Desarrollo y generación de interfaces de usuario a partir de técnicas de análisis de tareas y casos de uso

María Dolores Lozano[†], Pascual González[†], Isidro Ramos[‡], Francisco Montero[†], Jose Pascual Molina[†]

[†]Departamento de Informática
Escuela Politécnica Superior de Albacete
Universidad de Castilla-La Mancha
-ALBACETE-
[mlozano, pgonzalez]@info-ab.uclm.es

[‡]Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n
-VALENCIA-
iramos@dsic.upv.es

Resumen

El objetivo de todo desarrollo de sistemas de información es obtener un sistema que permita a los usuarios finales alcanzar sus objetivos y llevar a cabo, de forma efectiva y eficiente, las tareas necesarias para conseguirlos. Para lograr esto se hace necesario, no sólo tener en cuenta los requisitos del sistema, sino además, incorporar nuevas técnicas que ayuden a captar las necesidades del usuario incorporando criterios de usabilidad, para conseguir además desarrollar interfaces de usuario intuitivas y fáciles de usar que ayuden al usuario final a sacar el máximo provecho de los sistemas informáticos. Con este objetivo en este trabajo se propone un entorno de desarrollo de interfaces de usuario partiendo de técnicas de casos de uso y análisis de tareas.

Palabras clave: Interfaces de Usuario, Interacción Hombre-Máquina, Orientación a Objetos, Técnicas de Especificación y Modelado, Sistemas Interactivos.

1. Introducción

El principal objetivo a la hora de construir una aplicación informática es facilitar a los usuarios finales alcanzar sus objetivos y llevar a cabo sus tareas con ese sistema de forma efectiva y eficiente. Por tanto, en este proceso se hace necesario, no sólo tener en cuenta los requisitos del sistema, que nos llevarían a la obtención de un buen sistema, robusto y completo desde el punto de vista informático, pero que tal vez el usuario final no fuera capaz de utilizar de forma correcta al no haber tenido en cuenta sus necesidades concretas como usuario final, sino que se hace necesario incorporar en este proceso nuevas técnicas que ayuden a captar las necesidades del usuario incorporando criterios de usabilidad, para conseguir además desarrollar interfaces de usuario intuitivas y fáciles de usar que ayuden al usuario final a sacar el máximo provecho de los sistemas informáticos.

Esta forma de construir los sistemas se conoce como diseño orientado al usuario. El principal objetivo de esta forma de desarrollar software es hacer los sistemas más usables y accesibles a los usuarios finales, siendo éstos la pieza principal en este proceso de desarrollo. De acuerdo a la ISO/DIS 13407 [ISO-98], el proceso de diseño orientado al usuario implica:

- Activa participación del usuario así como una clara comprensión del contexto de uso y de las tareas y requisitos del usuario final.
- Una apropiada distribución de funciones entre los usuarios y la tecnología, especificando qué funciones deben ser llevadas a cabo por los usuarios y cuales son de índole tecnológica.
- La iteración de las soluciones de diseño en las que la retroalimentación del usuario es esencial como fuente de información.
- Una perspectiva de diseño multidisciplinar que requiere de gran variedad de especialidades o disciplinas (desarrolladores de software, expertos en usabilidad, especialistas en ergonomía, etc).

Una de las aproximaciones más ampliamente utilizadas por muchos grupos de investigación en el área HCI (Human Computer Interaction) para llevar a cabo esto es el análisis de tareas. El propósito de esta técnica es desarrollar modelos genéricos y por tanto abstractos de las tareas del usuario final. Otra técnica utilizada con este fin son los casos de uso.

En este trabajo se propone un entorno de desarrollo de interfaces de usuario basado en modelos [Pue94b] dentro del proceso de desarrollo de software orientado a objetos, que satisfaga los objetivos comentados anteriormente. En este proceso de desarrollo se hace uso de técnicas de casos de uso y análisis de tareas y se tienen en cuenta criterios de usabilidad. Así, se propone un modelo de interfaz de usuario que modela la interacción del usuario con el sistema y proporciona una representación formal del diseño de interfaz. Este modelo incluye además la generación de diagramas gráficos para representar los diferentes aspectos de la IU y permite generar de forma automática la IU final a partir de los modelos definidos.

