ANÁLISIS COMPARATIVO A LAS TÉCNICAS PARA MODELAR EL FUNCIONAMIENTO DE LOS SISTEMAS DE INFORMACIÓN

COMPARATIVE ANALYSIS OF THE TECHNIQUES FOR MODELLING THE PERFORMANCE OF INFORMATION SYSTEMS

ANALYSE COMPARATIF DES TECHNIQUES POUR MODELER LE FONCTIONEMENT DES SYSTEMES D'INFORMATION

Gloria G. Madows
Georgia State University
madowsg@cis.gsu.edu

(Tipo de artículo: **REFLEXIÓN**. Recibido el 01/02/2011. Aprobado el 30/04/2011)

RESUMEN
En este trabajo se analizan las técnicas propuestas en los métodos actuales para especificar los aspectos de diálogo usuario-sistema de un producto software en el paradigma de desarrollo Orientado por Objetos. Esos métodos no ofrecen suficientes directrices acerca de cómo modelar dichos aspectos, y las técnicas disponibles necesitan algún refinamiento y elaboración para adaptar esa tarea en el proceso de especificación del software. Primero, se compara una serie de enfoques acerca de la temática; luego se describen los elementos comunes de los enfoques, y posteriormente se aplican a un conjunto de técnicas en el análisis de los requisitos funcionales.

Palabras clave
Sistemas de información, aspectos de diálogo, POO, aspectos funcionales, modelamiento.

ABSTRACT
In this work we analyze the techniques proposed in the current methods for specific aspects of user-system dialogue of a software product according to the paradigm of object-oriented development. These methods do not offer enough guidelines about how to model these aspects, and the available techniques need some refinement and elaboration to adjust that task in the software specification process. First, a number of approaches about the subject are compared; then the common elements for the approaches are described, and finally we apply a set of techniques in the analysis of the functional requirements.

Keywords
Information systems, dialogue issues, OOP, functional issues, modeling.

RÉSUMÉ
Dans ce travail on analyse les techniques proposées dans les méthodes actuelles pour spécifier les aspects de dialogue utilisateur-système d'un produit logiciel selon le paradigme de développement orientée objets. Ces méthodes n'offrent pas des suffisantes directives au sujet de comme modeler tels aspects, et las techniques disponibles nécessitent quelque raffinement et élaboration pour adapter ce tâche dans le processus de spécification du logiciel. D'abord, on compare un ensemble d'approches au sujet de ce thème ; après on décrit les éléments communs aux approches, et finalement on applique un group de techniques dans l'analyse des requises fonctionnelles.

Mots-clés
Systèmes d'information, rapports de dialogue, POO, rapports fonctionnels, modélisation.

1. INTRODUCCIÓN

Los métodos actuales de desarrollo orientados por objetos ofrecen un conjunto de técnicas para especificar los diferentes aspectos de los sistemas de información, y más concretamente prestan especial atención a los aspectos estáticos o estructurales –diagrama de relación de objetos–, de comportamiento –diagramas de estado– y de interacción –diagramas de secuencias. Pero para otras, como las reglas de negocio o el conocimiento del dominio que están embebidas en el sistema de información y la funcionalidad requerida por los usuarios –modelado de interacción usuario-sistema–, no prestan la misma atención. Además, las técnicas que ofrecen los métodos actuales no son suficientes para lograr este modelado, por ejemplo, la especificación de un diálogo entre el sistema y el usuario es difícil de modelar sólo con diagramas de estado y diagramas de relación de objetos; y los diagramas de secuencias tampoco son totalmente adecuados, ya que no permiten representar las derivaciones y las interacciones.

