Prototipado rápido de interfaces de usuario

Pedro J. Molina*
Sofia Martí
Óscar Pastor†

Enviado a IDEAS'2002

Resumen

Se propone un método de inferencia para favorecer el prototipado rápido de interfaces de usuario. Se logra un incremento en la productividad de desarrollo usando técnicas de producción automática de software sobre un modelo conceptual extendido para soportar información de interfaz de usuario.

1 Introducción

En el mundo de la industria, existe una necesidad clara de prototipado rápido que permita la pronta validación de los requisitos con el usuario. Por otra parte, este prototipo debe ser lo más completo posible para que la validación realizada por el usuario sea útil. Estos dos puntos están contrapuestos, ya que si se prototipa rápidamente, el prototipo es incompleto y si se crea un prototipo completo no puede hacerse rápidamente. Proponemos un proceso denominado *Inferencia*, que permite incrementar la velocidad de prototipado sin incrementar el esfuerzo destinado a completar el modelo conceptual.

Compenzaremos presentando brevemente la metodología sobre la que se asienta dicho proceso.

1.1 Metodología

OO-Method [8] es una metodología utilizada para la creación de modelos conceptuales. Consta de diagramas gráficos a la UML [3, 4] pero a la vez esta soportado por un lenguaje formal de especificación de sistemas orientado a objetos: OASIS [7]. Esta compuesta por los siguientes modelos:

*P.J. Molina y S. Martí ({pjmolina|smarti}@care-t.com) trabajan en el Departamento de I+D de CARE Technologies S.A., España.

†O. Pastor (opastor@dsic.upv.es) es profesor titular en el Departamento de Sistemas de Información y Computación de la Universidad Politécnica de Valencia, España.

- **Modelo de Objetos.** Modelo gráfico en el cual se definen las clases, sus atributos y las relaciones entre clases.
- **Modelo Dinámico.** Modelo gráfico que permite especificar las vidas válidas de los objetos de las clases y de sus interacciones.
- **Modelo Funcional.** Se utiliza para capturar la semántica asociada a los cambios de estado de los objetos provocados por los eventos (modificación de los atributos de la clase).
- **Modelo de Presentación [9, 10, 11, 12].** Modelo para especificar requisitos de interfaz de usuario que comprende requisitos sobre presentación, navegación, visibilidad, accesos y búsqueda, y que está basado en el uso de patrones.

El proceso de *Inferencia* al que nos referimos se realiza sobre el Modelo de Presentación y consiste en derivar la información de inferfaz de usuario faltante permitiendo prototipar rápidamente sin la necesidad de especificar por completo el Modelo de Presentación[1].

A continuación se introducirán los conceptos del Modelo de Presentación para pasar a describir los procesos de inferencia aplicados en cada uno de los elementos. Un pequeño caso de estudio mostrará el proceso de inferencia en práctica y su aplicación industrial. Finalente se presentarán los trabajos relacionados, las conclusiones obtenidas y la bibliografía empleada.

2 Modelo de Presentación

El Modelo de Presentación consta de diveros conceptos y patrones. El uso combinado de estos elementos permite la construcción completa de una especificación abstracta de interfaz de usuario. Estos conceptos se presentan en niveles[12] (véase Figura 1):

- *árbol de jerarquía de acciones*: define el acceso a la funcionalidad el sistema.
- *unidades de interacción*: presentan al usuario escenarios donde se produce el diálogo con el sistema.
- *patrones auxiliares*: funcionan como piezas base para crear las unidades de interacción y complementan a estos aportando semántica precisa.

2.1 Árbol de jerarquía de acciones

El árbol de jerarquía de acciones[2] permite especificar un árbol que proporciona el acceso a la funcionalidad de una vista. Está compuesto de nodos intermedios

[1] En el caso extremo, permite crear un prototipo sin Modelo de Presentación de partida, infiriéndolo por completo.

[2] En lo sucesivo lo referenciaremos por su acrónimo: AJA.