Para cubrir estos aspectos, el resto del artículo se estructura de la siguiente manera: en el segundo apartado se introduce brevemente el estado del arte para situar este trabajo dentro del contexto de otros trabajos similares; en el siguiente apartado se analizan las técnicas de casos de uso y análisis de tareas y se indica su uso en el entorno propuesto. A continuación se presenta brevemente OASIS[1] [Let98], el modelo orientado a objetos que establece el marco de trabajo donde se propone el entorno de desarrollo de interfaces de usuario, entorno que se describe en el apartado cuarto. Finalmente se introducen los criterios de usabilidad necesarios en este proceso de desarrollo de interfaces de usuario.

2. Estado del Arte

De entre las herramientas software de generación de interfaces de usuario basadas en modelos [Pue94b] construidas, podemos destacar las siguientes: UIDE [Fol95], ADEPT [Joh95], HUMANOID [Sze93], ITS [Wie90], MECANO [Pue94a], MOBI-D [Pue96], AME [Mär96], FUSE [Lon96], MASTERMIND [Sze96], TRIDENT [Bod95], [Bod96], GENIUS [Jan93], JANUS [Bal96], TADEUS [Elw95], IDEAS, modelo propuesto para OASIS.

A modo de resumen, en la siguiente tabla se indica el alcance de cada una de ellas:

	Casos de Uso	Tareas	Dominio	Usuario	Diálogo	Presentación
UIDE			✖ Clases C++			
AME			✖ Modelo OO			
JANUS			✖ Modelo OO			
GENIUS			✖ Modelo E-R		✖	
ADEPT		✖		✖		
MASTER MIND		✖	✖ CORBA IDL			✖
MECANO			✖ Modelo OO			
MOBI-D		✖	✖ Modelo Objetual MIMIC	✖	✖	✖
TRIDENT		✖	✖ Modelo E-R			
FUSE		✖	✖ Especif. Algebraica	✖	✖	
TADEUS		✖	✖ Modelo OO	✖	✖	
OVID		✖	✖ Modelo OO	✖	✖	✖
IDEAS	✖	✖	✖ Modelo OO	✖	✖	✖

Tabla1. Modelos declarativos y su aplicación en diferentes entornos de desarrollo de IUs

[1] ***O**pen and **A**ctive **S**pecification of **I**nformation **S**ystems*

3. Casos de Uso y Análisis de Tareas

Los casos de uso son una técnica ampliamente utilizada en ingeniería de requisitos orientada a objetos ya que proporcionan un buen mecanismo para determinar los límites de un sistema así como para describir requisitos. Los casos de uso describen los sistemas desde el punto de vista del usuario en términos de los requisitos funcionales y por tanto se pueden considerar como el punto de partida para poder especificar las necesidades de los usuarios finales. Sin embargo, una de sus principales deficiencias es que no capturan requisitos no funcionales tales como fiabilidad, eficiencia, mantenibilidad, portabilidad y sobre todo usabilidad, que son sin duda factores críticos para la aceptación de un sistema por parte de los usuarios finales, y tampoco sirven para capturar las necesidades de los usuarios, aspecto esencial a la hora de desarrollar interfaces de usuario.

Los casos de uso fueron introducidos en el método OOSE (Object-Oriented Software Engineering) por Jacobson [Jac92] y servían principalmente para capturar la funcionalidad del sistema (requisitos funcionales) desde el punto de vista del usuario final y para asegurar la trazabilidad de los requisitos, es decir, servían de hilo conductor a lo largo del proceso de desarrollo del software.

Los casos de uso normalmente se emplean, y así se hace en este trabajo, en la fase de definición de requisitos donde su papel es identificar la forma en la que será usado el futuro sistema.

Por otra parte, el Análisis de Tareas es una técnica utilizada en el análisis de requisitos de la interfaz de usuario. El análisis de tareas proporciona información acerca de cómo los usuarios llevan normalmente a cabo su trabajo y sirve para identificar la funcionalidad que debería proporcionar la interfaz de usuario del futuro sistema.

Ambos son conceptos muchas veces confundidos, por lo que en este apartado se pretende clarificar y distinguir el uso que se hace en este trabajo de estos conceptos.

Las técnicas de análisis de casos de uso normalmente son utilizadas por ingenieros de software involucrados en el desarrollo de sistemas orientados a objetos, mientras que las técnicas de análisis de tareas se utilizan principalmente en el área HCI con relación al análisis de requisitos para el desarrollo de interfaces de usuario. En este trabajo se utilizan para identificar las tareas que debe llevar a cabo el usuario para conseguir sus objetivos. Estas tareas se recogen en el modelo de tareas, uno de los 5 modelos que constituyen el modelo de interfaz propuesto.