UML propone algunas técnicas para modelar el comportamiento del sistema, sin embargo, no es claro cómo interactúan unas con otras ni tampoco cómo se combinan para modelar la interacción usuario-sistema. Además, estas técnicas son muy conceptuales –e informales–, o muy cercanas al diseño del software. En este trabajo se analizan cinco métodos actuales de desarrollo orientado por objetos y se estudian las técnicas que proponen para modelar el comportamiento del sistema y, más concretamente, cómo utilizarlas para modelar la parte del diálogo de un sistema –sección 2. En la sección 3, se comparan esas técnicas y se identifican los elementos comunes, y en la sección 4, se propone un *framework* teórico de las técnicas que va desde la elicitación de requisitos conceptuales, para detallar las especificaciones para la interacción usuario-sistema, hasta indicar cómo utilizar cada una como un refinamiento de la técnica más conceptual. En la sección 5 se presentan las conclusiones y se esbozan posibles nuevas investigaciones.

2. TÉCNICAS ACTUALES PARA MODELAR EL FUNCIONAMIENTO

2.1 UML

Los casos de uso describen los requisitos funcionales mediante la identificación de actores y sus escenarios de uso del sistema, y como tal esta técnica es un candidato válido para modelar la interacción entre el usuario y el sistema. Ofrece algunas posibilidades para la modularización mediante casos de uso que permiten *incluir* y *extender* otros casos de uso. La descripción detallada de un caso de uso está mayormente focalizada en el camino principal de eventos y los posibles caminos alternos. En todos los métodos, los casos de uso apoyan principalmente el diseño del sistema: su función es encontrar los objetos y determinar la estructura de los sistemas [1-9]. El principal problema con esta técnica es su definición informal: no tiene una semántica rigurosa para conceptos como actor o caso de uso, ni para las relaciones *include* y *extend*.

Los casos de uso es un buen punto de partida y es útil para elicitar requisitos, pero para un análisis más detallado de los componentes de diálogo de un sistema se necesita otro tipo de técnica, como los diagramas de secuencias que tienen como función modelar la interacción por medio de mensajes que pasa entre los objetos. Considerando que los casos de uso son bastante conceptuales, esta técnica puede caracterizarse como una técnica de diseño más detallado, que identifica las partes esenciales del código del programa orientado por objetos, como los objetos y los mensajes que éstos intercambian. Sin embargo, los diagramas de secuencias pueden utilizarse en un nivel conceptual superior, por ejemplo, para modelar la interacción entre un usuario y el sistema como un todo. El principal inconveniente de esta técnica es que no permite modelar escenarios alternativos –derivaciones–, iteraciones,... y así sucesivamente. Para modelarlos, UML propone los diagramas de actividades [2], similares a las redes de Petri, y que se focalizan en las actividades que llevan a cabo los objetos; además, permiten modelar ramificaciones, iteraciones y caminos paralelos. Al organizar las actividades en columnas es posible asignar las responsabilidades para las actividades de acuerdo con las unidades o los objetos organizativos. Los diagramas de actividades también pueden modelar flujos de trabajo, sin embargo, no es claro cómo vincularlos a los diagramas de secuencias, por ejemplo, ¿qué se necesita para modelar un diagrama de secuencias para cada escenario alternativo que identifica el diagrama de actividades? Esta pregunta aún no tiene respuesta en el UML [2] ni en los métodos asociados [1-9].

2.2 Fusion

Fusion [3] define la interfaz del sistema mediante la identificación de escenarios de uso. Estos escenarios muestran el flujo de la comunicación entre los agentes –externos– y el sistema, y los documenta mediante diagramas de líneas de tiempo. Estos diagramas son equivalentes gráficamente a los diagramas de secuencias, pero utilizan un nivel conceptual superior: asocian las líneas de vida a conceptos más abstractos como "agente" y "sistema", y utilizan las flechas de comunicación para identificar "unidades" de comunicación de más alto nivel. La fase de análisis no se ocupa de la mensajería interna entre objetos, y define la comunicación solamente en términos de operaciones del sistema. Además, define las operaciones del sistema como eventos de entrada y sus efectos asociados, que son invocados sólo por agentes. Las respuestas del sistema para el agente se llaman mensajes de salida. El modelo de interfaz lo conforma mediante el desarrollo de expresiones de ciclo de vida que generalizan los escenarios de uso. Las expresiones de ciclo de vida las define como expresiones regulares, que construye utilizando operaciones del sistema, mensajes de salida, expresiones de sub-ciclo de vida, secuencias de

operadores, alternativas, iteraciones, e interpolaciones. Además, define cada operación por medio de un esquema de operaciones en el diccionario de datos.

