

Conocer al usuario se refiere a conocer las preferencias del usuario y sus hábitos de trabajo. La idea es ofrecer ayuda en el momento adecuado y evitar ser molesto, presentando información relevante que sólo generará incomodidad y retrasará las actividades del usuario. Sin embargo, conocer al usuario trae a su vez los siguientes retos [Mid01]:

- Extraer los objetivos e intenciones del usuario de la observación y el feedback.
- Obtener el contexto suficiente en el cual ajustar los objetivos del usuario.
- Adaptarse a los siempre cambiantes objetivos del usuario.
- Reducir el tiempo necesario para el aprendizaje de dichos elementos.

En cualquier momento, un agente de interfaz debe tener una idea de lo que el usuario está tratando de hacer para ofrecer asistencia efectiva. Además, ese estado del usuario debe estar acompañado de suficiente información contextual para evitar la ayuda irrelevante. De igual manera, el usuario puede realizar múltiples tareas mientras interactúa con un software, por ello el agente debe ser capaz de percibir los cambios en las intenciones del usuario. Por último, los usuarios no están interesados en entrenar a un agente para que les ofrezca soporte, estos deben estar capacitados desde una primera interacción con un bagaje mínimo de conocimientos que le permitan soportar al usuario y a la vez aprender del mismo para mejorar la forma en que le ofrece soporte.

En cuanto a interactuar con el usuario, [Mid01] identifica los siguientes desafíos:

- Decidir que tanto control delegar en el agente.
- Desarrollar confianza en el agente por parte del usuario.
- Escoger una metáfora para la interacción del agente.
- Hacer que los sistemas de software sean sencillos para el usuario novicio.

En el contexto de agentes de interfaz inteligentes, es preciso manejar con cuidado cuáles tareas son cómodas para que el usuario delegue en el agente. Esto por supuesto apunta hacia la confianza en la acción del agente. De igual manera, los agentes deben construirse siguiendo una metáfora que rija su forma de interactuar con el usuario. Finalmente, los agentes de interfaz deben proporcionar mecanismos para que el usuario pueda evolucionar de un aprendiz a un experto en las tareas realizables desde el sistema de software.

Por último, [Mid01] señala las dificultades al intentar que un agente sea competente en la forma en que asiste al usuario:

- Conocer cuándo (y si es necesario) interrumpir al usuario.
- Desarrollar tareas autónomamente en la forma que prefiera el usuario.
- Encontrar estrategias para la automatización parcial de tareas.

El mismo autor señala que existe muy poca investigación respecto a atacar estas dificultades. Sin embargo, indica campos como el trabajo cooperativo soportado por computadores como áreas de investigación donde es posible hallar algunas respuestas.

2. Arquitectura de un agente de interfaz

Un área de interés en el desarrollo de agentes de interfaz lo constituye la arquitectura de los mismos. Básicamente, estas arquitecturas refieren a un agente inteligente descrito por [RN04]: sensores, base de conocimiento y actuadores; pero adaptados al contexto de asistir al usuario en su interacción con un sistema de software. A continuación, se mostrarán algunas propuestas en torno a la arquitectura para un agente de interfaz.

En [AD98], los autores proponen una arquitectura basada en cuatro bloques fundamentales: una unidad de clasificación de tareas, un banco de memoria asociativa basada en redes neuronales, un sistema basado en reglas y una unidad de colaboración con otros pares (Figura 1).

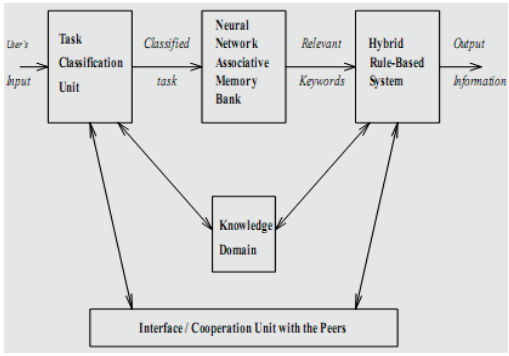

Figura 1. *Arquitectura presentada por [AD98]*

La unidad de clasificación de tareas recibe la entrada del usuario y procede a clasificar la tarea con base en el conocimiento del dominio que posee. El banco de memoria asociativa tiene por función generar palabras clave relacionadas con la tarea identificada por la unidad de clasificación. El sistema basado en reglas toma en cuenta las palabras clave identificadas por la unidad de memoria asociativa e infiere la información necesaria para ejecutar la tarea. Por último, la unidad de cooperación con otros pares permite al agente cooperar con otros agentes para preguntar y solucionar problemas.

