XV Jornadas de Ingeniería del Software y Bases de Datos
JISBD 2006
José Riquelme - Pere Botella (Eds)

VALIDACIÓN Y VERIFICACIÓN DE INTERFACES DE USUARIO EN EL ÁMBITO DEL DESARROLLO BASADO EN MODELOS

Miguel Romero, Juan de Lara

Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Ciudad Universitaria de Cantoblanco, 28049, Madrid
e-mail: {m.romero, jdelara}@uam.es

Palabras clave: Interfaces de Usuario, Desarrollo basado en Modelos, Redes de Petri Coloreadas, UsiXML.

Resumen. *En este trabajo, se presenta un enfoque para la verificación del diseño de interfaces de usuario. Nuestra aproximación se centra en el desarrollo basado en modelos propuesto por UsiXML, que considera cuatro niveles de abstracción: definición de tareas y conceptos, interfaz abstracta, concreta y final. Hemos añadido un proceso de validación y verificación basado en redes de Petri a esta arquitectura. La idea es transformar el modelo de interfaz concreta a redes de Petri coloreadas para su análisis con la herramienta CPNTools. La red de Petri obtenida permite verificar propiedades del diseño de la interfaz de usuario tales como ventanas no alcanzables, número máximo de ventanas abiertas a la vez, acciones no realizables, deadlocks y análisis de alcanzabilidad de ciertas configuraciones.*

1. INTRODUCCIÓN

Los ordenadores personales y las aplicaciones interactivas, han hecho el diseño de la interfaz de usuario un ingrediente esencial en la calidad final de la aplicación [1]. La filosofía MBUI (*Model-Based User Interfaces*) estudia el diseño basado en modelos de interfaces de usuario [2]. Propone una serie de modelos de distinto nivel de abstracción, que incluyen modelos de dominio, de presentación, de tareas, del diálogo, del usuario y de la plataforma. UsiXML [3] es uno de los lenguajes que sigue esta filosofía. Es un estándar abierto, y se alinea con MDA, proponiendo una serie de metamodelos para cada uno de los distintos tipos de diagramas.
A medida que las aplicaciones y la interacción con ellas se hacen más complicadas, son necesarios mecanismos automáticos para validar y verificar la corrección del diseño de sus interfaces. Por ejemplo, serían interesantes métodos para identificar si la navegación por las distintas ventanas es la correcta, si los distintos tipos de usuario pueden realizar las acciones disponibles para ellos, si la aplicación permite volver al punto inicial, si es posible alcanzar cierta posibilidad de interacción, si hay *deadlocks*, etc.
En el ámbito del desarrollo basado en modelos (DBM), la mayoría de las metodologías

carecen de un proceso explícito de validación (¿se ha construido el sistema correcto?) y verificación (¿se ha construido el sistema correctamente?) (V&V) [4]. La verificación de modelos que garantice su corrección es de especial relevancia en el DBM (y en particular en el desarrollo de interfaces de usuario basadas en MBUI) ya que los modelos son el activo principal, y en última instancia el código de la aplicación se genera a partir de ellos.
Las redes de Petri coloreadas (*CPN – Coloured Petri Nets*) [5] es un formalismo muy adecuado para describir concurrencia, sincronización y causalidad y es utilizado para modelar, analizar y realizar prototipos de sistemas dinámicos y concurrentes. Dos propiedades las hacen interesantes para la especificación: son base para la simulación y para la verificación formal.
En este artículo presentamos una propuesta concreta y novedosa para añadir un proceso de V&V al desarrollo de interfaces de usuario basado en modelos propuesto por UsiXML. Nuestro enfoque consiste en transformar el modelo de interfaz de usuario concreta a redes de Petri coloreadas para su análisis en la herramienta CPNTools [6].
Nuestro trabajo es una mejora de [7]. En él, se presenta una red de Petri diseñada para obtener ciertos valores de usabilidad sobre la navegación entre interfaces (reversibilidad de acciones, acceso a puntos principales de la secuencia de interacción, etc). Su trabajo está orientado a interfaces industriales por las características que éstas presentan como una gran carga de datos, condiciones de seguridad y preparación para interacciones urgentes. Nuestro trabajo es una mejora, ya que nos basamos en un lenguaje de descripción de interfaces como UsiXML, la red puede simular todo tipo de interfaces, definimos un proceso de creación de interfaces completo, se automatizan todas las tareas y combinamos la simulación con la ejecución (guiada por la simulación) de las interfaces en el lenguaje final. Además, en la simulación se pueden incluir todas las características visuales con las que se definió la interfaz gracias al modelo de interfaz de usuario concreta (colores, tamaños,…).
El artículo se ha estructurado de la siguiente manera. La sección 2 da una introducción al desarrollo MBUI con UsiXML. La sección 3 introduce de manera informal las redes de Petri coloreadas. La sección 4 presenta la arquitectura que proponemos. La sección 5 muestra un ejemplo, y finalmente la sección 6 concluye con las conclusiones y el trabajo futuro.