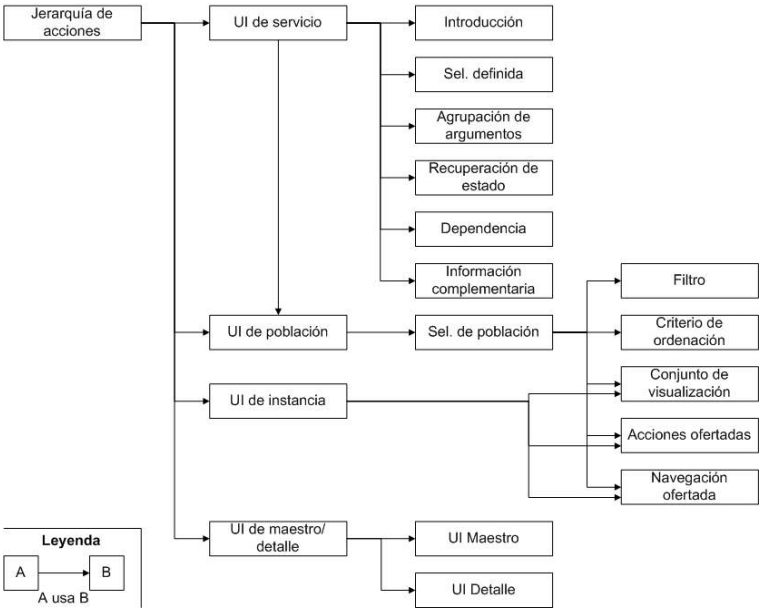

Figura 1: Relaciones de uso de patrones y conceptos en el Modelo de Presentación.

que actúan como meras etiquetas agrupadoras y nodos hoja que apuntan a unidades de interacción.

2.2 Unidad de interacción

Una unidad de interacción es la abstracción de una ventana, formulario, página Web, etc. Es un escenario particular para un usuario que le permitirá interactuar con el sistema. Existen cuatro tipos definidos:

Unidad de interacción de servicio Diálogo que permite al usuario proporcionar valores a los argumentos, necesarios para la correcta ejecución de servicios.

Unidad de interacción de instancia Diálogo que abstrae la presentación de un objeto. Está compuesto de los siguientes elementos: un conjunto de visualización con los datos a observar (*¿qué vemos?*), un conjunto de navegación ofertada (*¿dónde navegamos?*) y un conjunto de acciones ofertadas (*¿qué podemos hacer?*).

Unidad de interacción de población Diálogo centrado en la presentación de un conjunto de objetos y acciones sobre ellos. Está compuesto por los siguientes elementos: filtros (*¿cómo restringimos?*), criterios de ordenación (*¿cómo quedan ordenados?*), conjuntos de visualización que definen

los datos a observar (*¿qué vemos?*), un conjunto de navegación ofertada (*¿dónde navegamos?*) y un conjunto de acciones ofertadas (*¿qué podemos hacer?*). Esta unidad se utiliza tanto para seleccionar objetos para argumentos objeto-valuados, como para búsqueda de objetos sobre los que realizar acciones.

Unidad de interacción maestro/detalle Representa un diálogo compuesto a partir de otros con la semántica de cabecera-despliegue. Está formado por: un componente maestro (una unidad de interacción) y uno o más componentes detalles (a su vez, unidades de interacción).

2.3 Patrones auxiliares

Los patrones auxiliares son los siguientes:

Patrón de introducción Captura los aspectos relevantes de los tipos de datos que van a ser introducidos o visualizados por el usuario: formato con el cuál se introducirá el valor de un campo, el valor por defecto a asignar, rango posible de valores, etc.

Patrón de selección definida Proporciona por enumeración los valores válidos para un argumento. El conjunto definido de este modo, se comporta como un tipo enumerado definido por el analista que restringe el valor de los argumentos a los que se aplica a los valores especificados.

Patrón de información complementaria Captura información extra sobre el estado del objeto que debe ser mostrada al usuario para complementar al identificador de un objeto. Esta información de retroalimentación ayuda al usuario a confirmar que la selección realizada fue correcta.

Patrón de dependencia Describe las relaciones de dependencia que existen entre los argumentos de un servicio. Estas dependencias permiten dar un valor inicial a los argumentos en función de una condición o fijarlos para que no puedan ser modificados.