2.3 Syntropy

Syntropy [4] marca una diferencia entre el modelo del mundo real –denominado modelo esencial– y el modelo de software. Según este método, los eventos son elementos fundamentales de la especificación del software, y como tal son uno de los pilares esenciales del modelo. En el modelo del software, describe por medio de escenarios de evento a la interacción entre los agentes externos y el sistema y la documenta de forma tabular. En esa tabla dispone una columna para cada agente, y varias filas consecutivas para eventos consecutivos; en cada fila del evento marca en la columna de cada agente implicado un estímulo por parte del usuario al sistema –mediante un signo de interrogación '?'– o una respuesta del sistema al agente –mediante un signo de exclamación '!'. Cada columna de la tabla es un escenario del evento que describe el comportamiento global del sistema desde la perspectiva de un único agente. Es necesario tener en cuenta que el sistema software no está explícitamente modelado en esta tabla, ya que es equivalente a un diagrama de línea de tiempo o secuencia gráfica, que sólo documenta directamente las secuencias de acontecimientos, alternativas o repeticiones, y no otras estructuras. Syntropy no ofrece una técnica que permita modelar la estructura de interacción; de hecho, particiona la interacción completa entre los objetos del sistema que colaboran para realizar el escenario. Utiliza los diagramas de estado de los objetos individuales para generalizar los escenarios.

2.4 OO-SSADM

OO-SSADM [8] es uno de los pocos métodos que hace una separación explícita entre el modelado del dominio –modelo entidad-evento– y la funcionalidad que el usuario requiere –modelo externo. Además, define como *funciones de usuario* a una composición de eventos y consultas organizados para apoyar al usuario en la realización de alguna tarea o procedimiento, y propone a JSP como una técnica para diseñar el diálogo. Identifica una estructure JSP para las entradas del usuario y otra para las respuestas del sistema; utiliza esta técnica para combinar ambas estructuras en una estructura única que identifica el diálogo usuario-sistema. OO-SSADM manifiesta explícitamente que esa estructura JSP necesariamente no es la mejor estructura de programa. Pero sigue siendo, por supuesto, una especificación válida de la estructura de diálogo. También ofrece un *framework* para el diseño de la interfaz orientada a eventos. Dicho *framework* identifica una serie de interfaces de plantilla de usuario para diferentes tipos de eventos como creación, eliminación y modificación. Estas interfaces se pueden combinar en cuatro modelos de mini-diálogo. La principal característica de estos patrones de diálogo es su estructura lineal, ya que tienen un sólo escenario y ninguna derivación o bucle.

2.5 Modelo de casos de uso jerárquicos HUM

La combinación de casos de uso y diagramas de secuencias en el modelo de funcionalidad del usuario está lejos de ser ideal, ya que persiste una brecha significativa entre los niveles conceptuales de ambas técnicas. La transición a una descripción de la interacción usuario-sistema no es sencilla En realidad no es obvio cómo perfeccionar un caso de uso en uno o varios diagramas de secuencias, aunque el flujo de eventos se describa de una manera estructurada, por ejemplo, por medio de pseudocódigo. El modelo de casos de uso jerárquico –HUM– propuesto por Regnell *et al.* [7] es un intento de cerrar esa brecha. Este modelo adiciona una técnica de modelado de *episodio* entre casos de uso y diagramas de secuencias para cerrar dicha brecha. Refina los casos de uso especificando el proceso como una estructura de episodio. Esta estructura es una combinación de episodios que utiliza las secuencias de operadores, la selección, la iteración, la excepción y la interrupción. A su vez cada episodio puede descomponerse en una nueva estructura de episodio. En esa descomposición del episodio se refinan los nodos de la rama por medio de diagramas de secuencias. Aunque no está explícitamente definido en [7], se supone que los autores intentan que tales episodios-rama no tengan derivaciones o repeticiones, es decir, que tengan una estructura secuencial lineal.

