Prototipado de interfaces de usuario a partir de escenarios y modelos UML[1]

Juan Sánchez Díaz; Alberto Aparicio Vila; Oscar Pastor López; Juan Jose Fons

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia.
Camino de Vera s/n. 46022.Valencia (España)
{jsanchez, aparicio, opastor, jjfons} @dsic.upv.es

Abstract. En este artículo presentamos un proceso de ingeniería de requerimientos que genera de un modo automático prototipos de interfaces de usuario a partir de escenarios, obteniéndose una especificación formal del sistema en la forma de diagramas de transición entre estados. Esta especificación se incluye dentro de un entorno de ejecución, pudiéndose animar cada uno de los prototipos. Los escenarios se describen mediante *message sequence charts* (MSC), enriquecidos con información referente a la interface de usuario. El proceso utiliza un modelo de casos de uso. Para una familia de casos, se genera un conjunto de MSC. Los diversos MSC se transforman en diagramas de transición entre estados para los objetos de interface y de control. A partir de estos, se genera un prototipo de interface de usuario formado por un modelo de vistas por actor, un formulario por caso de uso y un modelo de navegación entre formularios, esto último se obtiene a partir de las relaciones entre los casos. Basándonos en la ejecución del prototipo y en la realimentación de los usuarios del mismo, este puede ser modificado de forma iterativa desde un entorno visual de construcción de interfaces de usuario. La propuesta, soportada por una herramienta CASE, es apropiada para aplicaciones que trabajen con bases de datos.

Keywords: Prototipos de interface de usuario, elicitación de requerimientos, validación de escenarios.

1. Introducción

La primera etapa dentro de la concepción de un sistema informático consiste en entender y representar de un modo apropiado los requerimientos de usuario. Este proceso recibe el nombre de ingeniería de requerimientos y ha sido reconocido como

[1] Trabajo subvencionado por el programa FEDER, con ref. TIC 1FD97-1102

una tarea crucial dentro del proceso de desarrollo (Bennett, 1997;Kotonya et al. 1998). Los errores originados en la etapa de requerimientos pueden permanecer sin detección hasta la etapa de operación, provocando fallos con serias consecuencias sobre todo en sistemas críticos.

Los errores en la etapa de elicitación de requerimientos están provocados principalmente por el *gap* existente entre los usuarios y el proceso de desarrollo. A los usuarios se le presenta una especificación abstracta del sistema, que en la mayor parte de los casos, es incompresible para ellos.

El comportamiento de un sistema puede ser descrito de forma intuitiva mediante la utilización de escenarios. Un escenario se define como una descripción parcial del comportamiento de un sistema y su entorno que se da en una determinada situación (Benner et al. 1993). La parcialidad de la descripción permite que un escenario cubra o abarque partes del comportamiento del sistema. Esta una característica interesante, ya que diferentes usuarios pueden percibir el sistema de forma diferente.

Los escenarios son una herramienta valiosa para captar y comprender requerimientos y para analizar interacciones hombre-máquina (Nielsen 1995). Un proceso estándar de ingeniería de requerimientos, basado en escenarios (Somé et al. 1996), contiene dos tareas principales. En la primera, se generan especificaciones a partir de los escenarios para describir el comportamiento del sistema. La segunda, consiste en validar estos con los usuarios mediante simulación o prototipación. Las dos tareas son tediosas si no están soportados por alguna herramienta automática o semiautomática.

La etapa de validación se suele abordar en algunos casos utilizando herramientas RAD (Desarrollo Rápido de Aplicaciones) (Kerr et al. 1994). En cualquier forma el proceso de definir y generar el prototipo de la interface de usuario es un proceso manual, ya que cada objeto debe crearse de forma explícita.

Nosotros estamos interesados en la validación de escenarios mediante la generación automática de prototipos de la interface de usuario de la aplicación y la ejecución simbólica de los mismos.

A diferencia de otras propuestas como las del grupo Gelo (Elkoutbi et al. 1999) nosotros disponemos de una herramienta visual que permite automatizar casi todo el proceso. Por otra, el método utilizado (véase sección 3 y posteriores) genera automáticamente un modelo de navegación entre formularios, basado en las relaciones usa/incluye y extiende entre los casos de uso.

En este artículo se presenta una propuesta metodológica y la herramienta asociada que le da soporte, dentro del campo de la ingeniería de requerimientos, basada en el lenguaje unificado de modelado (UML) y en los *message sequence chart* (MSC) (ITU, 1996). La aproximación emplea un proceso iterativo, que permite derivar de forma semiautomática, prototipos de la interface de usuario a partir de escenarios, y una especificación formal del sistema en la forma de diagramas de transición entre estados. Estos diagramas describen el comportamiento dinámico de los objetos de interface y de los objetos de control asociados a cada caso de uso o cada MSC. De las cuatro actividades del método, las iniciales no son automáticas, necesitan la intervención del analista, mientras que las dos úlltimas automatizan el proceso de validación de escenarios mediante prototipación.

