Adaptabilidad de Interfaces de Usuario por Reflexión

Víctor López-Jaquero, Francisco Montero, Antonio Fernández, Maria Lozano

Grupo de Investigación LoUISE, Escuela Politécnica Superior de Albacete
Universidad de Castilla – La Mancha. Campus Universitario. 02071 – Albacete
Teléfono: 967 59 92 00 – Fax: 967 59 92 24
{ victor, fmontero, caballer, mlozano}@info-ab.uclm.es

Resumen. La capacidad de adaptación de los sistemas software es una característica altamente deseable. El hecho de desarrollar una misma funcionalidad y dotarla de diferentes interfaces de usuario para que pueda estar disponible en múltiples dispositivos o para diferentes usuarios potenciales [9] no parece la mejor solución. Este artículo propone una arquitectura reflexiva para el desarrollo de interfaces de usuario cuyo fin último es la consecución de adaptabilidad dinámica.
Palabras clave. Interfaces de usuario, modelos, conectores, reflexión

1. Introducción

La importancia del diseño y creación de interfaces de usuario (IU) es cada vez mayor, siendo considerado un campo de investigación de gran relevancia dentro de la Ingeniería del Software y, tal vez, uno de los que producen un mayor impacto en los diferentes usuarios de los sistemas de información. La interfaz de usuario determina la forma de interactuar entre el usuario y el sistema. En su estudio y en su desarrollo tienen cabida varias disciplinas, desde la definición de métodos de análisis y diseño de IU, la creación de nuevas alternativas de representar o acceder a la información hasta el estudio de los factores humanos que están detrás del manejo de dichas interfaces; ergonomía, psicología social y organizacional, sociología, etc.

La complejidad del diseño e implementación de IU, viene justificada por todas las contribuciones citadas [6], pero, tal vez, la principal dificultad sea la asociada a comprender las tareas que el usuario desea realizar sobre el sistema y las características de los diferentes usuarios que lo manejarán. Fruto de esta última aportación –la debida al perfil del usuario final- la característica de la que adolecen los IU, generados en la actualidad, es la falta de adaptabilidad. La mayoría de las aproximaciones de desarrollo de IU no proporcionan los mecanismos adecuados para identificar y especificar las necesidades y requisitos de los usuarios, así como la validación de esos requisitos durante todo el proceso de desarrollo del software y especialmente cuando la adaptabilidad se busca en tiempo de ejecución.

El presente artículo presenta la adaptabilidad como uno de los retos más importantes a la hora de desarrollar IU, sobre todo considerando las nuevas formas de interactuar que surgen continuamente y el crecimiento de usuarios potenciales que acceden a los diferentes sistemas de información disponibles. Estableciendo las

herramientas de desarrollo de IU basadas en modelos como aquellas que más ventajas ofrecen para el desarrollo de los mismos [8], veremos que no son suficientes para dar respuesta al reto de la adaptabilidad en todas sus facetas. Se propone una arquitectura reflexiva basada en el uso de conectores para dar solución a un dinamismo deseable en la generación de interfaces.

2. Generación de Interfaces de Usuario

En la literatura podemos encontrar varias propuestas para solucionar el problema de la generación de IU que incorporen adaptabilidad. Este problema pasa por la integración de factores humanos en el proceso de desarrollo de software, es decir, tener en cuenta las necesidades del usuario final desde las primeras fases del desarrollo e incorporar estos aspectos en un modelo de usuario como parte del proceso de desarrollo.

Durante los últimos años se han creado distintas herramientas para soportar el desarrollo de IU, por ejemplo, Toolkits, User Interface Management Systems (UIMS), Interface Builders, etc. Una taxonomía de estas herramientas fue presentada por Myers en [7]. De ellas las que parecen ser las más adecuadas para el desarrollo de IU son las herramientas de generación basadas en modelos. Con ellas es posible especificar tanto el comportamiento dinámico, como la estructura de la IU.

La idea principal de la generación de IU basada en modelos es que toda la información necesaria para el desarrollo de la IU está recogida en modelos declarativos. Entre las herramientas de generación de IU basadas en modelos se pueden citar: UIDE, ADEPT, HUMANOID, ITS, MECANO, MOBI-D, MASTERMIND, TRIDENT, AME, OVID o IDEAS. Es esta última metodología de desarrollo de IU la que tomamos como referencia [5].

Por regla general, todas las herramientas de desarrollo de IU basados en modelos utilizan unos modelos declarativos diferentes. Los modelos más frecuentemente utilizados son: los modelos de tareas, de dominio, de usuario, de diálogo y de presentación.