3. ANÁLISIS COMPARATIVO

Cuatro de estos métodos identifican a los *eventos* como los bloques básicos de construcción: UML [2] (p. 227-228) describe el comportamiento de casos de uso como un *flujo de eventos*; Fusion [3] (p. 45) define las operaciones del sistema como *eventos de entrada* y sus efectos asociados; y en Syntropy y OO-SSADM identifican a los eventos en la fase de modelado de dominio y los aplican como bloques de construcción durante la fase externa o de modelado del software. Sin embargo, sólo Syntropy y OO-SSADM identifican los eventos del mundo real durante la fase de modelado de dominio, y los utilizan para expresar el comportamiento del sistema desde la perspectiva del usuario. Además, Fusion define una operación del sistema como un evento de entrada invocado por un agente. En UML y HUM, sin embargo, los eventos a los que se refieren los casos de uso y los diagramas de secuencias pueden ser tanto eventos en el mundo real como del sistema de información.

Todos los métodos analizados expresan la necesidad de estructurar el comportamiento del sistema en función de las operaciones básicas de secuencia, la opción –derivación– y la iteración. UML identifica las discusiones paralelas mediante diagramas de actividades, y Fusion lo hace a través del operador de interpolación. Los otros métodos discuten el paralelismo como un tema aparte, pero no necesariamente relacionado con el concepto de las discusiones paralelas en un diálogo. Sin embargo, es una característica típica de las interfaces de usuario avanzadas con la que los usuarios pueden realizar varias tareas en paralelo con un sistema software

único. Los sistemas de ventanas actuales ofrecen todo el apoyo tecnológico necesario para lograr ese paralelismo, por lo tanto, es una característica esencial en el modelado de diálogo. La Tabla I resume las técnicas de especificación de comportamiento utilizadas en los métodos analizados.

TABLA I
Comparación de las técnicas de especificación de comportamiento

	Nivel conceptual	**Estructura de comportamiento**	**Diseño detallado**	**Bloque básico de construcción**
UML	Casos de Uso	Seudocódigo estructurado en lenguaje natural o en Inglés	Diagramas de secuencias Diagramas de actividades	Evento
Fusion	Escenarios de Uso Diagrama de línea de tiempo	Expresiones de ciclo de vida Operadores: secuencia, selección, repetición, intercalado	Modelo de operación	Operación del sistema –evento de entrada– Mensaje de salida
Syntropy		Escenarios de evento		Evento –estímulos y respuestas–
OO-SSADM	Función de usuario	Mini diálogos estándar Diagramas JSD Operadores: secuencia, selección, repetición		Evento Consulta
HUM	Casos de Uso	Estructura de episodio Operadores: secuencia, selección, repetición, excepción, interrupción	Diagramas de secuencias	Mensaje

(Fuente: Autor)

4. FRAMEWORK TEÓRICO PARA MODELAR LA INTERACCIÓN USUARIO-SISTEMA

Durante la fase de elicitación se requieren técnicas que permitan identificar todas las funciones de usuario necesarias. Una posible fuente de información es buscar en los procesos de negocio y refinarlos al nivel de tareas elementales. Estas tareas se categorizan de forma totalmente manual, totalmente automática o interactiva. Tanto las tareas automatizadas como las interactivas requerirán alguna función de usuario en el sistema software. Estas funciones complejas de usuario se pueden especificar como casos de uso y refinar a la granularidad adecuada utilizando las relaciones *includes* y *extends*.