Este trabajo supone un avance cualitativo con respecto a trabajos nuestros anteriores (Sánchez 1999, Sánchez et al. 1999), la información de la interfaz de usuario se introduce en los MSC, los diagramas de transición generados ahora no son directamente manipulables, la herramienta es más robusta que la herramienta inicial.

El artículo está estructurado como sigue: la sección segunda introduce los modelos de UML que utilizamos en nuestra propuesta, junto al ejemplo que ilustrará el proceso de generación de los prototipos. La sección 3 contiene una discusión detallada de las actividades y procesos contenidos en nuestra aproximación. Los trabajos relacionadas se discuten en la sección 4. Finalmente, las secciones 6 y 7 contienen, respectivamente, las futuras líneas de trabajo y las conclusiones.

A lo largo del artículo se incluyen capturas de pantallas de la herramienta CASE que da soporte al método.

2 El lenguaje unificado de modelado

El lenguaje unificado de modelado (UML) (Booch et al. 1999) se ha impuesto como un estándar de hecho para la modelización de sistemas orientados a objetos. De las diversas vistas o modelos que propone, nosotros utilizamos en nuestra aproximación el modelo de casos de uso, los diagramas de transición entre estados y una variante de los diagramas de interacción o los diagramas de secuencia: los MSC. El modelo de objetos o diagramas de clases, que no describiremos, se utiliza en la etapa de construcción de los MSC para mostrar las clases emisoras y receptoras de eventos. Para ilustrar la propuesta utilizaremos un sistema de gestión de pedidos descrito en (Schneider et al 1998).

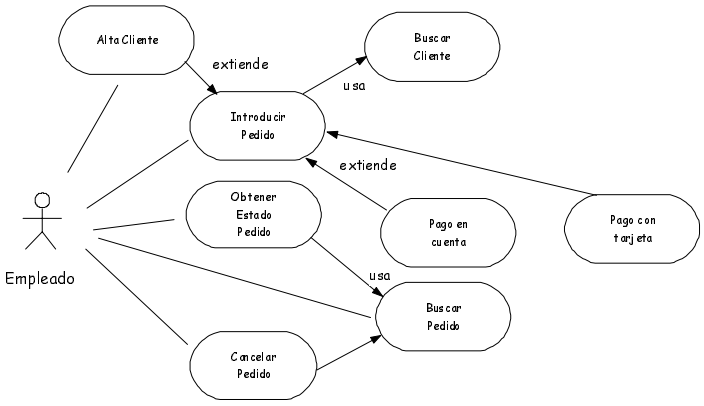

Fig. 1. Modelo de casos de uso para el sistema de pedidos.

2.1 El modelo de casos de uso

El modelo de casos de uso contiene los agentes o los actores que pueden interactuar con el sistema, esta interacción se muestra en la forma de caso de uso. Un caso de uso, tal como fue introducido por I. Jacobson et al. (1992), contiene una secuencia de transacciones/eventos que se intercambian los actores y el sistema cuando se ejecuta cierta funcionalidad del mismo. La figura 1 muestra el modelo de casos de uso, simplificado por cuestiones de espacio, para un sistema de generación de pedidos. El único actor participante es el empleado que se encarga de: introducir pedidos, cancelarlos, obtener el estado en el que se encuentran (admitido, listo para enviar, enviado) y finalmente de la devolución de artículos.

Dos relaciones pueden definirse entre casos: usa/incluye y extiende. La relación usa se emplea cuando un flujo de eventos puede insertarse textualmente dentro de otro caso de uso. Por el contrario la relación de extensión muestra un flujo alternativo de ejecución. Se puede considerar como una inclusión bajo ciertas condiciones.

2.2 Los MSC

Los MSC se emplean ampliamente dentro del campo de las telecomunicaciones para describir el intercambio de información entre las clases de un sistema. En su forma más simple, sin información de control y sin condiciones, se consideran equivalentes a los diagramas de interacción de UML. La figura 2 contiene la notación utilizada dentro de nuestra herramienta.

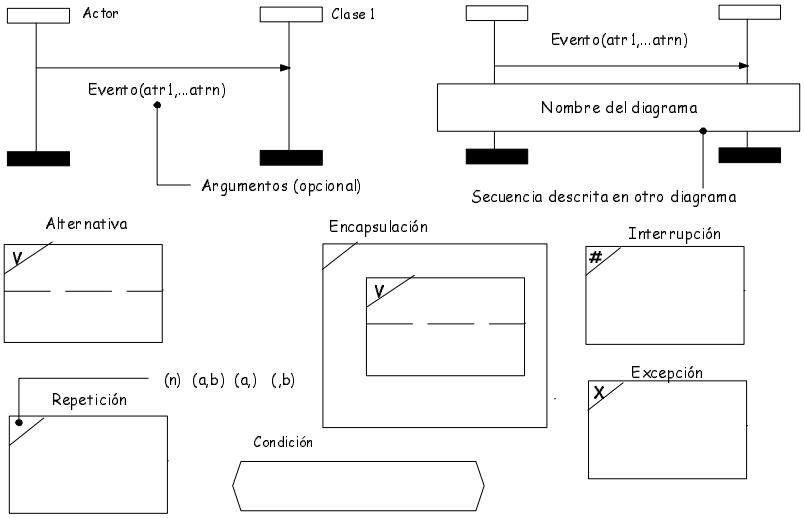

Fig. 2. Notación para los MSC.