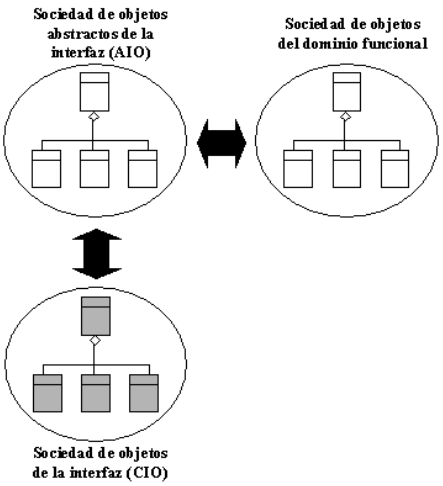
En nuestra búsqueda de la adaptabilidad todos los modelos son importantes, pero son de resaltar el modelo de usuario y el de diálogo. El modelo de usuario describiría las características del usuario final de la aplicación. Las características del usuario se pueden clasificar como independientes de la aplicación o dependientes de la aplicación. Las características independientes de la aplicación incluyen aspectos personales como las preferencias, las destrezas, las habilidades, etc. Mientras que las características dependientes de la aplicación incluyen objetivos, conocimiento del sistema y de la aplicación, etc. El modelo de diálogo se utiliza para describir la conversación hombre-máquina. Describe cuando el usuario final puede invocar comandos, seleccionar o especificar entradas de información y cuando el ordenador puede requerir información del usuario y presentar la información de salida. En otras palabras, el modelo de diálogo describe la secuencia de entradas, salidas y su interrelación.

Modelo de usuario y de diálogo, además, deben incluir los resortes necesarios para tomar decisiones de diseño de la interfaz de usuario, por ejemplo, la selección de los

objetos de interacción a partir de determinadas características del usuario o de la tarea y la información a presentar.

3. Funcionamiento de las Interfaces de Usuario

Resultado de la aplicación de la metodología IDEAS aparecen tres sociedades de objetos (ver Fig. 1a). Por un lado, aparecerá la sociedad que representa los objetos asociados al dominio funcional de la aplicación, por otro los objetos que representan de forma abstracta la IU (AIO), y por último, los objetos concretos de la plataforma donde se presenta la interfaz (CIO). Por tanto, el funcionamiento de la interfaz consistirá en la interacción entre los objetos de la IU, la interacción entre los objetos del dominio y las interacciones entre los objetos de ambas sociedades.




```
ContractClass CWindow_Management
  identification
    code: (code)
  task
    aWindow:Window_Management
  partners
  GUI objects // (AIO)
    aMainWindow:Window;
  domain objects
    aConfig:Configuration;
  constant attributes
    code:nat;
  variable attributes
    ConfigChanged:bool(FALSE);
  coordination
  CW: when aMainWindow.calls(aWindow.Close)
      do aConfig.SaveWindow(aMainWindow)
      with ConfigChanged = TRUE
  ...
  valuations
    [SavedConfig] ConfigChanged = FALSE
  ...
end class CWindow_Management
```

Fig. 1. (a) Sociedades de objetos en un sistema. (b) Especificación de un contrato en IDEAS.

La metodología IDEAS plantea la especificación de las interacciones entre sociedades de objetos abstractos y sociedades de objetos del dominio funcional de acuerdo al término contrato [1]. La Fig. 1b muestra un extracto de la especificación de un contrato de acuerdo a la plantilla propuesta para tal propósito dentro de la metodología para desarrollo de IU *IDEAS*. Para el ejemplo se ha escogido la especificación de la interacción entre la ventana principal de una aplicación (perteneciente a la sociedad de objetos de la IU (AIO)) y un objeto del dominio que representa la configuración global del sistema (*partners*) por ser está coordinación ampliamente conocida. Dicha interacción se engloba dentro de una tarea a la que se ha denominado *Window_Management* (gestión de las ventanas). El contrato representa explícitamente las reglas que determinan la forma en que la interacción

entre objetos debe ser coordinada para satisfacer los reglas del negocio, así como los mecanismos que hacen posible cambiar los requisitos del sistema sin tener por ello que modificar los objetos básicos que componen el sistema, ni tampoco en nuestro caso la interfaz. En definitiva, el contrato especifica la comunicación que se establece entre los objetos de interfaz y los objetos del dominio para la realización de las tareas requeridas por el usuario.

La adaptabilidad de una IU se puede llevar a cabo bien durante el diseño o bien en tiempo de ejecución. La primera posibilidad viene soportada normalmente por un modelo de usuario que representa los diferentes roles que el usuario puede desempeñar, otra posibilidad es la utilización de un modelo de usuario para cada tipo de usuario, mientras que la adaptabilidad en tiempo de ejecución requiere la utilización de un modelo de diálogo, bien para cada tarea, usuario o incluso dispositivo. En función de estas posibilidades, en función de la noción de contrato, existe una limitación de la aplicabilidad de los contratos. Con él podemos especificar la IU de forma estática pero no podemos proporcionar el dinamismo exigido para la consecución de la adaptabilidad deseada.

4. Búsqueda de adaptabilidad en IU: Proporcionando dinamismo

En la consecución del dinamismo exigido por la búsqueda de una adaptabilidad en tiempo de ejecución, se propone la utilización del concepto de conector [2]. Este vendría a complementar el presentado en el apartado anterior –contrato–. Un conector se define como un conjunto de roles y la especificación del nexo que los une. Los roles describen el comportamiento que debe desempeñar cada una de las partes implicadas en la interacción, mientras que la unión describe cómo se deben coordinar las instancias de cada uno de los roles.