2. DESCRIPCIÓN BASADA EN MODELOS DE INTERFACES DE USUARIO CON USIXML

UsiXML (*User Interface eXtensible Markup Language*) es un lenguaje de descripción de interfaces de usuario (*User Interface Description Language*, UIDL) basado en XML. Puede describir interfaces para múltiples contextos de uso, con independencia de instrumento, de plataforma y de modalidad. Está basado en los principios de modelado MDA, así como en el marco de referencia *Camaleon* [8], un proceso MBUI que define los pasos y tipos de modelos en el desarrollo de interfaces de usuario para aplicaciones interactivas multicontexto.
En particular, se proponen cuatro tipos de modelos, cuya jerarquía se muestra en la Figura 1. El *modelo de tareas y dominio* sirve para definir las tareas que se quieren realizar mediante la interfaz (similar a los casos de uso), así como conceptos orientados al dominio requeridos por las tareas. El modelo de interfaz de usuario abstracta (AUI, *Abstract User Interface*) se utiliza

para definir contenedores abstractos y componentes individuales agrupando en estos la ejecución de las tareas antes definidas. El modelo de interfaz de usuario concreta (CUI, *Concrete User Interface*) materializa la definición abstracta anterior en cada contexto de uso, mediante *widgets* y navegación entre interfaces. Finalmente la interfaz de usuario final (FUI, *Final User Interface*) contiene la interfaz de usuario operacional, generada a partir de las capas anteriores.
Así pues, este lenguaje posibilita tener dos procesos alternativos principales de elaboración de interfaces de usuario, que se muestran en la Figura 1. El primer proceso se ha etiquetado mediante *1.a* y *1.b*. En este proceso, mediante herramientas gráficas como *GrafiXML* [3] se define un modelo CUI, pudiendo además generar el código final FUI. El segundo proceso se ha etiquetado mediante *2.a*, *2.b*, *2.c* y *2.d* y mediante herramientas de modelado y transformaciones, completa todo el ciclo de desarrollo y los respectivos modelos.

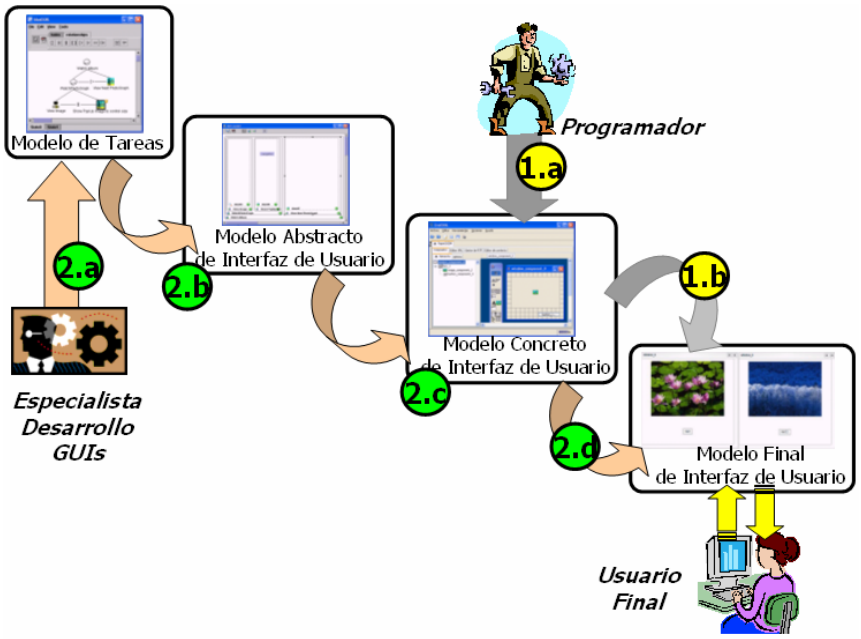

Figura 1. Jerarquía de modelos de UsiXML, y procesos en su construcción.