Patrón de agrupación de argumentos Permite agrupar los argumentos de un servicio utilizando un criterio definido por el analista, de forma que se reúnen bajo un mismo grupo los argumentos que tienen cierta relación entre ellos.

Filtro Permite construir la condición de búsqueda de la población de una clase para su posterior visualización.

Conjunto de visualización Consiste en un conjunto ordenado de atributos que se pueden observar de una clase. El conjunto de visualización permite al usuario observar el estado de un objeto.

Navegación ofertada Lista ordenada de unidades de interacción, donde cada elemento de navegación está etiquetado con la relación atravesada, bien

sea de agregación o de herencia, y la unidad de interacción destino. Se emplea para consultar objetos relaciondados con uno dado.

Acciones ofertadas Lista ordenada de unidades de interacción que expresa las acciones que pueden llevarse a cabo y conducen a una nueva unidad de interacción. Se emplea para ofertar acciones conducentes a modificar el estado del objeto.

Criterio de ordenación Conjunto ordenado de atributos de una clase, indicando para cada atributo orden ascendente o descendente. Este criterio permite indicar el orden en el que serán mostradas las instancias tras una búsqueda.

3 Inferencia

Durante la especificación de un esquema conceptual, el analista modela de forma incremental la información necesaria para la obtención de la aplicación final. Inicialmente se detalla la parte esencial (estática y dinámica básica) y posteriormente se completa de forma iterativa (dinámica compleja y presentación). El proceso de inferencia está constituido por un conjunto de heurísticos que completan la información que el analista no ha especificado en el modelo utilizando la información disponible en el modelo. El proceso de inferencia persigue dos objetivos básicos:

- permitir la obtención de prototipos de interfaz de usuario ejecutables sin la introducción de todos los datos del Modelo de Presentación, y
- beneficiar el *caso más frecuente*, es decir, que el analista especifique únicamente aquellos casos que requieran de cierta funcionalidad especial. El resto de casos con funcionalidad *estándar* se inferirán correctamente y no será necesario especificarlos, se asumirán como comportamiento por defecto.

El proceso de inferencia recibe como entrada un esquema conceptual donde el modelo de presentación está especificado parcialmente o incluso, no ha sido especificado en absoluto. El proceso de inferencia produce como salida un modelo de presentación, en el que se han inferido nuevos componentes y completado datos en los existentes.

A continuación se detallará los distintos casos donde se aplican los procesos de inferencia.

3.1 Inferencia de vistas

El prototipo que se le presentará al usuario debe estar siempre basado en una vista. Una vista está compuesta por un conjunto de interfaces entre una clase servidora y una clase agente, donde se indican los atributos que la clase agente

puede visualizar, los servicios que puede ejectuar y las relaciones de agregación por las que puede navegar.

Si no se especifica una vista con la que obtener el prototipo, se inferirá una vista por defecto que contiene todas las clases y todas las interfaces definidas en el modelo. Esta vista se usará para la posterior obtención del AJA.

En la vista hay información explícita sobre los servicios que se podrán ejecutar (interfaz), por lo que se conoce cuáles son las UI de Servicio que se le ofrecerán al agente conectado. Por el contrario, no se define información explícita sobre el resto de unidades de interacción, por lo que se presentarán en la vista todas aquellas que aparezcan en el AJA más las que sean accesibles desde éstas, es decir, desde argumentos de servicios, variables de filtro, navegación por agregación o herencia y desde acciones.

3.2 Inferencia del árbol de jerarquía de acciones

Si no se ha definido un AJA debe inferirse uno por defecto, que permita al usuario el acceso a todas las unidades de interacción a las cuales tiene acceso desde el escenario principal de la aplicación.

Una vez se tiene una vista se pasa a la inferencia del árbol. Para ello, se crea la raíz del árbol (*nivel 0*) y a primer nivel se añade un nodo por cada una de las clases de la vista etiquetado con el alias efectivo[3] de la clase (*nivel 1*). Por cada nodo intermedio, se crea a su vez tantos nodos hoja como unidades de interacción de la clase existan en la vista, etiquetando cada nodo final con el alias efectivo de la unidad de interacción (*nivel 2*). El orden en el que se añadirán estos nodos hijos será: UI de Instancia, UI de Población, UI de Maestro/Detalle y UI de Servicio.