El término “caso de uso” no tiene una definición fija, de hecho un reciente estudio [Hur97], ha identificado mas de treinta variantes de casos de uso y técnicas relacionadas. En este trabajo se utiliza una variante de la propuesta por Cockburn [Coc97], que es muy similar a la propuesta originalmente en [Jac92] pero mucho más fácil de comprender y utilizar.

Cockburn proporciona las siguientes definiciones:

Un *escenario* es una secuencia de interacciones que ocurren bajo ciertas condiciones para conseguir el objetivo del actor principal, y que tienen un resultado determinado con respecto a ese objetivo.

Las interacciones comienzan a partir de una acción de disparo (trigger) y continúan hasta que el objetivo es alcanzado o abandonado por alguna razón.

Un *caso de uso* es una colección de posibles escenarios entre el sistema y los actores externos, caracterizados por el objetivo del actor principal y que muestra cómo dicho objetivo puede ser completado con éxito o puede fallar.

Destacar que al ser el objetivo el hilo conductor a la hora de identificar los casos de uso según la aproximación de Cockburn, hace que la correspondencia con el análisis de tareas donde el énfasis está en comprender las tareas del usuario en términos de las actividades que tiene que llevar a cabo para conseguir el objetivo perseguido sea mucho más fácil y natural. En este trabajo se considera que un caso de uso se corresponde con una o más tareas de usuario.

4. OASIS como Lenguaje de Especificación de Interfaces de Usuario

El modelo OASIS provee una rica expresividad semántica para la especificación de modelos conceptuales bajo el paradigma orientado a objetos. La presentación de OASIS 3.0 como lenguaje, junto con su gramática se encuentra detallada en [Let98]. En este apartado y a modo de presentación, se describen brevemente sus características principales.

Uno de los objetivos a realizar en el trabajo de investigación que se lleva a cabo es precisamente realizar las extensiones, tanto sintácticas como semánticas, necesarias para que sirva como soporte

formal del proceso de desarrollo de interfaces de usuario.

En OASIS, los bloques básicos de construcción en un sistema de información son los *objetos*. Un **objeto** OASIS es un proceso observable [Ram93] que encapsula estado y comportamiento. Cada objeto tiene un identificador (oid) que será único y lo distingue de cualquier otro objeto que haya existido, exista o pueda existir. Los objetos poseen propiedades, representadas en el modelo a través de la noción de *atributo.*

El estado de un objeto viene dado por el conjunto de valores de sus atributos, de forma que un cambio en alguno de dichos valores representará un cambio de estado. La causa de que se produzca un cambio de estado en un objeto se denomina *evento*. La ocurrencia de un evento en la vida de un objeto sólo tendrá éxito si el objeto se encuentra en un estado que verifica la *precondición* asociada a dicho evento. Además, no todos los estados de los objetos van a ser estados permitidos; en este sentido, el modelo permite la definición de *restricciones de integridad* que definen precisamente qué estados serán aceptables en la vida de los objetos, en términos de valores permitidos para los atributos.

Los objetos no están aislados. La interacción entre objetos se modela en OASIS de dos formas: mediante compartición de eventos y relaciones de disparo. Las relaciones de disparo, representadas en el modelo OASIS por *triggers*, introducen actividad en la sociedad de objetos, permitiendo que un objeto actúe como agente de un evento de otro objeto cuando se satisfagan en él determinadas condiciones.

Los objetos que comparten las mismas propiedades se agrupan en *clases*. Un objeto individual es una *instancia* o ejemplar de una clase. En este sentido, la sociedad de objetos de nuestro modelo OASIS está compuesta de las siguientes clases de objetos:

Las **clases primitivas** o dominios están constituidas por "objetos" que tienen un único estado y que siempre existen. En general, abarcan el dominio de los tipos abstractos de datos (TAD). La sociedad de objetos se construirá tomándolos como nivel estructural básico sobre el que se declaran las demás clases de objetos.

A partir de los dominios se construyen las **clases elementales**. Estas vienen caracterizadas por el tipo que una colección de objetos comparte. En el tipo se definen los atributos, los eventos, las restricciones de integridad, las precondiciones, las relaciones de disparo, las transacciones y la parte proceso que caracterizan a los objetos de la clase.