Otro trabajo relacionado con el área de arquitecturas de agentes es el presentado por [GPG00]. En éste proponen una arquitectura basada en el enfoque de BDI (*Belief, Desire, Intention*). El agente se compone de una base de conocimientos, la cual posee conocimiento de tipo declarativo, de resolución y del dominio; una memoria de trabajo, en la que se registran los eventos disparados por el usuario; y una máquina de inferencia que se encarga de utilizar la base de conocimientos con base en los eventos registrados (Figura 2).

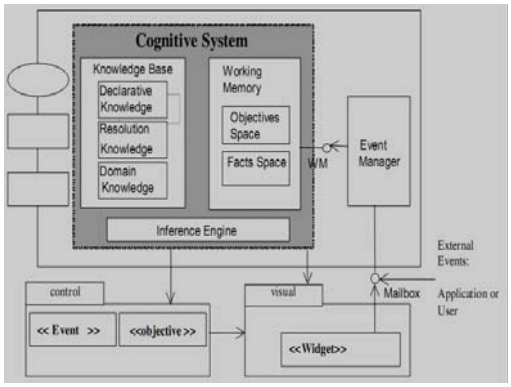

Figura 2. *Arquitectura propuesta por [GPG00]*

Por su parte, [BDM*03] identifican una arquitectura basada en una vista, un motor de sugerencias y unos perfiles. Para ello, las aplicaciones candidatas para integrar el agente deben estar concebidas bajo un esquema MVC (Modelo, Vista y Control). De esta forma, el motor registra el comportamiento del usuario en la vista, seguidamente toma de la base de perfiles la vista más adecua-

da a las necesidades del usuario y la propone a la aplicación (Figura 3).

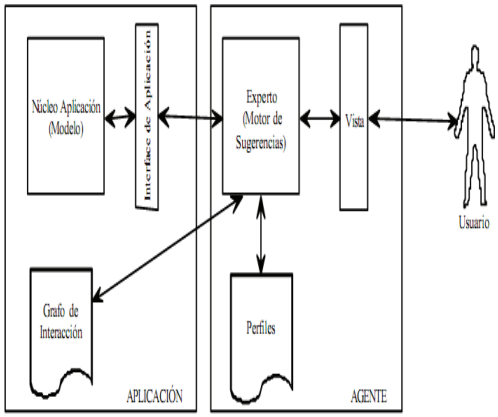

Figura 3. *Arquitectura para un agente de interfaz [BDM*03]*

Un elemento muy importante dentro de este esquema lo constituye el grafo de interacción, el cual refleja todos los posibles estados de la aplicación y su paso a otros estados en función de la comunicación con el usuario. Esto permite llevar al usuario hacia los estados que conduzcan al éxito en las tareas.

En 2007, [STM07] plantean una arquitectura de agente que aprende las preferencias del usuario implementado a través de la programación orientada a aspectos (Figura 4).

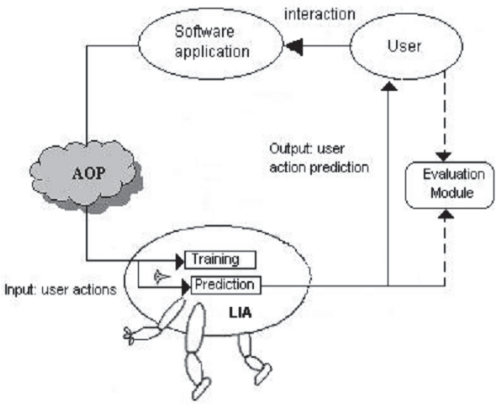

Figura 4. *Arquitectura propuesta por [STM07]*

Esta arquitectura le permitió a [STM07], desarrollar un agente para probar un método de aprendizaje supervisado de las acciones y preferencias del usuario, junto con unas métricas para la evaluación de las predicciones obtenidas a través de las inferencias del sistema basado en conocimiento.