Para las transacciones globales se añade un nodo de *nivel 1* etiquetado como 'Procesos generales' y como subnodos de éste se agregan las unidades de interacción de servicio de las transacciones globales etiquetadas como las anteriores con el alias efectivo de la unidad de interacción correspondiente.

En el caso en el que existan relaciones de herencia entre clases cada una de las clases descendientes se sitúan en el *nivel 2* incluidas dentro del nodo intermedio creado para su clase ascendiente. De forma recursiva, se añaden sus unidades de interacción y sus clases descendientes.

3.3 Inferencia de unidades de interacción

Los siguientes procesos de inferencia se aplican tanto para crear unidades no definidas en el modelo de presentación como para completar información en las unidades existentes (incluyendo también las unidades inferidas).

[3] El alias efectivo para un elemento es el alias definido sobre el elemento si éste existe, y si no al alias inferido para este elemento.

3.3.1 Inferencia de unidades de interacción de servicio

Si un servicio es ofertable[4] al usuario, aparece en la vista, pero no tiene unidad de interacción asignada, se le crea una unidad de interacción de servicio por defecto, que solicitará al usuario los argumentos declarados en el servicio.

Alias Si una unidad de interacción de servicio no tiene alias definido, se le aplica el alias definido del servicio, y si éste a su vez no existe, entonces se le asigna como alias el nombre del servicio.

Argumentos del servicio

Alias Si un argumento no tiene alias definido se le asigna el alias del atributo del que procede. Si bien no se ha podido inferir el atributo origen del argumento, o bien dicho atributo no tiene alias, se asigna el nombre del argumento.

Patrón de introducción Si un argumento dato-valuado de un servicio no tiene patrón de introducción asociado se le aplica el patrón de introducción de su atributo origen. En su defecto, no se aplica ninguno.

Patrón de selección definida Si un argumento dato-valuado de un servicio no tiene patrón de selección definida asociado se le aplica el patrón correspondiente de su atributo origen. En su defecto, no se aplica ninguno.

Información complementaria Si a un argumento objeto-valuado no se le ha asignado información complementaria, se le aplica la información complementaria por defecto de la clase del tipo del argumento. En su ausencia, no se aplica ninguno.

Unidad de interacción de población Si a un argumento objeto-valuado no se le ha asignado la unidad de interacción destino para selección de objetos, se asigna la unidad de interacción de población por defecto de la clase a la que pertenece el tipo del argumento.

3.3.2 Inferencia de unidades de interacción de instancia

Si una clase pertenece a la vista y no tiene, al menos, una unidad de interacción de instancia, se debe derivar una *por defecto* que estará formada por: un conjunto de visualización inferido, un patrón de navegación ofertada inferido y patrón de acciones ofertadas inferido (los tres componentes será descritos en el apartado 3.4).

Alias Si no tiene alias definido, se aplica como alias el efectivo de la clase[5].

[4] El servicio está incluido en una interfaz entre el usuario y la clase del servicio y dicha interfaz pertenece a la vista a generar.

[5] El alias efectivo de la clase es el alias asignado a la clase y en su ausencia, el nombre de la clase.

3.3.3 Inferencia de unidades de interacción de población

Si para una clase perteneciente a la vista no se especifica ninguna unidad de interacción de este tipo, se infiere una por defecto.

En esta unidad de interacción se añadirán: todos los filtros definidos en la clase, todos los criterios de ordenación definidos en la clase, conjunto de visualización inferido como *conjunto principal*[6] y como conjuntos alternativos el resto de conjuntos de visualización definidos en la clase, navegación ofertada inferida y acciones ofertadas inferidas.

Alias Si no tiene alias definido, se aplica el alias efectivo de la clase.

Filtro

Si no se ha definido ningún filtro, se proprocionará un *filtro nulo* que devuelve toda la población de la clase.

Alias Si el filtro no tiene alias, se aplica como alias el nombre del filtro.