Para obtener una especificación formal y sin ambigüedades del aspecto de comportamiento de una función de usuario se define cada caso de uso o función de usuario compleja como una composición de las funciones de usuario simples. Esta composición se define utilizando secuencia, selección, iteración y la composición paralela. Las funciones complejas se descomponen hasta obtener una granularidad de funciones de usuario "simples". Las funciones de usuario simples se caracterizan por el hecho de que tienen un diálogo lineal sin escenarios alternativos, iteraciones o hilos paralelos. Por lo general, como una función simple puede ser realizada por una ventana en una interfaz gráfica de usuario, o por medio de una secuencia de ventanas modelo. Las relaciones "*includes*" y "*extends*" pueden formalizarse en consecuencia: *include* se refiere a una sub-estructura, mientras que *extend* se refiere a una estructura opcional, consisten en elegir entre "no hacer nada" y la realización del caso de uso. Por último, cada función simple se documenta por medio de un conjunto finito de escenarios: uno para la secuencia básica y otro para cada posible excepción.

Tanto en OO-SSADM como en Syntropy, los eventos del mundo real se identifican como componentes básicos del modelo de dominio o del modelo del mundo real. Los eventos son unidades atómicas de acción ya que representan las cosas que suceden en el mundo. Sin eventos no pasaría nada, ya que conforman la información y los objetos que crean eventos; la información y los objetos que son modificados –eventos modificar–; y desaparecen del universo del discurso –eventos de finalización. Los eventos no son objetos, sin embargo, es posible registrar el hecho de que un evento ha ocurrido mediante el registro de la ocurrencia de dicho evento como un objeto. Por ejemplo, en un entorno bancario, "retirar dinero" es un evento que modifica el estado de un objeto "CUENTA BANCARIA". Se puede hacer un seguimiento a todos los retiros mediante la definición de "RETIRO" como un tipo de objeto adicional. A partir de entonces, un evento *retirar* tendrá un doble efecto: *modificar* el estado de una cuenta y *crear* un retiro. En la fase de análisis esto sería irrelevante para determinar la forma en que ambos objetos son notificados de la ocurrencia del evento de retiro. Por lo tanto, se asume –al igual que en Sintropy y OO-SSADM– que los eventos son transmitidos.

La separación de los eventos del mundo real de los eventos del sistema la información permite una visión más orientada hacia el usuario y a la tarea del diseño del sistema de información. Los eventos del mundo real son aquellos eventos que ocurren en el mundo real, incluso si no hay un sistema de información alrededor. Los eventos del sistema de información están directamente relacionados con la presencia de un sistema computarizado; están diseñados para permitir al usuario externo registrar la ocurrencia o de invocar un evento del mundo real. Por ejemplo, el uso

de un ATM para retirar dinero de una cuenta invocará el evento del mundo real "retirar" a través de varios eventos del sistema de información, como "insertar tarjeta", “digitar clave", "seleccionar/digitar valor", y así sucesivamente. Una vez que los eventos han sido identificados en el modelo de dominio, el modelo completo puede considerarse como un componente, cuya interfaz es el conjunto de todos los eventos que permiten crear, modificar y actualizar la información contenida en el modelo de dominio. Entonces, las funciones de usuario son una forma para invocar estos eventos del mundo real. La función de usuario traducirá los eventos del sistema de información como los clics del ratón y las pulsaciones de teclas en la invocación de uno o más eventos del modelo de dominio.

El meta-modelo de la Fig. 1 representa los componentes de una especificación e identifica las técnicas de modelado adecuadas para cada componente. El siguiente ejemplo bancario simplificado muestra cómo usar las técnicas propuestas para modelar una función de usuario compleja. El modelo de dominio simplificado del banco contiene los tipos de objetos CLIENTE y CUENTA relacionados por una asociación uno a muchos: un cliente tiene cero a muchas cuentas y una cuenta se mantiene exactamente por un cliente. Los eventos del negocio para CLIENTES son *crear_cliente, modificar_cliente, cliente_final*, y para CUENTA son *abrir, depositar, retirar, cerrar*.