La elección de uno u otro de los dos procesos depende de quién es el creador de la interfaz y qué importancia se le quiere dar al proceso de definición la misma. Si se trata de un desarrollo rápido y la interfaz la está desarrollando un programador, se puede utilizar un entorno como GrafiXML, donde además de crear una interfaz, implícitamente se está creando el modelo CUI de UsiXML (paso 1.a en la Figura 1). Posteriormente al desarrollo, sólo se tendrá que generar la interfaz final (FUI, paso 2.a) mediante esta misma herramienta.
Si se trata de un desarrollo dedicado y dirigido por especialistas en definición de interfaces y usabilidad, se puede ir mucho más allá y entran en juego varias clases de herramientas. Mediante *IdealXML* [3] se describen las tareas que los usuarios realizan usando la aplicación además de cómo se relacionan unas con otras (paso *2.a* en la Figura 1). Se capturan las tareas

de los usuarios y los comportamientos del sistema respecto a estas, creando de esta manera el modelo de tareas y dominio. También utilizando *IdealXML* se genera mediante transformación de grafos parte del modelo AUI de la interfaz de usuario (*2.b*), que posteriormente el diseñador puede completar. Mediante *TransformiXML* [3], partiendo del modelo AUI de la interfaz con una serie de reglas de transformación se consigue el modelo CUI (*2.c*) que representa la interfaz en uno o varios contextos con el nivel de interacción ya concretado. Mediante herramientas como GrafiXML y FlashiXML partiendo del modelo CUI, se puede generar el código final de la interfaz en diferentes lenguajes de programación, alcanzando de esta manera el modelo FUI (paso 2.d).

3. REDES DE PETRI COLOREADAS

En esta sección damos una descripción intuitiva e informal de las Redes de Petri. Para una descripción más extensa, el lector puede consultar por ejemplo [9][5]. Las redes de Petri [9] son un lenguaje gráfico y formal para la descripción, simulación y análisis de sistemas dinámicos discretos. Son adecuadas para modelar sistemas distribuidos y concurrentes, y han sido usadas con éxito por ejemplo en el diseño de protocolos de comunicación, diseño de sistemas operativos, diseño de hardware, sistemas embebidos, diseño de software y modelado del proceso de negocio.

Una red de Petri es un grafo bipartito, formado por dos tipos de nodos: lugares (*Places*) y transiciones. Los lugares se pueden conectar con las transiciones (y viceversa) por medio de arcos. Los lugares conectados con un arco a una transición son sus lugares de entrada. Los lugares que reciben un arco de una cierta transición, son sus lugares de salida. Los lugares contienen cero o más *tokens*. El número de tokens de cada uno de los lugares se conoce como el "*marcado de la red*", y define unívocamente su estado.

La simulación de una red de Petri consiste en el disparo de transiciones habilitadas. Una transición está habilitada si todos sus lugares de entrada contienen al menos un token. Si una transición se dispara, se elimina un token de cada uno de los lugares de entrada, y se añade un token a cada uno de sus lugares de salida.

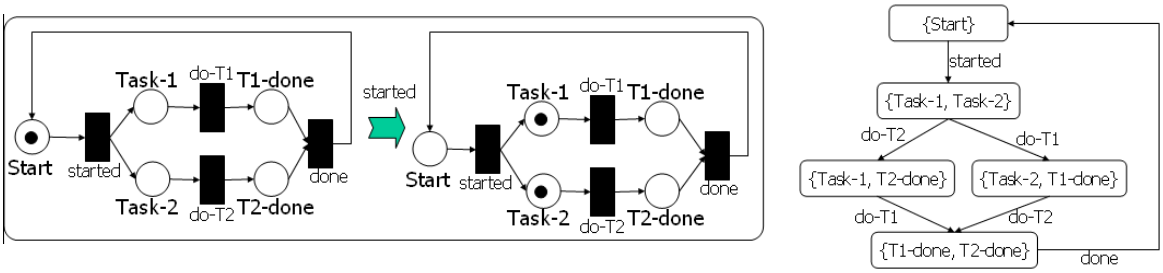

Figura 2. Disparo de una transición (izquierda). Grafo de alcanzabilidad de la red (derecha)

