INTERACCIÓN 2.000

I JORNADAS de INTERACCIÓN PERSONA-ORDENADOR

Actas de las Jornadas

Facultad de Psicología
Universidad de Granada
Granada, 19 y 20 de Junio del 2000

AIPO

Interacción'2000
I Jornadas de Interacción Persona-Ordenador

Comité Organizador

José J. Cañas.
Dpt. Psicología Experimental.
Facultad de Psicología. Universidad de Granada

Miguel Gea.
Dpt. Lenguajes y Sistemas Informáticos.
ETSI Informática. Universidad de Granada

Comité de Programa

José J. Cañas (Universidad Granada)
Pablo Castells (Universidad Autónoma Madrid)
Miguel Gea (Universidad Granada)
Jesús Lorés (Universidad de Lleida)

Entidades Colaboradoras

Facultad de Psicología
Asesoría Antonio Gijón

AIPO

Granada, 19-20 de Junio del 2000

Indice

Arquitectura para Agentes de Interfaz Inteligentes: el ordenador *sugerente*

J.R. Balsas, M.C. Díaz, A. Montejo, F. Martínez, M. García, L.A. Ureña
Departamento de Informática
Universidad de Jaén, Av. De Madrid 35, Jaén
e-mail: laruena@ujaen.es
tlf: 953 212 445 – fax: 953 212 420

Resumen

Desde los asistentes hasta los tutores interactivos podemos identificar una nueva forma de entender la interfaz de usuario. La interfaz inteligente se hace necesaria desde el momento en que la complejidad de las aplicaciones crece, así como la valiosa capacidad de reducir el tiempo de aprendizaje. Tras una clasificación de las soluciones comerciales más extendidas, identificamos las características deseables por parte de estos agentes inteligentes encargados de la interfaz con el usuario. Después proponemos una nueva arquitectura para el desarrollo de sistemas basadas en este tipo de interfaces.

Palabras clave: interfaz de usuario, agentes inteligentes, arquitectura, interacción persona-ordenador

1 Introducción

El uso de interfaces inteligentes para todo tipo de aplicaciones es una realidad creciente. Desde los asistentes hasta los tutores interactivos podemos identificar una nueva forma de entender la comunicación entre el usuario y la herramienta. La idea de un *diálogo interactivo* libera al usuario de la necesidad de conocer a fondo el uso de la herramienta en cuestión. La interfaz inteligente se hace necesaria desde el momento en que la complejidad de las aplicaciones crece, así como la valiosa capacidad de reducir el tiempo de aprendizaje.

Como se refleja en el trabajo de Brown y Santo [1], es difícil diseñar la interfaz debido a los diferentes sistemas informáticos, la carencia de estándares en cuanto a interfaces, la necesidad de que la interfaz se adapte a cierto estándar preestablecido y la falta de una metodología apoyada en una teoría sólida para la especificación de los requisitos de la interfaz. Entendemos, por tanto, que la mejor solución consiste en analizar aquellos sistemas que realmente han funcionado en el mundo comercial y, a partir de ellos, abstraer todo lo posible sus características para la forja de una nueva arquitectura.

2 Paradigma actual

Se hace necesario, primero, establecer una definición clara de qué consideramos como agente. Usaremos para ello la definición dada por Lieberman [2], quien nos dice que *"un agente es cualquier programa que puede ser considerado por el usuario como un asistente o un ayudante en su forma de actuar, en lugar de una mera herramienta al estilo de las interfaces convencionales de manipulación directa"*.

Si bien en un principio se hablaba de interacción entre la persona y el ordenador, debemos abstraer esta realidad como un modelo de interacción entre persona y sistema. Esto nos da una visión más amplia de lo que hoy en dia ocurre en los entornos distribuidos, cooperativos y multitarea. Las interfaces de última generación vienen acompañadas de asistentes y otros elementos bajo los cuales se ocultan agentes de interfaz. Los denominados *agentes software* están presentes en casi todas las interfaces de las últimas versiones de muchos programas.

3 Clasificación

Atendiendo a las soluciones comerciales actuales, identificamos la clasificación de las interfaces inteligentes como sigue:

- **Basados en menús**. Desde hace algún tiempo, los menús han dejado de ser estáticos, para presentarse con capacidades tales como sensibilidad al contexto, personalización, crecimiento adaptable... (*Windows, MacOS, ...*)
- **Asistentes**. Este tipo usa la metáfora del "experto" que nos acompaña en nuestra sesión de trabajo (*Microsoft Office*). Con apariencia antropomorfa, el asistente nos sugiere formas de actuación y ayuda.
- **Basados en plantillas/ejemplos**. Son los determinados "wizards" (*WinZip, InstallShield, ...*), donde, paso a paso, vamos completando la información necesaria en la consecución de una determinada tarea. Existen muchas propuestas de este tipo, desde las plantillas sin más, hasta tutores completos.
- **Mixtos**. Es difícil poder encuadrar las propuestas actuales en una sola de estas categorías. Hoy día es palpable que los interfaces intentan ofertar los tres tipos descritos anteriormente.