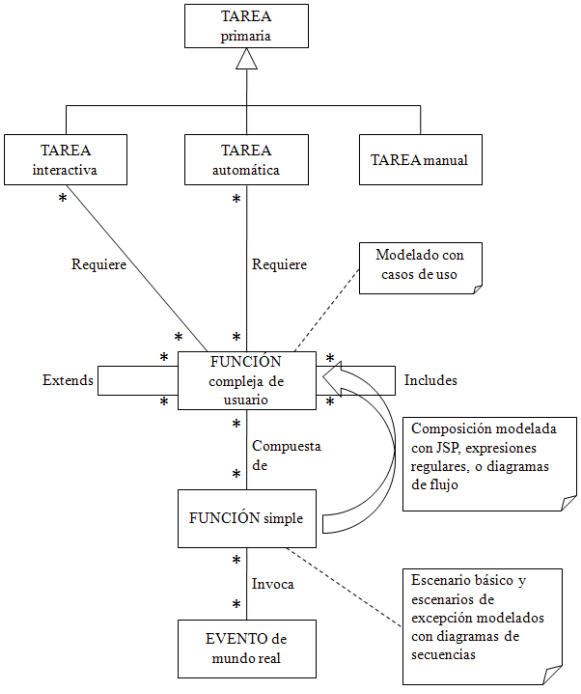

Fig. 1 Meta-modelo para los requisitos de funcionalidad
(Fuente: Autor)

El retiro de dinero en efectivo a través de un ATM es una función de usuario compleja que eventualmente invoca un evento *retirar*. La Fig. 2 representa la estructura de comportamiento de esta función de acuerdo con las anotaciones del HUM. La Tabla II representa la misma estructura con una expresión regular, como en Fusion. Esta definición utiliza una función recursiva de la sub-expresión SEGUNDO_PASO. Sin embargo, la definición recursiva se puede evitar utilizando el episodio vacío denotado con "1", como se muestra en la Tabla III.

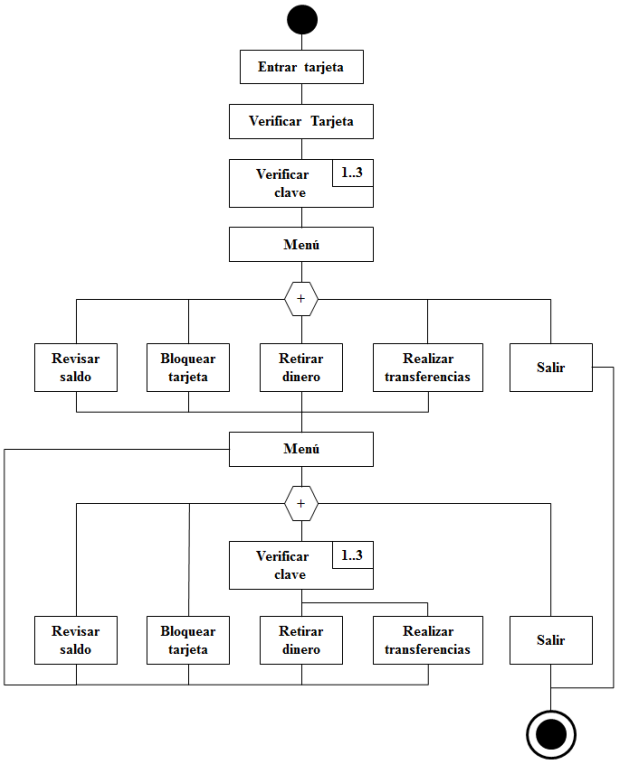

Fig. 2 Estructura del episodio retirar dinero en un ATM
(Fuente: Autor)

TABLA II
Estructura de la función Retirar_dinero en un ATM especificada con expresiones regulares

```
Usar_ATM = Entrar_tarjeta.(Verificar_tarjeta)
  (Verificar_clave)^1..3.Menu.
  (Revisar_saldo.SEGUNDO_PASO
  + Retirar_dinero.SEGUNDO_PASO
  + Salir)

SEGUNDO_PASO = Menu.
  (Revisar_saldo.SEGUNDO_PASO
  + (Verificar_clave)^1..3.Retirar_dinero.SEGUNDO_PASO
  + Salir)
```