Variable de filtro

Las variables de filtro son huecos que el usuario puede completar para restringir la búsqueda. Son variables libres que participan en la fórmula de búsqueda restringiendo la consulta.

Información complementaria Si a una variable de filtro objeto-valuada no se le ha asignado información complementaria, se le aplica la información complementaria por defecto de la clase del dominio de la variable. En su ausencia no se aplica ninguno.

Unidad de interacción de población Si a una variable de filtro objeto-valuada no se le ha asignado la unidad de interacción destino a la que debe saltar, se le asigna la unidad de interacción de población por defecto de la clase a la que pertenece el dominio de la variable.

3.3.4 Inferencia de unidades de interacción maestro/detalle

En este caso, no se realiza inferencia para crear estas unidades, ya que se trata de escenarios muy particulares, que es necesario que sean detallados por el analista explícitamente.

Alias Si la unidad de interacción no tiene alias definido, se aplica como alias el de la clase que actúa como maestro. Si la clase que actúa como maestro no tiene alias definido, en su lugar se usa el nombre de la unidad de interacción.

[6] Primer conjunto de visualización de la U.I. que se presentará al usuario.

3.4 Inferencia de patrones auxiliares

A continuación se detallará la inferencia que se realiza en aquellos patrones auxiliares que constituyen las piezas básicas para la construcción de las unidades de interacción simples, en particular para: conjuntos de visualización, navegación ofertada y acciones ofertadas. Estos patrones por defecto, sólo serán inferidos si son necesarios para completar alguna unidad de interacción.

3.4.1 Inferencia de conjunto de visualización

Si es necesario inferir un conjunto de visualización para una clase, se crea uno al que se añaden como elementos de visualización, todos los atributos de la clase ordenados por el orden de definición de los atributos.

Para favorecer el prototipado rápido, se permite indicar explícitamente marcar en su definición un conjunto de visualización como **(Auto)**. El uso de **(Auto)** en la especificación del Modelo de Presentación es similar al uso del asterisco '*****' en las sentencias SELECT de SQL. Al usarlo, el analista se independiza de los atributos de la clase en el momento de la especificación, de manera que si se añaden o eliminan posteriormente siguen estando incluidos en el conjunto de visualización.

Elementos del conjunto de visualización

Se infiere su información faltante del atributo del que proceden. Esta es:

Alias Si un elemento del conjunto de visualización definido no tiene alias, se le asigna el alias definido del atributo del que procede. Si el atributo a su vez tampoco tiene, se deja como alias el nombre del atributo.

Patrón de introducción Si el elemento del conjunto de visualización no tiene patrón de introducción asignado, se intenta aplicar el del atributo origen.

Patrón de selección definida Si el elemento del conjunto de visualización no tiene patrón de selección definida asignado, se intenta aplicar el del atributo origen.

3.4.2 Inferencia de navegación ofertada

La inferencia de una navegación ofertada se realiza creando una navegación que contiene saltos a todas las relaciones de agregación (tanto propias como heredadas) y relaciones de herencia.

Alias Si un elemento de navegación ofertada no tiene alias definido, se asigna el alias en función del tipo de relación por la que navegará.

Si se trata de una relación de agregación, se utiliza como alias, el alias definido del último rol que conforma el path de roles atravesado. Si este no existe, se empleará en su lugar el alias efectivo de la unidad de interacción destino.

Si la relación es de herencia, se usará como alias, el alias definido de la última clase del path de herencia y en su ausencia, el alias efectivo de la unidad de interacción destino.

3.4.3 Inferencia de acciones ofertadas

Si se desea inferir un conjunto de acciones ofertadas para una clase se crea un patrón que permite el acceso a todos los servicios ofertables al usuario de la clase en cuestión para la vista actual del sistema. Cada salto tiene como unidad de interacción destino, la unidad de interacción del servicio correspondiente. Se debe tener en cuenta que además de los servicios propios de la signatura de la clase, también se deben ofertar los servicios heredados (servicios definidos en la signatura de las clases ascendentes).

Al igual que los conjuntos de visualización y la navegación ofertada, se pueden crear acciones ofertadas tipo **(Auto)**.