Mediante un mecanismo constructivo, se definen las **clases complejas** a partir de las clases elementales o de otras clases complejas definidas de antemano, aplicando sobre ellas los operadores de clases**.** Los operadores más utilizados son la *especialización, agregación*, *generalización*, *asociación* y la *composición paralela*.

El modelo objetual visto está soportado por el lenguaje OASIS. OASIS es un lenguaje formal de especificación para sistemas de información abiertos y activos, según el modelo OO presentado. El marco formal viene dado por la lógica dinámica, propuesta por Harel [Har84]. En este trabajo, se utiliza la versión OASIS 3.0 [Let98], basada en la lógica dinámica que permite, entre otras cosas, representar los operadores de obligación, prohibición y permiso usados en lógica deóntica.

La especificación formal de la interfaz de usuario se abordará teniendo en cuenta el nivel de abstracción donde nos encontremos. Así se producirán unas especificaciones a nivel de requisitos y análisis para formalizar los modelos correspondientes a estas fases y otras especificaciones a nivel de diseño, donde se formaliza el diseño de la interfaz.

5. IDEAS[2]: Desarrollo de Interfaces de Usuario en un Entorno de Producción Automática de Software Orientado a Objetos

El objetivo de este trabajo es la integración de un Modelo de Interfaz de Usuario dentro del entorno de desarrollo de software definido por el modelo OASIS presentado en el apartado anterior, a partir del cual se pueda generar la interfaz de la aplicación.

El proceso de especificación de la Interfaz de Usuario se realiza de forma paralela al desarrollo de lo que es la aplicación en sí. Este proceso de desarrollo se puede ver en la figura 1.

De esta forma, IDEAS [Loz99b, Loz00] pretende ser un sistema de desarrollo automático de Interfaces de Usuario integrado en el marco de trabajo de OASIS para soportar de forma automática la producción de interfaces de usuario de alta calidad.

[2] ***I**nterface **D**evelopment **E**nvironment within OA**S**I**S***

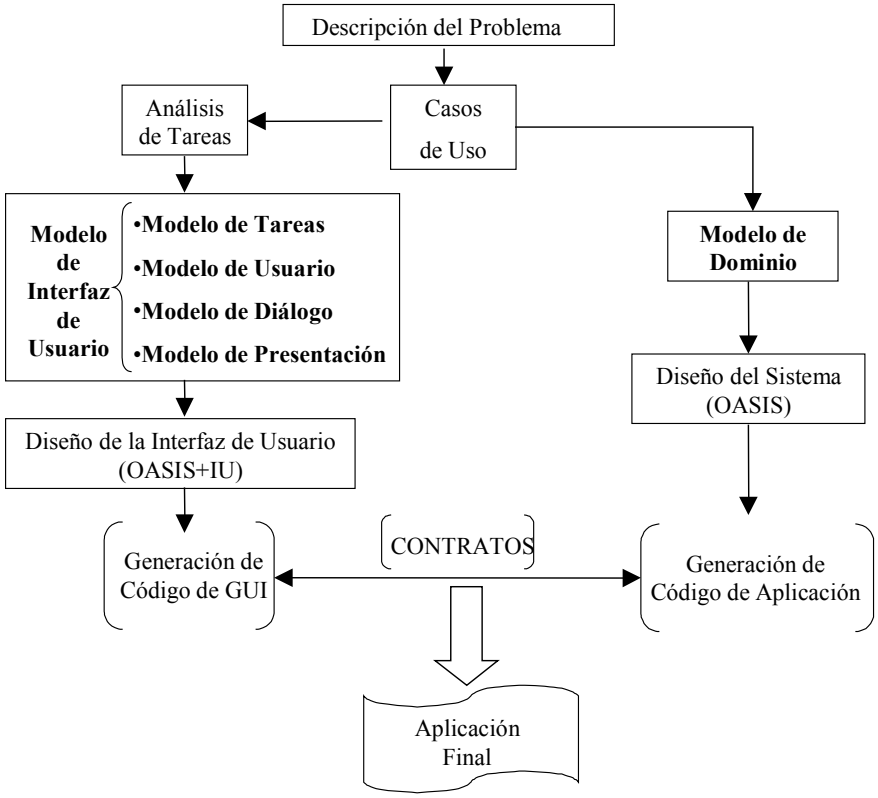

Figura 1. Integración del Proceso de Desarrollo de Interfaces de Usuario en el Marco de OASIS.