La izquierda de la Figura 2 muestra un modelo que representa un proceso simple formado por dos tareas (*Task-1* y *Task-2*). Una vez empezado el proceso (transición *started*), ambas se pueden realizar en cualquier orden (primero *do-T1* y luego *do-T2* o al revés). Una vez

concluido el proceso (transición *done*), se puede empezar de nuevo. La parte izquierda muestra el disparo de la transición *started*, que hace que se consuma el token del lugar *Start* y se creen dos tokens en *Task-1* y *Task-2*. La derecha de la Figura 2 muestra un grafo con todas los posibles estados de la red. Este grafo se llama "*grafo de alcanzabilidad*" y sólo se puede calcular de manera exacta si el número de tokens que puede haber en la red está acotado [9]. Este grafo nos permite conocer propiedades del modelo tales como si es posible alcanzar una configuración a partir de otra (así como la(s) secuencia(s) de disparos que nos llevan a ella), si hay deadlocks, si la red es reversible (si podemos volver al estado inicial), etc.
Las redes de Petri coloreadas [5] extienden la descripción anterior añadiendo estructura (similar a los registros de Pascal) a los tokens, así como la posibilidad de partir una red en páginas para mejorar su comprensibilidad. El valor estructurado de los tokens se conoce como "*color*". Éste puede ser seleccionado por los arcos, evaluado por las transiciones, y cambiado por su disparo.

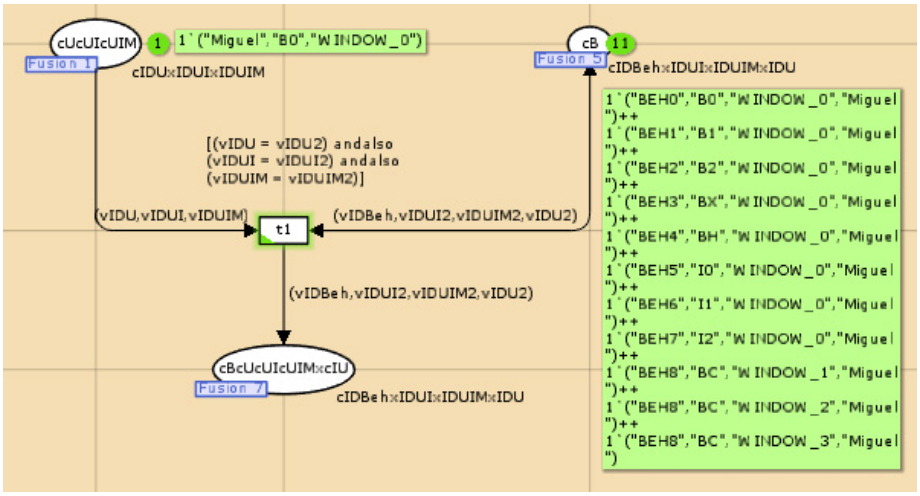

Figura 3. Un ejemplo de CPN, que es una parte de la red de simulación de UsiXML

La Figura 3 presenta una página de una CPN que simula una interfaz de usuario. El lugar *cUcUIcUIM* representa el objeto de interfaz concreto (CIO, *Concrete Interface Object*) de UsiXML con el que un usuario está interaccionando. El token que contiene representa mediante su color (tres cadenas de texto) al usuario *Miguel* que ha interaccionado con el objeto *B0* (el identificador de un botón) de la ventana con identificador *WINDOW_0*. El lugar *cB* representa los comportamientos definidos para cada CIO y para cada usuario de la aplicación. La transición *t1* está habilitada ya que es posible seleccionar un token de los dos lugares de entrada que hacen que la condición sea verdadera. En particular el token ("Miguel", "B0", "WINDOW_0") y el ("BEH0", "B0", "WINDOW_0", "Miguel") pueden disparar la transición ya que las tres primeras componentes del token de *cUcUIcUIM* son iguales a la cuarta, segunda y tercera componentes del token de *cB*. Al dispararse *t1* se crea un token nuevo con otro color (determinado por la inscripción del arco de salida) que indica que la interacción con ese CIO viene definida en el comportamiento *BEH0*. Además, el token con

el comportamiento seleccionado se devuelve al place *cB*. Obsérvese que cuando existen arcos de entrada y salida a un place con la misma inscripción, estos se representan como un arco bidireccional.
CPNTools [6] es una herramienta que permite la definición, simulación y análisis de CPNs. Una vez construida la red, se pueden realizar simulaciones interactivas, automáticas o mezcla de ambas. Las simulaciones interactivas permiten elegir la transición a disparar, así como el conjunto de tokens a consumir. Este tipo de simulación se suele emplear como medio de *debugging* y para comprobar casos concretos de simulación. Las simulaciones automáticas permiten analizar la red, por ejemplo mediante el cálculo del grafo de alcanzabilidad. Finalmente, las simulaciones mixtas permiten monitorizar la ejecución. Esto es, se pueden crear agentes o monitores programados para que graben el estado de la red (o parte de él) cuando se cumplan una serie de condiciones.