Elementos de las acciones ofertadas

Los elementos de las acciones ofertadas, indican a qué unidad de interacción se va a saltar desde la unidad de interacción actual.

Alias Si un elemento de acciones ofertadas no tiene alias definido se utiliza el alias efectivo de la unidad de interacción destino.

4 Caso de estudio

Vamos a presentar la gestión de un aparcamiento de automóviles. El aparcamiento tiene una serie de plazas de garaje que se alquilan por fracción de horas (la primera hora se paga completa). También es posible alquilar una plaza de garaje a un cliente a cambio de una cuota mensual. Con este sistema se pretende mantener información actualizada sobre el número de plazas ocupadas para su posterior facturación.

El modelo conceptual que representa este sistema es muy sencillo, consta de las siguientes clases: `Aparcamiento` (contiene información sobre la ubicación del aparcamiento, así como las tarifas a aplicar), `PlazaGaraje` (almacena la ubicación de la plaza), `Cliente` (almacena información sobre los cliente que tienen un contrato de cuota mensual, a cada cliente se le asigna una plaza de garaje), `Ticket` (representa el ticket que toma cada automóvil al entrar al garaje, registrando la hora de entrada y salida para el cálculo del importe a pagar) y `Administrador` (del sistema).

En el caso de estudio, se ha especificado completamente todo el esquema conceptual a excepción del modelo de presentación, por lo que el prototipo de interfaz de usuario obtenido será completamente inferido.

Tras realizar el proceso de inferencia, se obtiene como resultado una aplicación con un menú principal cuyos elementos a primer nivel son las clases del modelo. En cada submenú (véase Figura 2-1) se encuentra una entrada para

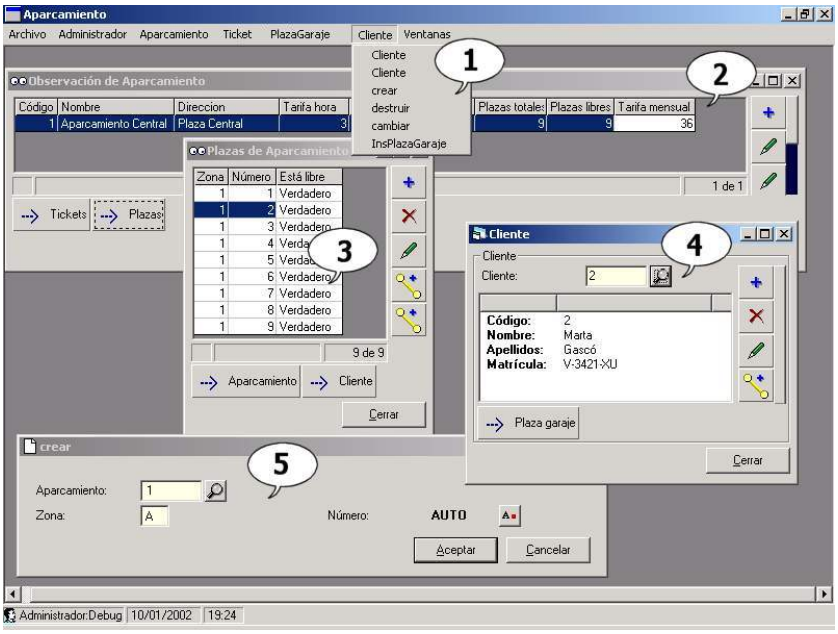

Figura 2: Caso de Uso: Gestión de un Aparcamiento de Automóviles

cada formulario traducido[7] para cada clase. Cada tipo de unidad de interacción genera un tipo de formulario distinto. Cuando la inferencia es total[8] se generan tres tipos de formularios: formularios de población, instancia y servicio (los formularios maestro/detalle proceden de UI especificadas, no inferidas). Los formularios de población (Figuras 2-2 y 2-3) permiten la interacción con conjuntos de objetos (filtrado, ordenación, etc.). Cada formulario de instancia (Figura 2-4) permite visualizar los atributos de un objeto, la navegación a objetos relacionadas y el acceso a los servicios disponibles. Por último los formularios de servicio (Figura 2-5) solicitan los argumentos para la ejecución de cada servicio.