Las líneas verticales representan clases o actores del sistema, el envío de eventos o el intercambio de información se refleja en la forma de líneas horizontales. Cada evento puede llevar asociado un conjunto de atributos. Estos pueden ser tipos básicos (enteros, reales, etc.) o tipos clase. Existe una notación adicional que refleja: repetición de eventos, alternativa, excepciones, interrupciones y condiciones que pueden reflejar el estado en el que se encuentra el sistema. Los MSC se pueden descomponer por niveles, de forma que en cada diagrama el analista puede utilizar el nivel de abstracción que desee. Para cualquier detalle puede consultarse ITU (1996) o Telelogic (1999, Relesase Notes).

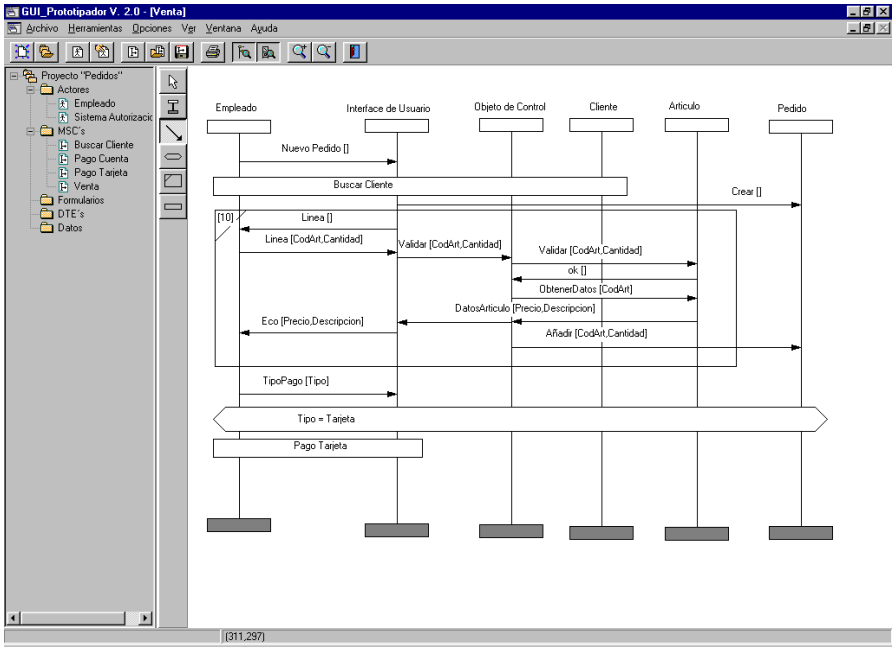

Fig. 3. MSC para realizar venta con tarjeta.

La figura 3 contiene el MSC para realizar una venta. Este diagrama muestra los eventos cuando se introduce un pedido en el sistema y cuando se selecciona pagar con tarjeta. En cada MSC incluimos un objeto de control que hace de mediador entre la interface de usuario y la sociedad o modelo de objetos del dominio del problema. Los rectángulos contienen el nombre del diagrama MSC asociado que contiene la expansión: *buscar cliente* y *pago con tarjeta*. Esta expansión es sencillamente una forma de reutilizar flujos de eventos.

2.3 Diagramas de transición entre estados

Los diagramas de transición entre estados (DTE, Harel 1987) se emplean dentro de las metodologías orientadas a objetos para describir el comportamiento de las clases del sistema. Sirven para reflejar los ciclos de vida de los objetos. También pueden utilizarse para describir el comportamiento de la interface de usuario (Horrocks, 1998) de una aplicación interactiva. En nuestra aproximación, los emplearemos para describir el objeto de interface de usuario y el objeto de control que aparece en cada diagrama MSC. Los DTE serán utilizados en el proceso de animación de la interface de usuario.

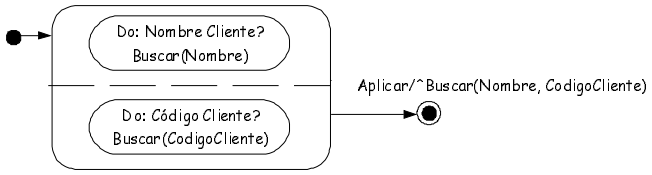

Fig. 4. Diagrama de transición entre estados de la I.U. para buscar cliente.