Los conectores tienen ventajas frente a los contratos. La principal de estas ventajas está en que permiten que un determinado rol sea instanciado por distintos componentes siempre que estos sean capaces de dar el servicio requerido. Es con esta última idea con la que se puede lograr el dinamismo y la generación dinámica de IU en función de la tarea a realizar y de la información involucrada en dicha tarea.

Los conectores serían parte integrante de una arquitectura reflexiva orientada a objetos de desarrollo de aplicaciones [3]. En ella se incorporan dos partes: una parte de la aplicación y otra parte reflexiva. Es decir, los componentes que representan el propio sistema y aquellas que representan la aplicación residen en dos niveles diferentes: nivel base y nivel meta, respectivamente.

- Nivel base, contiene los objetos que resuelven un problema y retornan información sobre el dominio de la aplicación o dominio externo.
- Nivel meta, formado por objetos que desarrollan computación sobre el sistema materializado por objetos en el nivel base, en nuestro caso la computación que realizan sería la adaptabilidad. El dominio computacional interno del sistema, resuelve problemas y retorna información sobre el nivel base. A este nivel se

llevaría a cabo el paso de AIOs a CIOs mediante conectores y devolviendo el control al nivel base.

La literatura identifica diferentes modelos reflexivos dentro de un paradigma orientado a objetos, entre ellos están el de modelo estructural y el de modelo de comportamiento. La diferencia entre uno y otro es qué establecemos a nivel meta, si hay metaclases será el primer caso, si hay metaobjetos será el segundo. OASIS [4] proporciona un modelo estructural para la especificación de un sistema reflexivo, basándonos en él es posible el desarrollo de un modelo de metaobjetos.

En un lenguaje reflexivo basado en metaobjetos, el comportamiento de cada objeto es determinado por el metaobjeto asociado que controla y define las operaciones del objeto. De esta manera, es posible, por ejemplo, introducir nuevo comportamiento, cambiar el mecanismo de herencia, controlar la activación de los objetos o modificar la ejecución de los métodos. En nuestro caso las acciones que recaen sobre el nivel meta son la de establecer la interfaz en función de la tarea y de la información que estemos manipulando para el desarrollo de la misma.

Un protocolo de metaobjetos brinda a los usuarios la habilidad de modificar incrementalmente el comportamiento e implementación de un lenguaje de programación y la habilidad de escribir programas con dicho lenguaje. El primer mecanismo está relacionado con la información que puede ser manipulada en el nivel meta. El comportamiento computacional del nivel base es transformado en dato, esta actividad es denominada reificación. Las entidades reificadas constituyen la metainformación por la cual es posible desarrollar comportamiento reflexivo. Los otros dos mecanismos están relacionados con definir cómo la asociación entre ambos niveles es implementada y cómo el mecanismo de reflexión es activado. Es donde habrían entrado en escena los conectores.

5. Conclusiones

Este artículo presenta como reto fundamental en el desarrollo de IU la adaptabilidad. Esta cualidad presentaría dos componentes una estática y otra dinámica. Utilizando herramientas basadas en modelos para el desarrollo de IU, el dinamismo se lograría mediante la utilización de una arquitectura reflexiva constituida, en principio, por dos niveles: uno base y otro meta. En el nivel base estaría dispuesta la funcionalidad deseada por nuestra aplicación. En el nivel meta se dispondrían los objetos que en función de la tarea a realizar, la información que esta tarea involucrase y el dispositivo final con el que el usuario interactuase determinaría la conexión entre objetos de interfaz abstractos y objetos de interfaz concretos.

Agradecimientos

Este trabajo está financiado a través de los proyectos CICYT TIC 2000-1673-C06-06 y CICYT TIC 2000-1106-C02-02.

Bibliografía

1. Andrade, L.F., Fiadeiro, J. L.: Interconnecting Objects via Contracts. Proceedings of the International Conference on the Unified Modeling Lnaguage. (1999)
2. Andrade, L.F., Fiadeiro, J. L.: Coordination Technologies for Managing Information System Evolution. Proceedings of the International Conference on Computer Aided Software Engineering. (2001)
3. Marcos, C.: Arquitecturas de Software Reflexivas. Technical Report. Instituto de Sistemas Tandil. Facultad de Ciencias Exactas. (1998)
4. Letelier, P., Ramos, I., Sánchez, P., Pastor, O. : OASIS versión 3.0: A formal Approach for Object Oriented Conceptual Modelling. Edited by the Technical University of Valencia. (1998)
5. Lozano, M., Ramos, I., González., P.: User Interface Specification and Modeling in Object Oriented Environment for Automatic Software Development. IEEE 34th International Conference on Technology of OO Languages and Systems. (2000)
6. Myers, B. A.: Why are Human-Computers Interfaces Difficult to Design and Implement. Tech. Report CMU-CS-93-183, CMU. (1993)
7. Myers, B. A.: User Interface Software Tools. ACM Transactions on Computer-Human Interaction 2. (1995)
8. Paternò, F.: Model-Based Design and Evaluation of Interactive Applications. Springer. (1999)
9. Shneiderman, B.: Designing the User Interface. Strategies for Effective Human-Computer Interaction. Addison Wesley. (1998)