4 Características deseables

Como en todo proceso de ingeniería del software, pasamos a identificar los requisitos deseables de nuestro futuro sistema. Considerando contempladas las características exigidas a toda interfaz [3], vamos a centrarnos sobre estos requisitos: los deseables en los agentes y los deseables en las aplicaciones.

En cuanto a los agentes. De todas las propuestas estudiadas, se extraen las siguientes características principales:

- *Colaboración*: capacidad de los agentes para comunicarse entre si en forma de llamadas-respuesta.
- *Autonomía*: capaces de realizar *ciertas* tareas de forma independiente, sin necesidad de comunicación con otros agentes y/o usuario/s.
- *Adaptable*: aprenderán de la evolución que la interacción experimente y serán transportables a diferentes entornos y contextos.

Existen más características, también contempladas. Para más información ver [4]. Las aquí incluidas nos parecen las más críticas para un agente de interfaz.

En cuanto a las aplicaciones. En un marco como el que se pretende implantar el sistema a proponer, se hace necesaria la determinación de ciertas características que las aplicaciones deberían contemplar para su total integración en nuestro esquema:

- *Examen*: Las aplicaciones deben facilitar la supervisión de la interacción.
- *Memorización*: Debe poder registrarse esta interacción (ver [5]).
- *Descriptible*: La comunicación con la aplicación debe ser posible a través de algún lenguaje o protocolo (no sólo mediante la interfaz que la aplicación misma posee); por ejemplo, *AppleScript* en entornos *Macintosh*, lenguaje *Visual Basic* en *Windows*... ([6])

5 Arquitectura

En este apartado vamos a presentar el diseño más simple de nuestra arquitectura, aquella en la que un único usuario se enfrenta a una sola aplicación. La idea de agentes de interfaz no es nueva, pero aunque se han implementado muchas soluciones, basadas sobre todo en prototipos "forzados" sobre sistemas previos [6], nuestra propuesta plantea un sistema nuevo, de principio a fin, que proporcione una solución definitiva a los problemas a los que se enfrentan los intentos seguidos hasta el momento y que, al mismo tiempo, satisfaga todos los requisitos identificados para un agente de interfaz inteligente. Básicamente consiste en una maduración del modelo multi-agente PAC de Joelle Coutaz [10]. Como podemos ver en la *figura 1*, en nuestro modelo el usuario se comunica con la aplicación a través de un *agente de interfaz* [2].

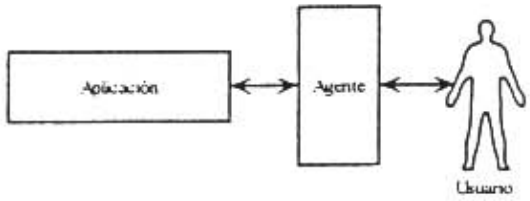

Figura 1. Esquema básico de un agente de interfaz

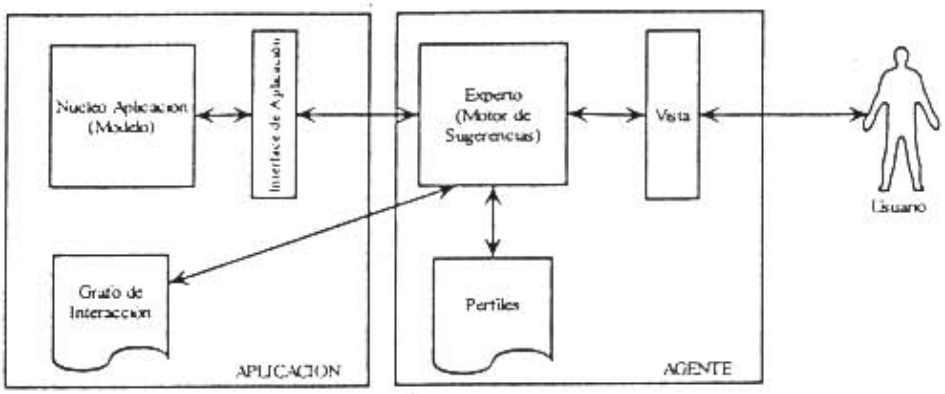

Figura 2. Esquema de un agente de interfaz monousuario y monoaplicación

La *figura 2* muestra con más detalle el modelo propuesto En este esquema el usuario interacciona con una *vista* (similar a la *presentación* del modelo PAC y a la *vista* del modelo MVC (*Modelo-Vision-Control*). Esta vista transfiere y recibe información de interacción con el núcleo del agente, que nosotros hemos denominado *motor de sugerencias*. Este motor registra el comportamiento del usuario en una base de datos de *perfiles* y propone una interfaz de comunicación a la *vista* en función de los datos del *perfil* y del *grafo de interacción*. Una vez determinada una comunicación, se notifica a la aplicación de la acción a realizar mediante una interfaz cuyo protocolo sea estándar.