4. ARQUITECTURA PROPUESTA

Nuestra propuesta parte del modelo CUI de UsiXML, y pretende simular y analizar dicho modelo mediante CPNTools. Para ello, se ha creado una CPN (capaz de representar cualquier interfaz de UsiXML), cuyo marcado se obtiene a partir del modelo CUI. El funcionamiento de esta red se muestra en el esquema de la Figura 4.

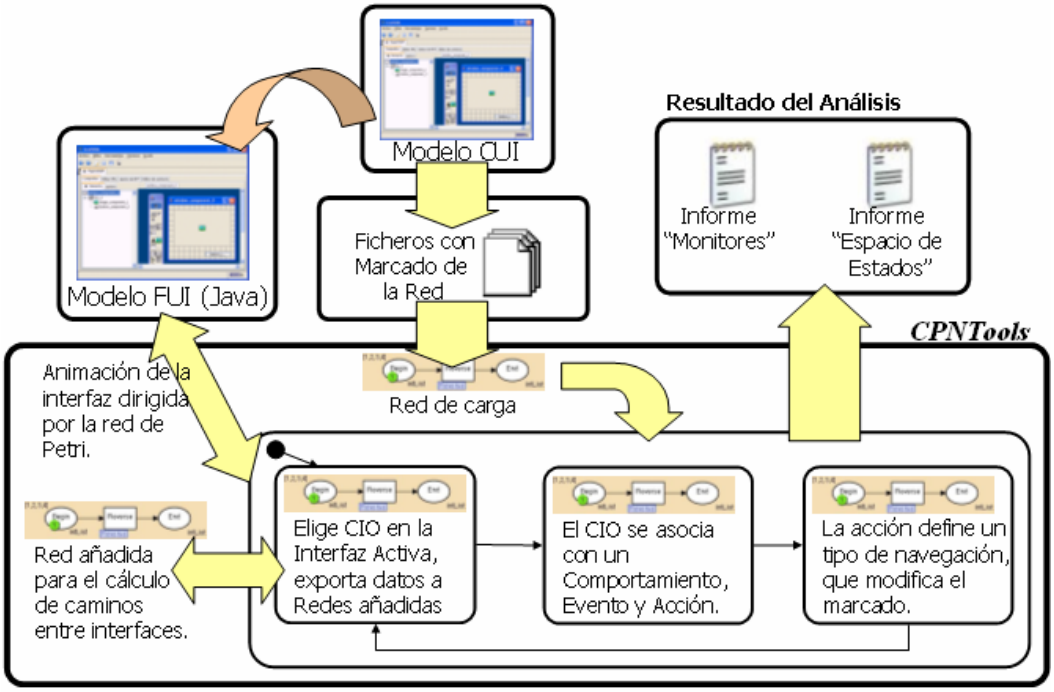

Figura 4. V&V de una interfaz de usuario UsiXML mediante CPNTools.

Una página de la red (que llamamos *red de carga*) se encarga de la carga del marcado inicial,

obtenido a partir del CUI. Una vez cargado el marcado, primero se elige un CIO para la simulación de la interfaz inicial (esto representa que el usuario interacciona con él). A la vez, se exportan los datos de esta elección y de la interfaz a páginas con redes adicionales para estudios concretos. Segundo, este CIO se relaciona con su correspondiente *Comportamiento*, *Evento* y *Acción* que son características del modelo CUI de UsiXML y que se utilizarán para observar las interacciones que se realizan en la interfaz. Tercero, se ejecuta la acción asociada a la interacción que se está ejecutando. En UsiXML las acciones vienen definidas mediante *transiciones*, *métodos*, o *transformaciones*. En cualquiera de los casos, una acción provoca cambios en la interfaz y una posible navegación. Esta navegación ocasiona un cambio de la ventana en la que se está interaccionando, que puede ser la creación de otra ventana superpuesta, el cierre de la ventana actual y la apertura de otra o la aparición de interfaces diálogo. Estos tres pasos se han mostrado en la parte inferior derecha de la Figura 4 y se han modelado en 3 páginas distintas de la CPN, que puede entenderse como una máquina virtual para la ejecución de modelos CUI.