De esta manera, el prototipo de la interfaz permite comprobar rápidamente cuestiones tales como la navegación por todas las relaciones definidas en el modelo, visualización de los atributos definidos y la ejecutación todos los servicios, así como la comprobación de los permisos de los agentes del sistema.

5 Aplicación a un proceso industrial

En CARE Technologies hemos implementado dichos procesos de inferencia en dos vertientes:

[7] Cada Unidad de Interacción da lugar a un formulario

[8] La inferencia total se produce cuando el analista no especifica modelo de presentación.

En una herramienta de modelado: La herramienta de modelado (que permite construir esquemas conceptuales y especificaciones de interfaz de usuario) soporta el proceso de inferencia. De este modo, puede ir guiando al analista. Muestra qué información será inferida y las consecuencias de dicha inferencia. Ahorrando, de este modo, tiempo de especificación para aquello que tiene un comportamiento por defecto.

En traductores de interfaz de usuario: Los traductores de código implementan el proceso de infererencia para completar la especificación con la información faltante antes de proceder a la traducción de la interfaz.

En un estudio[6], Myers prueba que en el desarrollo de aplicaciones, el esfuerzo dedicado al desarrollo de interfaces de usuario supone casi el 50% del esfuerzo total. En la especificación de modelos conceptuales hemos comprobado que la recogida de los requisitos funcionales corresponde al 60% del esfuerzo mientras que los requisitos de interfaz de usuario consumen el 40% restante. Esta última tarea es la que se ve mejorada con las técnicas de inferencia anteriormente descritas.

La especificación de un modelo conceptual transcurre en una espiral de ciclos de construcción y evaluación destinados a verificar que los requisitos están correctamente expresados. Estos ciclos concluyen cuando se da la especificación por validada y completa.

En los primeros ciclos, la especificación se centra en recoger y validar los requisitos funcionales. En estos ciclos, el esfuerzo de especificación de interfaz de usuario debe ser el mínimo impresdindible para alcanzar los prototipos. Aquí hemos apreciado que los procesos de inferencia reducen la especificación de UI en un 85% respecto a una especificación sin inferencia.

Conforme se va alcanzando la especificación final, el analista completa la especificación con los requisistos del usuario. Dicho esfuerzo de especificación sigue siendo menor (respecto a especificación sin inferencia) debido al uso de valores por defecto y el principio de *beneficiar el caso frecuente*. Aquí hemos apreciado una mejora del 50%.

El uso de estas técnicas para la obtención de aplicaciones, ha mejorado la productividad notablemente. Los ciclos de prototipado-evaluación se han acortado ostensiblemente y en menos tiempo han podido llevarse a cabo más iteraciones. Lo cual redunda en una validación más profunda de los requisitos por parte del usuario.

Este tipo de prototipado, no sólo permite obtener interfaces rápidamente, sino que además, permite verificar el correcto funcionamiento de la lógica de negocio espeficada al ser la interfaz obtenidad totalmente funcional y no una mera fachada. Al contrario, el prototipo final si cumple con los requisitos de calidad puede ser empleado como aplicación final.

6 Trabajos relacionados

No es habitual encontrar trabajos que aúnen en un método la especificación de modelos conceptuales junto con la especificación de interfaz de usuario y soportadas por técnicas de inferencia y de generación de código. Aún así, cabe citar JANUS[2] como herramienta de generación de prototipos de interfaz de usuario a partir de modelos orientados a objetos.

JANUS es un trabajo de la Universidad de Ruhr en Alemania que partiendo del Análisis Orientados a Objetos[5] genera parte de la interfaz de usuario asociada. JANUS emplea un modelo de configuración de clases donde se recogen: los atributos y métodos para cada clase y las relaciones de agregación y herencia entre las clase.

A partir de este modelo de objetos básico JANUS es capaz de generar parte de la interfaz de usuario a partir de reglas bien definidas como las siguientes. Para cada clase genera:

- Un formulario de clase que contiene: controles de visualización para los atributos de la clase y botones de acción para todos los servicios de la clase.
- Un formulario de tipo tabla que contiene: controles para los atributos de la clase, botones de acción para todos los servicios de la clase y una rejilla que muestra en cada fila un objeto perteneciente a la población de la clase.