Los *perfiles* son, básicamente, visiones parciales enriquecidas del grafo de interacción de la aplicación. El grafo de interacción es la estructura fundamental del sistema. Un grafo de interacción refleja todos posibles estados de una aplicación y su paso a otros estados en función de la comunicación que se establece con el usuario. Podemos considerarlo como un subgrafo dentro de un espacio de fases de dimensión igual al número de todas las posibles comunicaciones que se establecen con el usuario. Aquí NO nos interesa el estado de la aplicación en cuanto a procesamiento, sino sólo en cuanto a su comunicación con el usuario. Cada aplicación posee su propio grafo de interacción. Los grafos de interacción de distintas aplicaciones pueden estar conectados. Es un modelo para lo que se denomina *base de conocimiento* en [7].

El *perfil* de un determinado usuario es un grafo que crece a partir de los caminos que el usuario va siguiendo por el grafo de interacción de la aplicación. Similar al *modelo mental*

propuesto por Lieberman [6]. Toda esta dinámica de flujo del usuario hacia la aplicación es simétrica, es decir, también se aplica en la dirección de la aplicación hacia el usuario. Se establece así un sistema de mensajes de doble sentido: *orden-notificación*.

6 Conclusiones y trabajo futuro

Intentamos con nuestro modelo superar los problemas que plantea el trabajo con sistemas previos, rediseñando un nuevo sistema que permita integrar fácilmente todos los requisitos, modelos y protocolos diseñados en el mundo de las interfaces de usuario basadas en agentes.

Toda esta estructura tiene multiples extensiones que amplian el ámbito de actuación de esta arquitectura. Una de estas extensiones es crear un sistema *multiaplicación* para que un solo agente gestione la interacción con varias aplicaciones. También contemplamos la posibilidad de agentes multiusuario, que permita la comunicación y la colaboración entre usuarios, además de realizar tareas cooperativas entre usuario y agentes (como en [8]).

La arquitectura propuesta permite integrar un *Servidor de Aplicaciones*, de forma que el agente se comunica con él en lugar de con la aplicación. Esta estructura nos permitiría tener un sistema distribuido de aplicaciones. Cuando integramos varias aplicaciones en el sistema hemos de tener en cuenta que los grafos de interacción de cada aplicación no son elementos aislados, por lo tanto, debemos establecer algún método de relación entre grafos.

Para la comunicación de los agentes sería interesante utilizar el formato *ACL* (*Agent Communication Lenguage*, [?]), el cual utiliza *KQML* (*Knowledge Query and Manipulation Lenguage*) para la comunicación del agente hacia el exterior (p.e. otro agente) y *KIF* (*Knowledge Interchange Format*) para comunicaciones internas.

En la comunicación entre el motor y la aplicación se estudia la utilización de DCOM o CORBA que son las interfaces más extendidas en la actualidad.

Para la implementación del agente se puede utilizar Java, ya que son multiples las ventajas que ofrece. Además de la portabilidad, el lenguaje Java soporta el uso de comunicaciones y el desarrollo de aplicaciones multitarea y multiplataforma.

La evaluación del sistema se basará en el desarrollo de perfiles mediante un entrenamiento masivo. En esta fase el sistema registrara la interacción de un número determinado de usuario durante un tiempo prefijado. Una vez obtenidos varios perfiles se analizarán las diferencias de las métricas satisfacción y productividad del usuario.

Referencias

[1] Scott M. Brown, Eugene Santo Jr. "IaDEA: A Development Enviroment Architecture for Building Generic User Interface Agents", 1998

[2] Henry Lieberman, "Autonomus Interface Agents", ACM Conference on Human-Computer Interaction [CHI-97], Atlanta, March 1997.

[3] L. Bartfield. "The User Interface. Concepts & Design". Addison Wesley. 1993

[4] Todd Sundsted. "An introduction to agents". Java World on-line publication, Junio 1998.

[5] Yezdi Lashkari, Max Metral, Patti Maes, "Collaborative Interface Agents". IT Media Laboratory, Autonomous Agents Group.

[6] Henry Lieberman. "Integrating User Interface Agents with Conventional Applications". Proceedings of the ACM Conference on Intelligent User Interfaces, San Francisco, January 1998.

[7] Robert J.K. Jacob James G. Schmolze , "A Human-Computer Interaction Framework for Media-Independent Knowledge". Department of Electrical Engineering and Computer Science

[8] M. Tambe, D.V.Pynadath, N. Chauvat y otros. "Adaptive Agent Integration Architectures for Heterogeneous Team Members". Information Sciences Institute and Computer Science Department, University of California.

[9] R. Genesereth, S.P. Ketchpel. "Software Agents". Standford University. 1994

[10] Gaelle Calvary, Joëlle Coutaz, Laurence Nigay. "From Single-User Architectural Design to PAC: a Generic Software Architecture Model for CSCW". CHI 97 Conference Proceedings, Atlanta, Georgia, 242-249. March 1997. ACM/Addison-Wesley 1997, ISBN0-201-32229-3