CPNTools permite calcular el grafo de alcanzabilidad de la CPN (total o parcialmente, según la interfaz a simular) y el de componentes conectados. Ambos son presentados al usuario mediante un informe. Se pueden generar otros informes mediante la monitorización de puntos concretos de la red y mediante redes añadidas creadas por el diseñador para realizar cálculos personalizados. En el caso de nuestra arquitectura se ha creado una red añadida que calcula los diferentes caminos que existen para llegar de una interfaz origen a una final. Además CPNtools permite comunicar la simulación con un lenguaje final como Java mediante una serie de librerías (*javaCPN*). De esta manera, teniendo en cuenta que herramientas de UsiXML generan código ejecutable para la interfaz a partir del modelo CUI, se podría llevar a cabo esta asociación observando la simulación del modelo en la interfaz final. Actualmente, estamos trabajando todavía en la implementación de la conexión mediante javaCPN.

Para llevar a cabo la simulación de nuestra CPN, hemos usado la simulación automática y el estudio de los informes que se generan. Estos ofrecen información sobre estadísticas, acotación (número máximo y mínimo de tokens y distribución de éstos), alcanzabilidad (posibilidad de alcanzar marcados de la red desde todos los demás estados), vivacidad (puntos donde la red termina o transiciones que no se ejecutan) y "*fairness*" (frecuencia con la que se disparan transiciones). Con esta información podemos analizar la interfaz para obtener por ejemplo, estados donde la aplicación provoca un fallo no controlado (mediante los "*Dead Markings*"), si se han ejecutado o no ciertas navegaciones, número de ventanas abiertas a la vez, bucles, deadlocks, etc. Esta información resulta útil para el diseñador de la interfaz, que puede decidir modificar el modelo CUI (en el caso del proceso "1" de la Figura 3), o bien modelos de más alto nivel de abstracción, como el AUI, e incluso el modelo de tareas (en el caso del proceso "1" de la Figura 3). La sección siguiente muestra un ejemplo concreto de uso de la arquitectura propuesta.

5. EJEMPLO

Supongamos que deseamos analizar la interfaz CUI de la izquierda de la Figura 5. Esta

consiste en una ventana (*Container* en UsiXML) llamada "*window_0*" con un botón y una imagen. Si se pulsa el botón, se cambian los CIOs dentro la misma ventana y si se pulsa sobre la imagen, ésta se despliega en otro container (llamado *window_1*, *window_2* o *window_3*), y deja de soportar interacción. Se puede cerrar el nuevo container con un botón ("X" en la figura, que se llama "BC" en el modelo CUI). En el contenedor *window_0* existen definidos CIOs de ayuda (botón "?", llamado "BH") y de salida de la aplicación ("X", llamado "BX").