JANUS se basa un modelo conceptual orientado a objetos (a la Coad & Yourdon [5]) sin extensiones. Dicho modelo captura el sistema con la expresividad proporcionada por el método OOA pero no captura requisitos de interfaz de usuario propiamente dicho. Esta carencia de información acerca de los requisitos de la interfaz de usuario provocan que la interfaz generada por JANUS sea tosca y solo contemple la parte estática de la interfaz.

Por tanto, puede decirse que JANUS permite el prototipado de interfaz de usuario de un modelo orientado a objetos, sin embargo, no se dispone de modelo para las especificación de interfaces, limitando mucho las interfaces de usuario generadas a partir de modelos OOA.

7 Conclusiones

El proceso de inferencia permite obtener prototipos más rápidamente respecto a los métodos convencionales. Dichos prototipos ofrecen la funcionalidad completa del sistema de un modo estandarizado para ayudar al analista en la validación de los requisitos del sistema con el usuario.

Debido a que la inferencia es un conjunto de heurísticos, el resultado del proceso de inferencia puede no ser siempre el deseado por el analista, aún así, en un porcentaje muy alto de casos ofrece un resultado aceptable que permite reducir visiblemente el tiempo invertido en la especificación de la interfaz de usuario. Es decir, incrementa notablemente la productividad en su elaboración y además simplifica el modelo resultante.

Referencias

[1] R. Balzer, T.E. Cheatham, C. Green, "Software Technology in the 1990s: Using a New Paradigm", *IEEE Computer*, Noviembre de 1983, pags. 39-45.

[2] H. Balzert, "From OOA to GUIs: The JANUS system", *IEEE Software*, 8(9), Feb. 1996, pags. 43-47.

[3] *UML Semantics. Version 1.1*, documento OMG ad/97-08-04, 15 de Septiembre, 1997.

[4] *UML Notation. Version 1.1*, documento OMG ad/97-08-05, 15 de Septiembre, 1997.

[5] P. Coad, E. Yourdon, *Object Oriented Analisis*, Prentice Hall, 1991.

[6] B. A. Myers, M. B. Rosson, *Survey on User Interface Programming. In Striking a Balance.*Proceedings of CHI'1992. Monterey, New York: ACM Press, pags. 195-202, Mayo de 1992.

[7] O. Pastor, F. Hayes, S. Bear, "OASIS: An OO Specification Language," *Proc. of CAiSE-92 (Conference)*, Lecture Notes in Computer Science 593, Springer-Verlag, 1992, Manchester (UK), pags. 348-363.

[8] O. Pastor et al., "OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods", *In Proceedings of 9th International Conference, CAiSE97*, Lecture Notes in Computer Science 1250, Springer-Verlag, Junio de 1997, Barcelona, Spain, pags. 145-159.

[9] O. Pastor, P.J. Molina, A. Aparicio. *"Specifying Interface Properties in Object Oriented Conceptual Models."*, In Proceedings of Working Conference on Advanced Visual Interfaces, AVI 2000, pags. 302-304, Palermo, Italia. Ed. ACM, Mayo de 2000.

[10] P.J. Molina, O. Pastor, S. Martí, E. Insfrán. *"Ingeniería de Requisitos aplicada al modelado conceptual de interfaz de usuario"*. In Proceedings of IDEAS'2001, pags. 181-192, Santo Domingo, Heredia, Costa Rica, Abril de 2001, Ed. CIT.

[11] P.J. Molina, O. Pastor, S. Martí, J. Fons, E. Insfrán. *"Specifying Conceptual Interface Patterns in an Object-Oriented Method with Code Generation"*. In Proceedings of UIDIS'2001, pages 72-79, Zurich, Suiza, Mayo de 2001, IEEE Computer Society.

[12] P.J. Molina, S. Meliá, O. Pastor. *"JUST-UI: A User Interface Specification Model"*. In Proceedings of CADUI'2002 Conference, pages 323-334, Valenciens, Francia, Mayo de 2002, Kluwer Academics (en prensa).