3. Descripción de la propuesta

La figura 5. contiene una representación esquemática de las actividades contenidas en el método propuesto. Como hemos comentado anteriormente las dos primeras actividades: representación de escenarios y descripción de casos de uso son actividades manuales que debe realizar el analista. Las dos últimas: generación de la especificación y generación de los prototipos, están completamente automatizadas

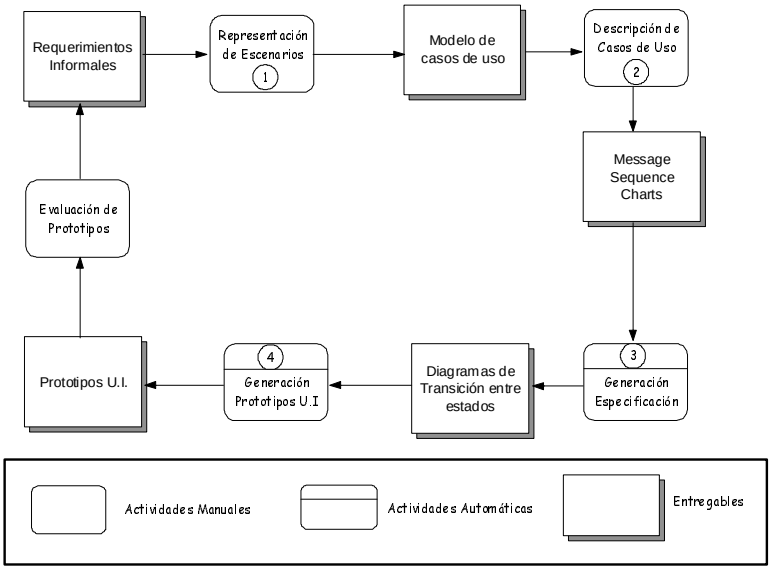

Fig. 5. Representación esquemática del método.

La figura 5 contiene también el orden en el cual se llevan a cabo esas actividades. El proceso comienza por la etapa de representación de escenarios donde se crea un modelo de casos de uso. La siguiente etapa consiste en describir los casos de uso mediante MSC. De un conjunto de MSC's se obtiene automáticamente un conjunto de DTE en la etapa de generación de la especificación. Finalmente, la última etapa consiste en generar, también de un modo automático, los prototipos de las interfaces de usuario. El metodo es de naturaleza iterativa, el prototipado sirve para validar y

enriquecer los requerimientos iniciales. A continuación explicaremos en detalle cada una de las fases.

3.1. Representación de escenarios

En la etapa de representación de escenarios, el analista se encarga de construir el modelo de casos de uso del sistema. Contiene tres capas o fases: el modelo inicial, el contextual y el estructurado (Sánchez et al 1999, 2000). El modelo inicial muestra los actores y los casos de uso agrupados por funcionalidad similar.

Nombre: Buscar Cliente	**Propósito**: El caso de uso se ejecuta cada vez que se desea buscar un cliente
Prioridad: Alta	**Estado de desarrollo**: Análisis
Relaciones	
Utiliza a:	
Extiende a:	
Proceso/Descripción	
Precondiciones:	Ninguna
Postcondiciones:	Ninguna
Actores:	Empleado
Flujo de Eventos	
Comunicación Actor Sistema (Intenciones de Usuario)	
Responsabilidad Sistema (Obligaciones del sistema)	
1. El empleado solicita buscar cliente	
2. El empleado introduce el nombre del cliente y/o un código de cliente	
3. El empleado selecciona “Aplicar”	
4. El sistema efectua la búsqueda por nombre o por código	
5. El sistema presenta el resultado de la misma.	
Extensiones asíncronas	
6. En cualquier punto el empleado puede anular la búsqueda	
Extensiones síncronas	
7. Si en 3 no se introduce ni nombre ni código se escribe un mensaje de error.	

Table 1. Plantilla para describir casos de uso.

La información contextual de cada caso, que contiene el apartado de precondiciones, postcondiciones, etc; y finalmente, el modelo estructurado que muestra las relaciones definidas en UML: incluye/usa y extiende.

Los casos de uso se pueden describir empleando plantillas en lenguaje natural, nosotros emplearemos una variante de la propuesta por L. Constantine et al (1999).

La plantilla de la tabla 1 muestra la descripción del caso de uso *buscar cliente*. El flujo de eventos está dividido en dos columnas: intenciones de usuario y obligaciones del sistema. Esta división permite identificar las solicitudes de servicios, por parte de los actores, y la provisión de información, por parte del sistema. Es util para construir los MSC en la siguiente etapa del método.

3.2 Sintesis de casos de uso

La definición formal de un caso de uso se consigue utilizando un lenguaje gráfico no ambigüo como es MSC. En esta fase, que es manual, las plantillas sirven como ayuda para que el analista pueda detectar los eventos enviados por los actores y por las clases del dominio del problema. En cada MSC, ademas de los actores participantes (iniciador, secundarios), situamos una clase para la interface de usuario y una clase que actua como objeto de control de acuerdo a la propuesta contenida en Objectory (Jacobson et al. 1992) y en UML (Jacobson et al 1999, Rosenberg 1999).