[AA09], proponen una arquitectura basada en 3 componentes: un observador de eventos de la interfaz de usuario que mapea sus resultados hacia un modelo de la aplicación basado en árboles de tareas concurrentes y que le permite al agente identificar la tarea en la que se encuentra sumergido el usuario para luego asistirlo (Figura 5).

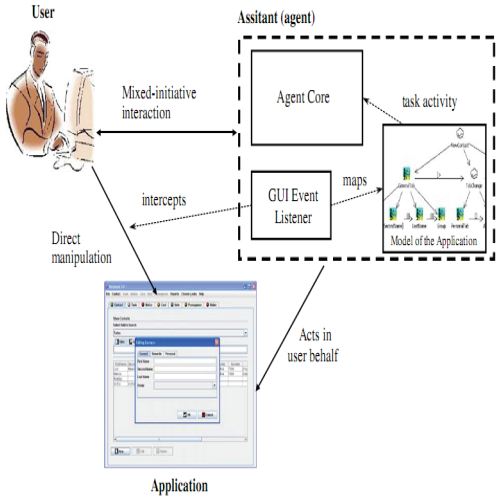

Figura 5. *Arquitectura propuesta por [AA09]*

De igual manera, [She09] desarrolla una arquitectura para un agente de interfaz que le permite implementar soporte a los usuarios de servicios de preguntas y respuestas frecuentes. Para ello, hace uso de modelos ontológicos del dominio y técnicas para el procesado de lenguaje para hacer más efectivos los sistemas de pregunta y respuestas frecuentes (Figura 6).

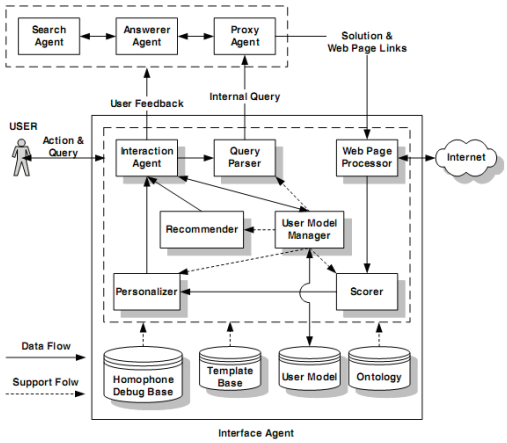

Figura 6. *Arquitectura presentada por [She09]*

En resumen, las arquitecturas revisadas tienen como características en común tener módulos orientados a monitorear las actividades del usuario para desencadenar acciones de predicción, a través del uso de una base de conocimiento que permita, de manera inteligente, decidir si el usuario necesita soporte y qué tipo de soporte.

3. Agentes de interfaz y la orientación a aspectos

El desarrollo orientado a aspectos es un paradigma para la construcción de software que en un principio se ideó para resolver problemas complementarios de código disperso y enmarañado que no se resuelven fácilmente con paradigmas tradicionales de desarrollo de software, como la orientación a objetos [TSS08].

La orientación a aspectos se basa en la separación de concerns (separación de intereses, asuntos o propiedades del sistema). Este principio se orienta hacia la descomposición del dominio del problema y de la solución, con el fin de reducir la complejidad, eliminar fallas de interpretación, y estructurar sistemas complejos a través de subsistemas, módulos o elementos simples de una forma más natural [EFB01], [KHH*01].

La descomposición está orientada a la modularización del espacio del problema bajo una perspectiva dual, es decir: en una, se definen los módulos principales de la solución (también llamados intereses base – *base concerns* en ingles), y en la otra separa la funcionalidad transversal en módulos independientes (ortogonales) denominados intereses transversales (*crosscutting concerns*) o aspectos al nivel del diseño y la implementación [EAB02]. La composición está dirigida a cruzar (tejer) este tipo de intereses en diferentes módulos del núcleo del sistema. El comportamiento cruzado puede corresponder a modelos, políticas, reglas de negocio, vistas de usuario, procesos de negocio, y atributos de calidad tales como auditoría (*logging*), seguridad, persistencias, rendimiento, entre otros. [TSS08].

La utilización del desarrollo orientado a aspectos se ha visto también como una forma de atacar problemas concernientes a características en sistemas multiagentes y en la incorporación de agentes de interfaz a aplicaciones existentes.