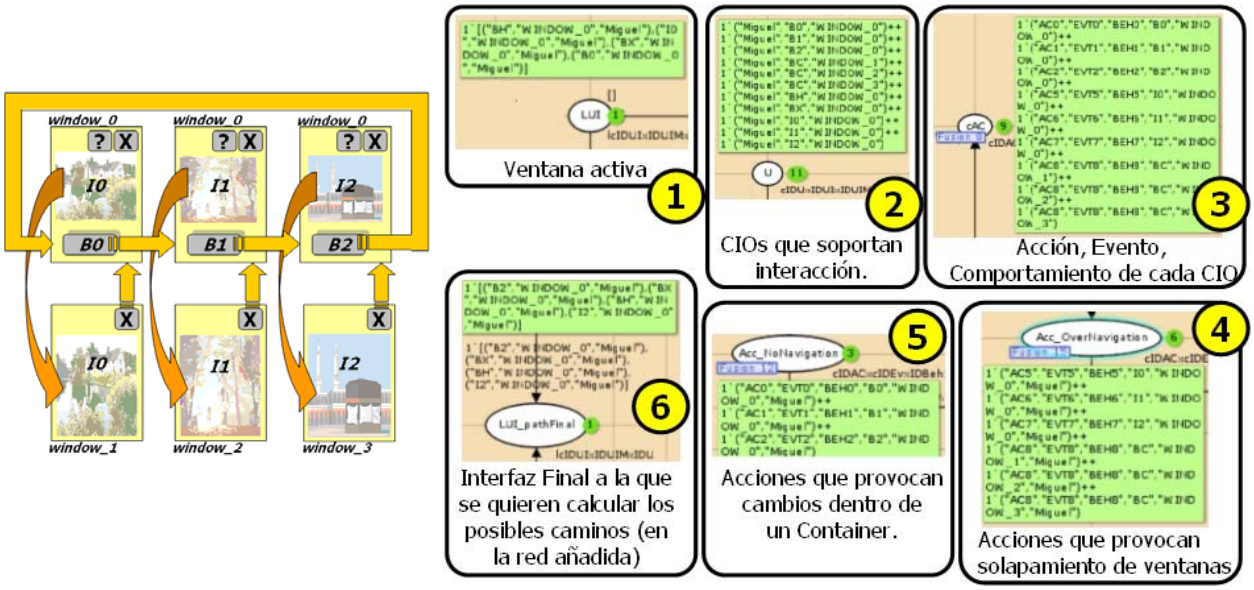

Figura 5. Interfaz que se quiere analizar (izquierda). Marcado inicial (derecha, cuadros 1-5) y Estado a alcanzar (cuadro 6).

A partir del modelo CUI de la interfaz (no mostrado por falta de espacio), obtenemos el marcado de la red. En los cuadros 1-5 de la Figura 5, se muestran algunas zonas de la red con el marcado inicial. El cuadro 1 contiene información sobre los CIOs de la ventana activa inicial. El cuadro 2 muestra los CIOs que soportan interacción en toda la interfaz. El cuadro 3 tiene las asociaciones de acciones a cada evento que pueda soportar cada CIO. El marcado inicial también contiene la semántica de cada acción. El cuadro 4 muestra las acciones que provocan la apertura o cerrado de una ventana. En el cuadro 5 se encuentran las acciones que no provocan navegación, sino un cambio en los CIOs dentro de la ventana activa.

Para realizar el análisis, generamos el informe "*Espacio de Estados*", que se muestra parcialmente en la Figura 6. La sección sobre estadísticas ("Statistics") informa sobre el grafo de alcanzabilidad. El atributo "*Status*" indica si el grafo se ha calculado completa ("*Full*") o parcialmente ("*Partial*"). En el primer caso, la interfaz no tiene problemas de navegación: no hay deadlock, ciclos, ni interacciones que lleven la interfaz a un error. Si el estado es "*Partial*", la interfaz tiene algún problema, que habrá que estudiar en otras partes del informe. La sección sobre acotación (no mostrada en la figura), indica el número máximo y mínimo de tokens, así como el contenido de éstos. Esta es información importante a la hora de saber

número de CIOs, comportamientos, acciones, eventos, caminos generados, etc. Así, se pueden obtener desde estadísticas sobre la complejidad de la interfaz que se está simulando hasta fallos de definición por la necesidad de algún token que falta.