El formalismo utilizado, a diferencia de por ejemplo los diagramas de trazas de OMT (Rumbaugh et al 1997), permite reflejar alternativas en el envio de eventos, iteraciones, excepciones, etc. (véase la Figura 2). La notación permite mostrar gráficamente las relaciones de dependencia entre casos de uso. Si un caso *A* usa a un caso *B*, en el MSC de *A* se situa un rectángulo con el nombre *B*. Esto indica que el diagrama puede expandirse en el diagrama B. De un modo análogo pueden tratarse, a nivel gráfico, la relación de extensión. Para esta se emplea una condición y a continuación la llamada al diagrama que produce la extensión.

En la figura 3 se muestra un MSC *introducir pedido* con tipo de pago, pago con tarjeta. La relación usa/incluye con el caso de uso buscar cliente, aparece reflejada en el diagrama en la forma de rectángulo. En cambio para el punto de extensión, pago con tarjeta, utilizamos una condición y el rectangulo que nos indica el caso que extiende al considerado.

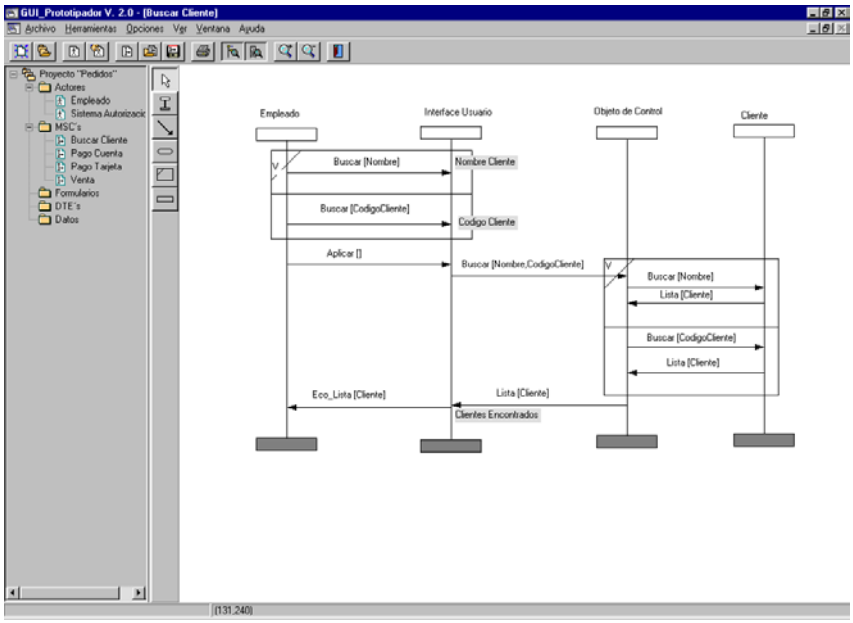

Fig. 6. MSC con etiquetas para buscar cliente.

En esta fase también debe introducirse información relativa a las etiquetas que aparecen en la interface de usuario. Cada vez que se tenga una entrada/salida de

información, el analista indicará en el MSC la etiqueta correspondiente que acompañará a esa información.

La figura 6 contiene el MSC o caso de uso *buscar cliente*. El analista ha situado tres etiquetas: nombre cliente, código cliente y clientes encontrados. Se corresponden con dos campos de entrada y con un eco o campo de salida. Esta información aparecerá literalmente en el formulario asociado al caso de uso.

Además de las etiquetas, debe situarse información relativa al tipo de los atributos de cada evento del diagrama. Los tipos permitidos son los tipos básicos (número, booleano, carácter, strings, enumerados) y los tipos clase.

El tipo de los atributos de los eventos se emplea también, véase siguientes secciones, en el proceso de generación de la interface.

3.3. Generación de la especificación

La especificación está formada por un conjunto de diagramas de transición entre estados que describen el comportamiento del sistema. Un diagrama para cada función de la interface de usuario y otro para cada objeto de control que aparezca en un MSC.

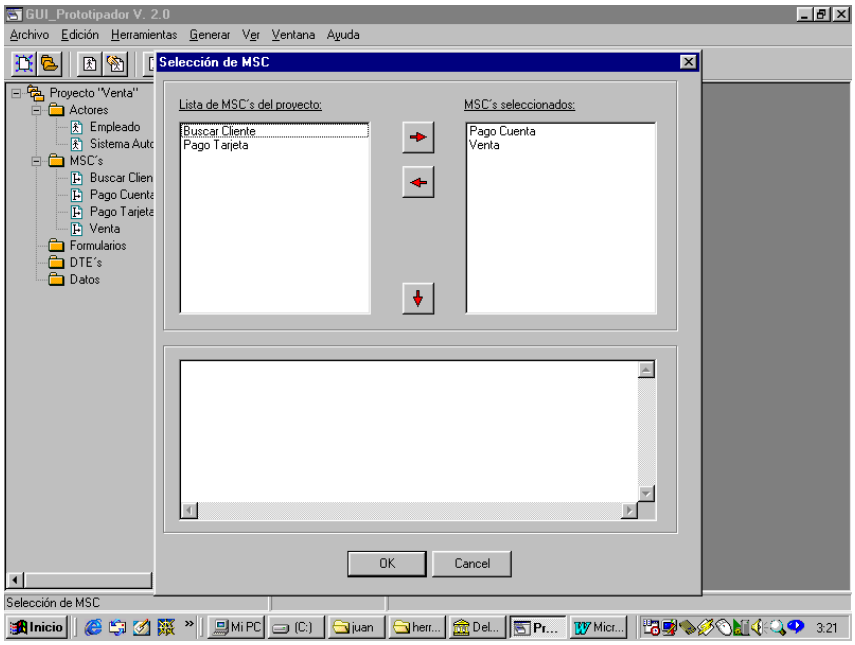

Fig. 7. Proceso de selección de los MSC para generar la especificación.