En cuanto al uso del paradigma de orientación a aspectos en sistemas multiagentes podemos citar a [GKL05], [GKS*06] y [KGLA05]; los cuales señalan un conjunto de requerimientos transversales al sistema multiagente (interacción, adaptación, autonomía, entre otros) que pueden ser canalizados a través de aspectos.

Por su parte, [STM07] relatan la construcción de un agente que predice el comportamiento del usuario, el cual fue implementado usando programación orientada a aspectos. Este ejemplo es de particular interés porque los autores señalan un conjunto de beneficios asociados al uso de este paradigma en el desarrollo de su agente de interfaz:

- Clara separación entre el sistema de software y el agente de interfaz.
- El agente puede ser fácilmente adaptado e integrado con otros sistemas de software.
- El sistema de software no necesita ser modificado para obtener la información necesaria para el agente.
- El código necesario para la captura de las acciones del usuario no está disperso en toda la aplicación, aparece en un solo lugar: el aspecto.
- Si se necesita alguna información adicional acerca de la interacción del sistema de software con el usuario solo necesita ser modificado el aspecto.

4. Agente de interfaz: una propuesta

Un agente de interfaz debe estar orientado hacia el desarrollo de las competencias del usuario: llevarlo de una situación de aprendiz hasta convertirlo en un experto en el uso y en el contexto del dominio del sistema de software para el cual fue creado.

Un agente así se le confiere entonces un conjunto de características pedagógicas que juegan un papel importante a la hora de desarrollarle. Por ello, en la construcción de un agente de interfaz que proporcione asistencia al usuario debe estar presente un diseñador instruccional y un experto del dominio, que permitan desarrollar un agente que cuente no solo con un sentido de asistir al usuario sino de formarlo en su interacción con el sistema de software. Este conocimiento necesario del dominio y el cómo presentarlo, debe actualizarse constantemente para que el agente pueda mantener sus intervenciones adecuadas a lo largo de su vida útil.

Respondiendo a las características mencionadas anteriormente, un modelo de casos de uso para el agente de interfaz deseado se muestra en la Figura 7.

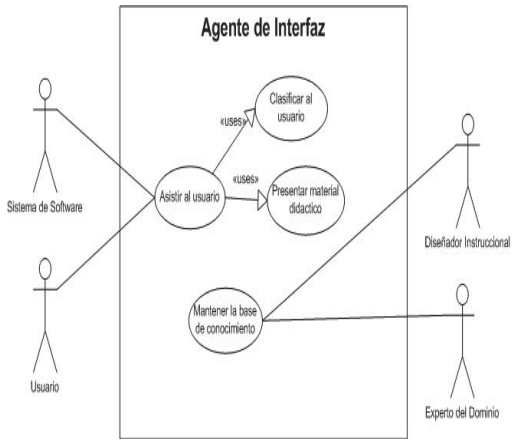

Figura 7.*Modelo de casos de uso asociado a la propuesta de agente de interfaz*

Aquí observamos que la razón principal del agente de interfaz propuesto es asistir al usuario en la interacción con el sistema de software sin importar en que funcionalidad se encuentre (Figura 8).

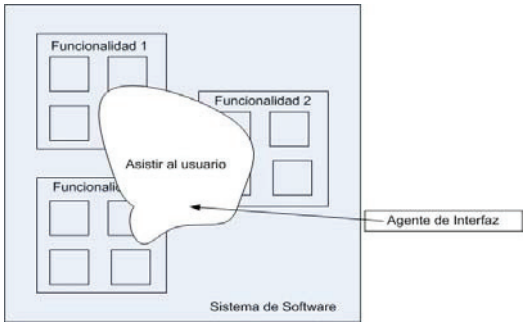

Figura 8. *El agente de interfaz como un elemento que toca múltiples funcionalidades*

Es por ello que el agente de interfaz puede ser modelado como un aspecto del sistema de software, porque efectivamente se transforma en un requerimiento transversal a las funcionalidades propias del mismo. Esto viene a reforzar todavía más las bondades encontradas por [STM07], porque entonces no es solo aprovechar las bondades de la programación orientada a aspectos sino que el agente de interfaz puede ser concebido en el proceso de desarrollo del sistema de software como un interés transversal (*crosscuting concern*) y ser así considerado durante el ciclo de vida del desarrollo del sistema de software.