(Fuente: Autor)

Tabla III
Estructura de la función con definiciones recursivas

```
Usar_ATM = Entrar_tarjeta.(Verificar_tarjeta)
  (Verificar_clave)^1..3.Menu.
  (Revisar_saldo + Retirar_dinero + 1)
  .SEGUNDO_PASO*
  .Salir

SEGUNDO_PASO = Menu.
  (Revisar_saldo + (Verificar_clave)^1..3.Retirar_dinero + 1)
```

(Fuente: Autor)

Como era de esperar, la especificación de la estructura de comportamiento por medio de diagramas de flujo tiende a ser menos estructurada que la descripción equivalente por medio de expresiones regulares. Note también cómo la especificación formal de la estructura

de diálogo permite identificar fácilmente las partes reutilizables del diálogo. El refinamiento de los aspectos del diálogo de cada episodio puede hacerse utilizando un diagrama de secuencias de alto nivel. En tal diagrama, se incluyen todos los agentes externos, la función y el modelo de dominio. Además, en este diagrama se distingue claramente entre "eventos del sistema de información" y "eventos de dominio". Los eventos del primer tipo son los que cruzan los límites del sistema de información, y los otros son los eventos generados por la función y transmitidos a los objetos del dominio. La Fig. 3 muestra un diagrama de secuencias para la función simple *Retirar_dinero*.

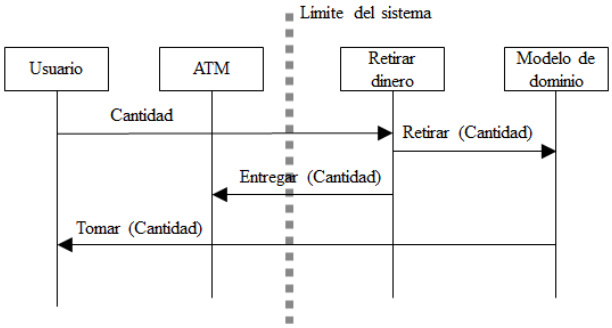

Fig. 3 Diagrama de secuencias para Retirar_dinero
(Fuente: Autor)

En este ejemplo, se supone que el Retiro se realiza correctamente. La especificación de todas las posibles excepciones a la secuencia lineal de eventos de base es un refinamiento mayor de funciones simples. Esto significa que cada excepción identificará un punto de salida adicional en caso de que alguna acción falle. En el ejemplo anterior, una posible excepción es la negación de la operación de Retiro por el modelo de dominio de acuerdo con algunas reglas de negocio – por ejemplo, saldo insuficiente. La excepción se puede especificar como un escenario alternativo para el escenario base. En principio, cada función simple tendrá un escenario base y cualquier número de "escenarios de excepción", documentados por medio de un diagrama de secuencias. El escenario de excepción para Retirar_dinero se presenta en la Fig. 4.

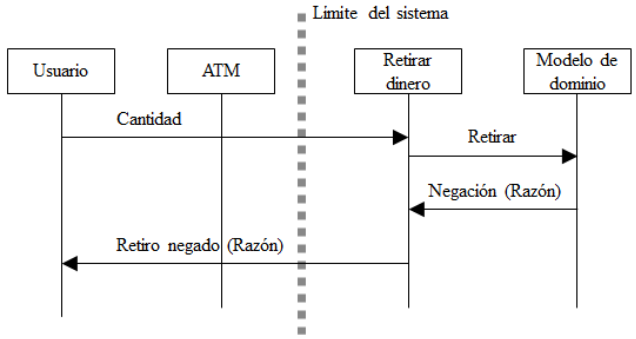

Fig. 4 Escenario de excepción para *Retirar_dinero*
(Fuente: Autor)

Estos bloques básicos de construcción se pueden reutilizar para formar más diálogos complejos. En el ejemplo dado, Revisar_saldo y Retirar_dinero son las funciones simples que se vuelven a utilizar en la función compleja Usar_ATM. Un servicio que permite retirar dinero en el mostrador reusará los mismos bloques de construcción y tal vez, además, la función de Revisar_cliente.