El analista selecciona los MSC para los cuales desea generar la especificación, esta especificación se almacena en el repositorio de la herramienta. Para los DTE generados, a diferencia de la versión anterior de la herramienta (Sánchez et al, 2000),

no utilizamos una representación gráfica, se almacenan y visualizan en forma de tabla de transiciones.

En el proceso de generación de las tablas de transiciones, utilizamos una variante del algoritmo propuesto por Systa (1997), los detalles pueden verse en (Sánchez 1999). El proceso consiste en asociar a caminos dentro de un MSC, recorridos en un DTE.

La figura 6 muestra el proceso de selección de diversos MSC del proyecto en curso para proceder a la generación de su especificación. De cada MSC se seleccionan las clases para las cuales se genera de modo automático el diagrama de transición entres estados. Lo habitual es seleccionar la interfaz de usuario y el objeto de control.

3.4. Obtención de los formularios y de los componentes de la I.U.

El proceso de generación del prototipo de la interfaz de usuario, para la aplicación a desarrollar, comienza analizando el modelo estructurado de casos de uso o bien su equivalente los MSC. Para cada actor que sea un actor iniciador se obtiene un modelo de vistas por formulario, este contiene los formularios que puede invocar directamente desde el menú de la aplicación. Cada caso de uso que pueda ejecutar algún actor se convierte en un formulario. El modelo de navegación entre estos se obtiene analizando las relaciones usa y extiende.

Entrada de Información: E(a1,..,an)		
Tipo ai	**Condición**	**Componente generado en el formulario**
Enumerado	Talla <4	Botón de selección por elemento del tipo
	Talla >4	ComboBox conteniendo los elementos del tipo
Booleano	-	Cuadro de Selección
Carácter	-	Campo de edición.
Numero	-	Campo de edición.
String	-	Campo de edición.
Tipo Clase	-	Grid, una entrada para cada atributo de la clase
Salida Información: Eco(a1,..,an)		
Básico		Caja de salida
Clase		Grid, una columna para cada atributo de la clase
Lista de objetos		Grid con barra de desplazamiento horizontal.

Table 2. Esquema de generación de los componentes de un formulario.

A partir de la figura 1 se generarían los siguientes formularios: alta cliente, buscar cliente, obtener estado de pedido, buscar pedido, cancelar pedido, e introducir pedido. Obsérvese que no se generan dentro del modelo de vistas formularios para Pago en Cuenta y Pago con Tarjeta.

- Si A (Introducir Pedido) es un caso que utiliza un caso B (Buscar Cliente), entonces en el formulario asociado a A se sitúa un botón que permite navegar hacia el formulario asociado con B.
- Si B (Pago en Cuenta) es un caso que extiende a un caso A (Introducir Pedido) y B no puede ejecutarse directamente entonces el formulario asociado a B aparece

unido al formulario asociado a A. En caso contrario, si B (alta cliente) puede ejecutarse, en A se sitúa un botón que permite navegar hacia B.

Para cada formulario que se desea generar, se analiza la información de interfaz (etiquetas, tipo de los atributos de los eventos) contenida en los MSC. Para crear los componentes la herramienta usa el esquema de la tabla 2. Puede verse también como un conjunto de reglas cerrado que se emplean en el proceso de generación.

Para el MSC de la figura 6, y de acuerdo a la tabla anterior, la herramienta genera automáticamente, un formulario con dos campos de entrada y con etiquetas "Nombre Cliente" y "Código Cliente". Para el eco que producirá el sistema, al estar este formado por una lista de objetos, se genera una rejilla, donde cada columna se corresponde con un atributo de la clase Cliente. Los botones Aplicar y Salir se sitúan por defecto en todo formulario generado.

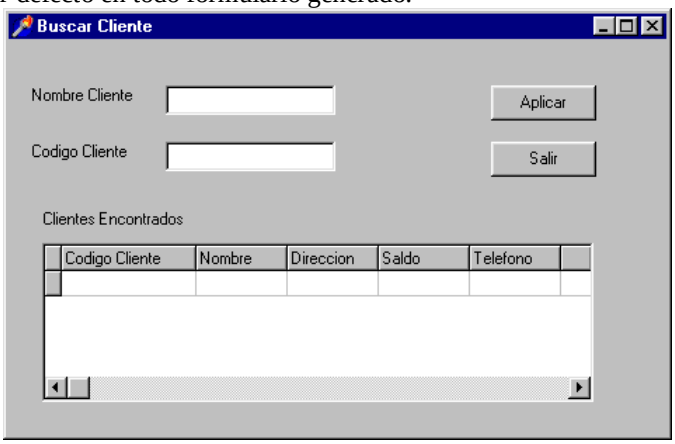

Fig. 8. Formulario para el caso de uso buscar cliente.