En aras de atender las funcionalidades descritas en el modelo de casos de uso de la Figura 8, se presenta a continuación un modelo de la arquitectura propuesta para el agente de interfaz deseado (Figura 10).

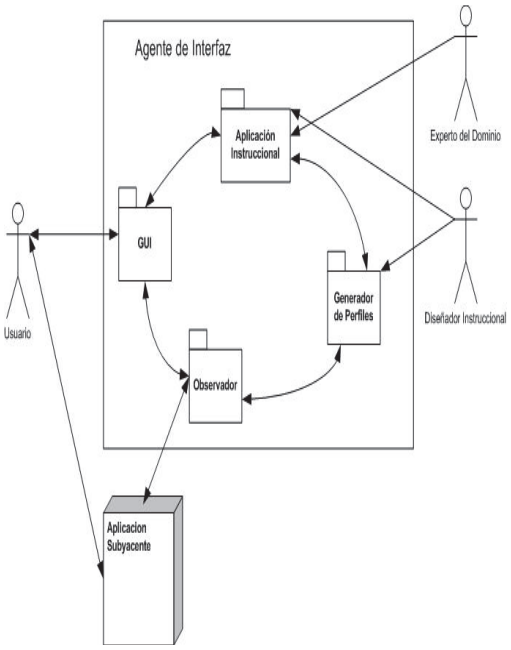

Figura 9. *Arquitectura propuesta para el agente de interfaz*

A continuación, se describen los componentes de dicha arquitectura:

- Aplicación subyacente: Se refiere a la aplicación a la cual sirve el agente de interfaz. Ésta debe estar concebida para permitir el progreso del usuario.
- GUI: Interfaz gráfica de usuario del agente. Constituye el puente de comunicación entre el usuario y las intervenciones del agente, sean éstas disparadas por el agente o solicitadas por el usuario.
- Observador: Paquete de rutinas orientadas a seguir la traza de operaciones del usuario en su interacción con la aplicación subyacente.
- Generador de Perfiles: Utilizando como entrada la traza de operaciones del usuario con la aplicación subyacente, el generador de perfiles cataloga al usuario. Para ello, el generador de perfiles hace uso de una base de conocimientos de escenarios [Mon05]. Dichos escenarios refieren a las posibles acciones alcanzables por parte del usuario en su interacción con el software.
- Aplicación instruccional: Toma como base el perfil y la traza del usuario para así presentar la ayuda apropiada a la situación del usuario y su nivel de pericia sobre el dominio de la aplicación subyacente. Para ello se hace uso de un conjunto de unidades instruccionales adaptadas al contexto de agentes de interfaz y que tienen por fin lograr la recuperación del usuario para la ejecución satisfactoria de sus tareas. Para la construcción de las unidades instruccionales se cuenta con la colaboración tanto de un experto del dominio como de un diseñador instruccional de forma tal que las unidades instruccionales estén orientadas a la formación del usuario desde un punto de vista sintáctico (manejo de la aplicación), como desde un punto de vista semántico (significado de las tareas alcanzables con la aplicación).
- Usuario: Se refiere al usuario de la aplicación subyacente. Puede interactuar directamente con el agente solicitando su ayuda o puede interactuar indirectamente una vez que el agente ha encontrado una razón para intervenir.
- Experto del dominio: Es el encargado de organizar y registrar el conocimiento acerca del dominio en donde opera la aplicación subyacente.
- Diseñador instruccional: Es quien tiene por tarea diseñar las estrategias pedagógicas e instruccionales adecuadas para cada perfil de usuario y situación en las que el agente deba intervenir.

De acuerdo a las descripciones presentadas de los componentes de la arquitectura, se puede observar que estos se alinean para atender los aspectos primordiales señalados por [Mid01] para la construcción de un agente de interfaz: Conocer al usuario, interactuar con el usuario y ser competente en la asistencia al usuario.

5. Conclusiones

En el área de Agente de Interfaz es necesario tener en cuenta tres aspectos para la construcción de los mismos [Mid01]:

- Conocer al usuario
- Interactuar con el usuario
- Ser competente en la asistencia al usuario

Es por ello que las arquitecturas de los agentes de interfaz deben apuntar hacia resolver estas inquietudes.