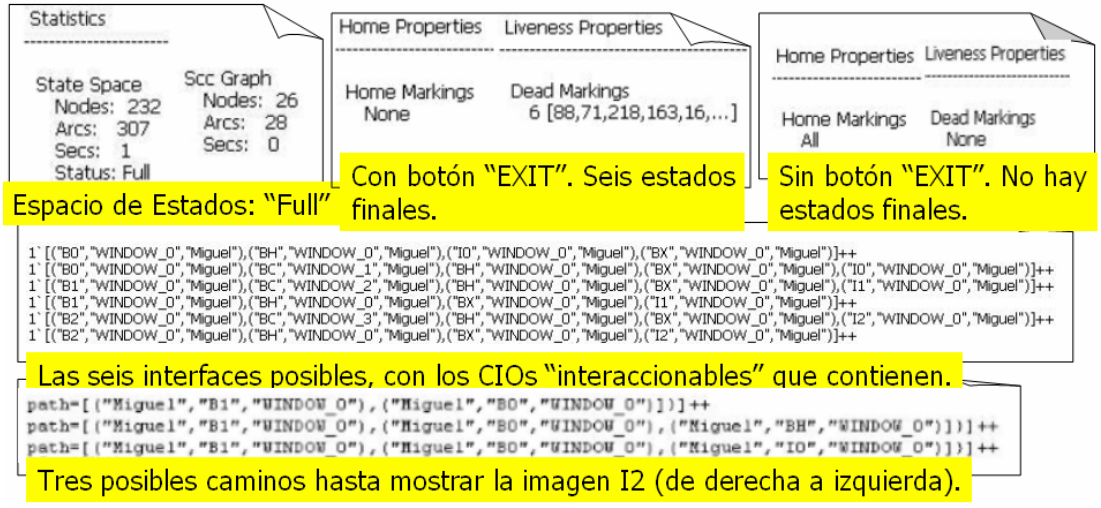

Figura 6. Resultados del Análisis

La sección de alcanzabilidad ("*Home Properties*" en la Figura 6) indica si un cierto estado puede ser alcanzado desde todos los otros. En el ejemplo, existen 6 estados diferentes de la interfaz en los que se puede salir. Además, se puede acceder al estado de la interfaz en ese momento, consultando en el estado del grafo los diferentes estados [88,71,..]. El "*Home Marking*", en este caso, no es representativo ya que el botón "*exit*" provoca paradas en la simulación de la red. Si a modo de ejemplo se elimina el botón "*exit*", tendemos que todos los estados pueden alcanzar todos los estados de la red, es decir, de cualquier interfaz se puede llegar a cualquier otra. Resultados intermedios (p.e. *HomeMarking=5*, *DeadMarking=2*), suponiendo que se han quitado los botones para salir de la aplicación, indican situaciones de deadlock de la aplicación por algún error en su definición.

La Figura 6 también muestra seis posibles interfaces con sus CIOs activos (es decir, sólo se muestran los que admiten interacción). Estos son los seis posibles estados de la interfaz durante la simulación, y se corresponden con los 6 estados mencionados en el párrafo anterior. Mediante esta información se puede controlar por ejemplo el número de ventanas abiertas a la vez durante la ejecución, lo cual puede interesar desde el punto de vista de la usabilidad. En la parte inferior de la figura hemos incluido algunos de los posibles caminos de navegación (que se han generado con la red añadida) desde el estado inicial hasta que la interfaz que muestra la imagen *I2* (el marcado de dicha interfaz se muestra en el cuadro 6 de la Figura 5).

Además del análisis anterior, existe la posibilidad de visualizar la simulación en la interfaz final como forma de validación (en implementación). De esta manera, se pueden comprobar otras propiedades relacionadas con la usabilidad como por ejemplo, el color de ciertos CIOs.

6. CONCLUSIONES Y TRABAJO FUTURO

En este artículo hemos presentado un enfoque de V&V de interfaces de usuario mediante su transformación a redes de Petri coloreadas. Nuestra propuesta se adapta a los procesos de desarrollo basados en MBUI, y en concreto a la propuesta del lenguaje UsiXML. En particular, se transforma el modelo CUI a un marcado particular para una red de Petri que es fija. Para el análisis de la red hemos utilizado la herramienta CPNTools.
Estamos trabajando en la implementación de la parte de validación mediante la conexión del modelo FUI a la red de Petri. Además, se pretende mostrar al diseñador de la interfaz los resultados del análisis de manera más entendible, en el contexto de los modelos origen (UsiXML) y no de los modelos de análisis (CPN). También se estudia el diseño de un lenguaje de alto nivel para expresar requisitos de usabilidad, que puedan traducirse a una red añadida para su verificación.

Agradecimientos: Este trabajo ha sido subvencionado en parte por el Ministerio de Educación y Ciencia, proyecto MOSAIC (TSI2005-08225-C07-06). Los autores quieren agradecer además los comentarios y sugerencias de los revisores.

REFERENCIAS

[1] Shneiderman, B., Plaisan, C. *Designing the User Interface*. Addison-Wesley (2005).
[2] Puerta, A.R., and Szekely, P. Model-Based Interface Development. CHI'94, Boston (1994). Disponible en: http://www.arpuerta.com/pdf/chi94.pdf
[3] Página web de UsiXML: http://www.usixml.org
[4] Lucas, F. J., Molina, F., Toval. A. *Una propuesta de V&V en el marco de MDA*. II Taller sobre Desarrollo Dirigido por Modelos. MDA y Aplicaciones. Granada. Vol. 157 de CEUR Workshop Proceedings (2005).
[5] K. Jensen: Coloured Petri Nets: A High-level Language for System Design and Analysis. In: G. Rozenberg (ed.): Advances in Petri Nets 1990, Lecture Notes in Computer Science Vol.483, Springer-Verlag 1991, 342-416.
[6] Página web de CPNTools: http://wiki.daimi.au.dk/cpntools/cpntools.wiki
[7] Turnell, M. de F. Q. V., Scaico, A., de Sousa, M. R. F., Perkusich, A.. *Industrial User Interface Evaluation Based on Coloured Petri Nets Modelling and Analysis*. 8th International Workshop, DSV-IS, Glasgow, LNCS 2220, Springer. (2001).
[8] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. "A Unifying Reference Framework for Multi-Target User Interfaces". *Interacting with Computers*. Vol. **15** (3), pp. 289-308 (2003).
[9] Peterson, J.L. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, INC., Englewood Cliffs, N.J (1981).