Dado que buscar cliente no utiliza ni es extendido por ningún otro caso de uso no existen los botones de enlace que permiten la navegabilidad. Desde el formulario asociado a introducir pedido se pude alcanzar a buscar cliente.

El formulario se puede modificar desde el entorno visual Delphi y también se puede animar simbólicamente, simulando la ejecución del diagrama de transición entre estados asociado.

4. Trabajos relacionados

En el campo de la generación de interfaces de usuario existe un variado abanico de propuestas, que van desde aquellas que emplean modelos de datos y generan únicamente la parte estática de la interfaz, hasta aquellas basadas en escenarios que soportan desde un automatismo limitado hasta un automatismo total.

Dentro de los modelos de datos se suele utilizar alguna variante del modelo entidad-relación, o bien un modelo de objetos. Por el contrario las basadas en escenarios suelen emplear redes de Petri o diagramas de transición entre estados. A continuación comentaremos algunas de las propuestas más relevantes.

Entre las aproximaciones, basadas en modelos de datos, que generan únicamente la parte estática de una interfaz cabe citar: Janus y Trident. En Janus (Baltzer, 1996) se construye un modelo de objetos, a cada clase no abstracta del modelo se le asocia una ventana, el analista selecciona manualmente los métodos y atributos que son relevantes de cara a esa interfaz. El entorno, genera entonces una vista estática. Trident (Vanderdonckt, 1995) utiliza una aproximación más sofisticada que la anterior. A partir de un modelo entidad relación, se dibuja un grafo de actividades que conecta tareas interactivas con funciones y datos del sistema. El grafo se emplea como entrada a un proceso que selecciona las distintas unidades de presentación. Al igual que en el caso anterior se genera únicamente la parte estática.

En TADEUS (Schlungbaum et al. 1996) se emplea una etapa llamada genéricamente diseño de dialogos abarca dos tareas que tienen que abordarse manualmente: el diseño del dialogo navegacional que describe la secuenciación entre las distintas vistas, y el diseño del dialogo de procesamiento que describe a su vez los cambios que sufren los objetos dentro de una vista de dialogo. El formalismo empleado está basado en las redes de Petri. En el proceso de generación se obtienen scripts que se pueden incluir en un sistema de gestión de interfaces.

Finalmente, la propuesta más próxima a la nuestra es SUIP (Elkoutbi et al. 1999; Khriss eta al. 1999) (Scenario-based User Interface Prototyping) de la Universidad de Montreal. Emplean diagramas de colaboración, que definen utilizando ficheros planos. Estos diagramas están enriquecidos con información de la interface de usuario. No obtiene, a diferencia de nuestra propuesta, de modo automático un modelo navegacional. En la actualidad están modificando su propuesta para utilizar redes de Petri (Elkoutbi et al. 2000) en lugar de DTE.

5. Conclusiones

La propuesta se enmarca dentro del campo de la validación de requisitos mediante la generación semiautomática de prototipos de la interface de usuario. En lugar de girar en torno a los aspectos de estructura del problema, esta gira en torno a los aspectos de comportamiento.

A diferencia de algunas aproximaciones, está soportada por una herramienta CASE de calidad casi industrial. Hemos construido un metamodelo para el modelo de casos de uso y para los MSC. Este metamodelo se ha utilizado para diseñar una base de datos relacional que actuó como repositorio de información. La herramienta está programada en el lenguaje Delphi, mientras que la base de datos es Interbase 5.1.

Otro aspecto también a destacar, es que permite obtener de forma automática el modelo de navegación entre formularios. El aspecto menos conseguido de la propuesta radica en la apariencia de los formularios generados.

6. Trabajos futuros

En la actualidad estamos tratando de incorporar nuestro esquema de generación y definición de interfaces dentro de RSI (Requirements/Service/Interface) de M. Collins (1999). En esta aproximación se emplean tres categorías de casos de uso: requerimientos, interface y servicios. Los casos de uso de requerimientos documentan los procesos de negocio automatizables. El modelo de interface describe de forma estática las interfaces de usuario de la aplicación, en el sentido de utilizar únicamente dibujos con los distintos componentes de la misma. Finalmente, el modelo de servicios describe la funcionalidad que el sistema ha de proporcionar, independientemente de las necesidades de una interface de usuario particular. Nuestra propuesta encajaría perfectamente dentro del modelo de interface, es decir la capa de interfaz se puede describir mediante un prototipo generado automáticamente.

Finalmente, una mejora consistiría en parametrizar el proceso de generación, sobre distintos lenguajes de programación: Visual C++, Java, Visual Basic y también pudiendo modificar las reglas descritas en la tabla 2. En la actualidad los formularios se generan únicamente para el lenguaje Delphi. Estamos evaluando la posibilidad de generar un lenguaje de script intermedio y traductores específicos para diversos entornos visuales.