De manera general, las arquitecturas de agente de interfaz remiten al modelo de agente inteligente propuesto por [RN04]: sensores, base de conocimiento y actuadores; pero adaptados al contexto de asistir al usuario en su interacción con un sistema de software.

Se presenta una propuesta de agente de interfaz que está orientado no solo a asistir al usuario en un sentido sintáctico de operación de la aplicación, sino también en un sentido semántico en el dominio en que opera el sistema de software.

Dicha propuesta tiene como característica fundamental aprovechar las potencialidades del desarrollo orientado a aspectos al considerarse a sí mismo (el agente de interfaz) como un aspecto transversal a la aplicación subyacente.

6. Trabajo futuro

En la actualidad se encuentra en fase de diseño y construcción los módulos orientados a seguir la traza del usuario y el generador de perfiles para una aplicación con propósitos educativo-prácticos en el área de inteligencia artificial. Se planifica completar el agente de interfaz siguiendo la arquitectura presentada.

Referencias

[AD98] Aboulenien, H. y De Wilde, P. : A simple interface agent. Joint Conference on Information Sciences (JCIS). (1998).

[AA09] Armentano, M y Amandi, A.: A framework for attaching personal assistants to existing applications. Computer Languages, Systems & Structures 35. (2009). 448-463.

[BDM*03] Balsas, J., Díaz, M., Montejo, A., Martinez, F., Garcia, M. y Ureña, L.: Una Arquitectura para agentes de interfaz inteligentes: el ordenador sugerente. (2003).

[EAB02] Elrad, T., Aldawud, O., Bader, A.: Aspect Oriented Modelling: Bridging the gap between implementation and design. GPCE 2002, LNCS 2487. Springer Verlag . (2002). 189-201.

[EFB01] Elrad, T., Filman, R., Bader, A.: Aspect Oriented Programming. Communications of ACM Vol. 44 No. 10. (2001).

[GKL05] Garcia, A., Kulesza, U. y Lucena, C.: Aspectizing multi-agent systems: from architecture to implementation. SELMAS 2004, LNCS 3390, Springer Verlag. (2005). pp 121-143.

[GKS*06] Garcia, A., Kulesza, U., San't Anna, C., Chavez, C. y Lucena, C.: Aspects in Agent-Oriented Software Engineering: Lessons Learned. AOSE 2005, LNCS 3950. Springer Verlag. (2006). pp. 231 – 247.

[GPG00] Gomez-Sanz, J., Pavón, J. y Garijo, F.: Intelligent Interface Agents Behavior Modeling. MICAI 2000, LNAI 1793. Springer Verlag. (2000). pp. 598-609.

[KGLA05] Kulesza, U., Garcia, A., Lucena, C. y Alencar, P.: A generative approach for multi-agent system development. SELMAS 2004, LNCS 3390. (2005). pp. 52-69.

[KHH*01] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.: Getting Started with AspectJ. Communications of ACM Vol. 44 No. 10. (2001).

[Mae94] Maes, P.: Agents that reduce work and information overload. Communications of the ACM 37(7). (1994). pp. 30-40.

[Mid01] Middleton, S.: Interface agents: A review of the field. Technical Report Number: ECSTR–IAM01-001. ISBN: 0854327320. (2001)

[Mon05] Montaño, N.: Un enfoque interdisciplinario para la construcción de productos de software basados en el proceso enseñanza-aprendizaje del usuario. Tesis Doctoral Universidad Central de Venezuela. (2005).

[RN04] Russell, S. y Norvig, P.: Inteligencia Artificial: Un enfoque moderno. Ed. Prentice Hall. (2004). pp. 1212

[STM07] Serban, G., Tarta, A., y Moldovan G.: A learning interface agent for user behavior prediction. Human-Computer Interaction, Part III, HCII 2007, LNCS 4552, Springer Verlag. (2007). pp. 508–517.

[She09] Sheng-Yuan Y.: Developing of an ontological interface agent with template-based linguistic processing technique for FAQ services. Expert Systems with Applications 36. (2009). pp 4049-4060.

[TSS08] Tabares, M., Salinas, G. y Salinas, E.: El desarrollo de software orientado a aspectos: un caso práctico de ayuda en línea. Avances en Sistemas e Informática, Vol. 5, No. 2. Medellín, ISSN 1657-7663. (2008).