4. CONCLUSIONES

El modelo de diálogo usuario-sistema no se encuentra muy elaborado en los métodos actuales de modelado orientado por objetos. En la práctica, el diálogo de usuario a menudo es diseñado, probablemente, directamente por medio de prototipos. Sin embargo, un enfoque más sistemático independiente de la eventual tecnología de implementación puede mejorar el diseño de interacción usuario sistema.

Aunque el estudio al conjunto de métodos no fue exhaustivo, todos los métodos investigados muestran un patrón similar. Además de la identificación de eventos, se requiere algún método de especificación formal para el establecimiento de la estructura de diálogo, que tiene que ser descompuesta hasta los componentes de diálogo simples definidos y que presentan una secuencia de eventos lineales de base. Para esta secuencia de eventos se pueden añadir las posibles excepciones como caminos alternativos. Como resultado de esto, cada función simple puede describirse mediante un conjunto finito de diagramas de secuencias: una para el escenario base y una para cada posible excepción. Identificar estos componentes de diálogo constituye la base de fomento a la reutilización. Si los eventos del mundo real se identifican como elementos de base de un modelo de dominio, se pueden crear funciones complejas de manera ascendente. Cada evento de dominio dará lugar a la definición de una función simple que invoca dicho evento. Además, deben identificarse las consultas de base, por lo general dos por objeto de dominio: una para ver la lista de objetos en una clase de objeto de dominio y una para ver los detalles de un objeto aislado. Las funciones simples identificadas de esta forma se pueden componer para formar más diálogos de usuario avanzados.

Aunque las técnicas propuestas fueron elaboradas para el modelo diálogos en línea, también pueden utilizarse en cierta medida para modelar servicios fuera de línea. Las investigaciones futuras deberían evaluar la usabilidad de las técnicas en esta área. Sin embargo, la primera preocupación será la elaboración matemática de las técnicas propuestas por medio del álgebra de procesos. En [10] se demuestra que el modelado de dominio puede ser suficiente formalizado por medio de un álgebra de procesos similar a CSP [5]. Sería interesante para elaborar esta álgebra de procesos que se permitiera también formalizar la funcionalidad de las técnicas de modelado. Además, está planeada una evaluación práctica más avanzada de las técnicas por medio de ejemplos del mundo real.

REFERENCIAS

[1] G. Booch. “Object Oriented Analysis and Design with Applications”. Redwood City: Benjamin/Cummings, 1994.

[2] G. Booch, J. Rumbaugh & I. Jacobson. “The unified modeling language user guide”. USA: Addison Wesley, 1999.
[3] D. Coleman. “Object-oriented development: The FUSION method”. New York: Prentice Hall, 1994.
[4] S. Cook & J. Daniels. “Designing object systems: object-oriented modeling with Syntropy”. USA: Prentice Hall, 1994.
[5] C. A. R. Hoare. “Communicating Sequential Processes CPS”. USA: Prentice Hall International, Series in Computing Science, 1985.
[6] I. Jacobson, M. Christerson & P. Jonsson. “Object-Oriented Software Engineering, A use Case Driven Approach”. USA: Addison-Wesley, 1992.
[7] B. Regnell, M. Andersson & J. Bergstrand. “A hierarchical use case model with graphical representation”. *Proceedings of the IEEE international symposium and workshop on engineering of computer based systems ECBS'96.* Friedrichshafen, Germany, March 1996.
[8] K. Robinson & G. Berrisford. “Object-oriented SSADM”. New York: Prentice Hall, 1994.
[9] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy & W. Lorensen. “Object Oriented Modeling and Design”. USA: Prentice Hall International, 1991.
[10] M. Snoeck & G. Dedene. “Existence Dependency: The key to semantic integrity between structural and behavioural aspects of object types”. *IEEE Transactions on Software Engineering*, Vol. 24, No. 24, pp.233-251. April 1998.