Si se observa alguna de las figuras que contienen capturas de la herramienta, existe una carpeta etiquetada con “datos”. La idea es poder definir datos reales y tratar de animar el prototipo con esos datos. En la línea de los validadores de modelos de objetos basados en DTE (Bridge Point, 1999).

7. Agradecimientos.

A los miembros del grupo de investigación de modelización conceptual OO del departamento de sistemas informáticos y computación, por sus valiosas sugerencias durante la elaboración de este artículo. En particular nos gustaría agradecer muy especialmente a Ruben Requena Navarro y a Dani Antón Piqueras que han efectuado la implementación de la herramienta que da soporte al método.

Referencias

Balzert, H.; “From OOA to GUIs: The Janus System”. IEEE Software, 8(9), Febraury 1996, pp 43-47.

Benner K.M.; Feather M.S.; Johnson W.L. “Utilizing Scenarios in the Software Development Process. En Information System Development Process, pp 117-134. Elsevier Science Publisher, 1993. Editores: N. Prakash, C. Rolland, y B. Percini.

Bennett, D.W. “Designing Hard Software: The Essential Tasks”. Manning Publications co, 1997.

Booch G; Rumbaugh J; Jacobson I. “El lenguaje unificado de modelado”. Addison-Wesley. 1999.

Bridge Point. Herramienta CASE Project Technology. http://www.projtech.com

Collins-Cope, M; “RSI- A Structured Approach to Use Cases and HCI Design”. Manuscrito no publicado (comunicación personal), Ratio Group Ltd. 1999.

Constantine L.L; Lockwood L.A.D. “Software for Use: A practical Guide to the Models and Methods of Usage-Centered Design”. Addison Wesley 1999.

Elkoutbi M; Khriss I; Keller R; “Generating User Interface Prototypes from Scenarios”. Proceedings of the Fourt IEEE International Symposium on Requeriments Engineering (RE’99). Limerick Ireland 1999.

Elkoutbi M; Keller R; “User Interface Prototyping based on UML Scenarios and High-level Petri Nets”. Application and Theory of Petri Nets (21 ATPN). Junio 2000. Springer-Verlag.

Harel D; “State Charts: a visual formalism for complex systems”. Science of Computer Programming, 8(3), 231-274. 1987.

Horrocks I. “Constructing the User Interface with Statecharts”. Addison-Wesley, 1998.

ITU: Recommendation Z. 120: Message Sequence Chart (MSC). ITU, Geneva, 1996.

Jacobson I et al. “Object-Oriented Software Engineering: A use case driven approach”. New-York ACM Press, 1992.

Jacobson I; Booch G; Rumbaugh J. “The Unified Software Development Process”. Addison-Wesley,1999.

Kerr J; Hunter R; “Inside RAD”. McGraw-Hill 1994.

Kotonya, G; Sonmmerville, I. “Requirements Engineering: Process and Techniques”. John Wiley & Sons. 1998.

Khriss I; Elkoutbi M; Keller R; “Scenario-based User Interface Prototyping”. Grupo Gelo. http://www.iro.umontreal.ca/~elkoutbi/SUIP/index.html

Nielsen J. “Scenarios in Discount Usability Engineering”. Scenario-Based Design: Envisioning Work and Technology in System Development. John Wiley & Sons, 1995. pp 59-85

Rosenberg, D; Scott, K. “Use Case Driven Object Modeling with UML”. Addison Wesley, 1999.

Rumbaugh J; et al. “Modelado y Diseño Orientado a Objetos”. Prentice Hall 1997.

Schlungbaum E; Elwert T; “Modeling a Netscape-like browser using TADEUS”. CHI’96. Vancuver 1996. Pp 19-24

Schneider G.; Winters J.P. “Applying Use Cases: A practical Guide”. Addison Wesley 1998.

Sánchez Díaz J; “Construcción de diagramas de transición entre estados para objetos de interface a partir de trazas de eventos”. TR-DSIC,-1999. U. Politécnica de Valencia.

Sánchez J; Pelechano V; Pastor O; “Un entorno de generación de interfaces de usuario a partir de casos de uso”. WER’ 99. pp 106-116. Buenos Aires. Septiembre 1999.

Sánchez J; Pelechano V; Insfrán E; “Un entorno de generación de prototipos de interfaces de usuario a partir de diagramas de interacción”.Ideas 2000. Pp 145-155. Cancún (Mexico). Abril 2000.

Systa T. “Automated Support for Constructing OMT Scenarios and State Diagrams”. Department of Computer Science. University of Tampere. Technical Report A-1997-8

Somé S.S.; Dssouli R; Vaucher, J. "Toward an Automation of Requirements Engineering using Scenarios". Journal of Computing and Information, vol 2,1, 1996, pp 1110-1132,

Telelogic. Herramienta CASE Telelogic Tau 3.6.2. http://www.telelogic.com. 1999.
Vanderdonckt J. “Knoledge-Based Systems for Automated User Interface Generation: the TRIDENT Experience”. RP-95-010. Marzo 1995.