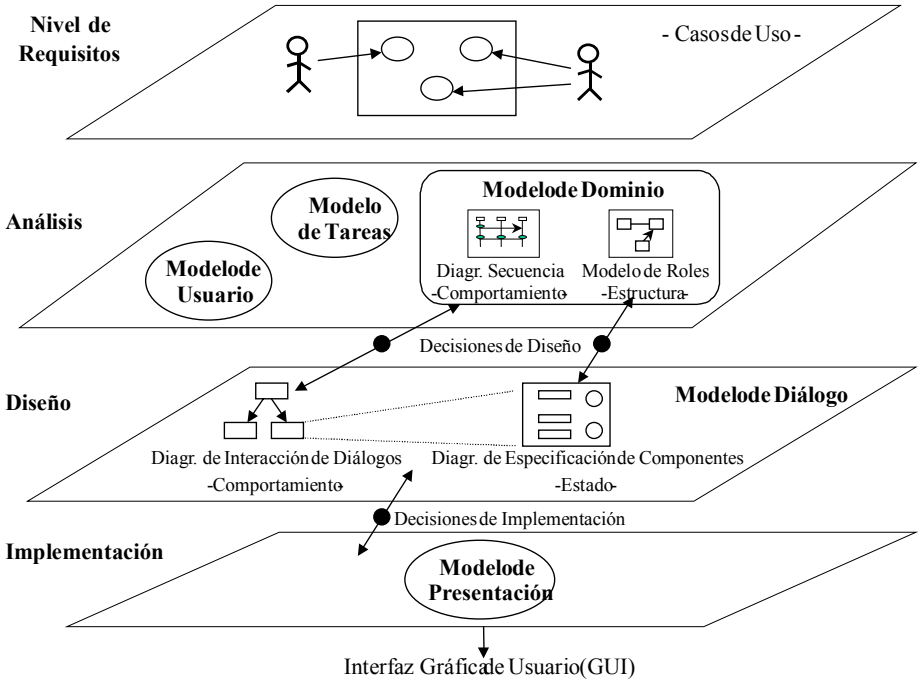

Figura 1. Proceso de Desarrollo de Interfaces de Usuario Propuesto.

A continuación describimos este proceso de desarrollo por niveles y siguiendo el orden en el que se van desarrollando los diferentes modelos que conforman nuestra propuesta. En la figura 2 se representa este desarrollo.

A nivel de requisitos se emplea la técnica de casos de uso [Jac92]. En esta primera etapa y en relación con cada caso de uso se van identificando los diferentes tipos de usuarios del sistema. Posteriormente cada caso de uso se refina y se especifican las políticas del negocio, entidades y actores que participan en él.

A nivel del análisis se generan, en primer lugar el Modelo de Tareas, identificando para cada caso de uso de la fase anterior la/s tarea/s correspondiente/s.

A continuación se realiza el Modelo de Dominio, que en este caso viene dado por el Diagrama de Secuencia asociado a cada uno de los casos de uso (tareas) identificados previamente, que define el comportamiento del sistema, y el Modelo de Roles, que define la estructura de las clases que intervienen en el diagrama de secuencia asociado junto con la relación existente entre ellas especificando el rol que desempeña cada una de las clases en ese caso concreto.

Una vez definidos el modelo de tareas y el modelo de dominio, se define el Modelo de Usuario. Para cada tipo de usuario del sistema se establece el subconjunto de tareas que le están permitidas llevar a cabo. A su vez para cada una de esas tareas se establece una proyección de las acciones que dentro de esa tarea puede o no acometer. Y por último y de acuerdo a las características particulares (independientes de la aplicación) del usuario se establece la forma más adecuada de mostrar la información.

A nivel de diseño se elabora el Modelo de Diálogo [Loz99a]. Hasta ahora los modelos generados no contienen ningún aspecto gráfico o de contenido de la interfaz de usuario. Es a partir de aquí cuando se empiezan a diseñar estos aspectos y se hace especial hincapié en cómo se llevará a cabo la interacción del usuario con el sistema.

El modelo de diálogo consta de dos diagramas: el Diagrama de Interacción de Diálogos y el Diagrama de Especificación de Componentes. El primero especifica el comportamiento de la IU, es decir, las transiciones de ventanas o diálogos que van teniendo lugar en función de las selecciones que sobre la IU va haciendo el usuario. Para generar este diagrama se tiene en cuenta el Diagrama de Secuencia elaborado en la etapa de análisis. En función de las decisiones de diseño que tome el diseñador de la IU se establece la secuencia de ventanas o diálogos necesarios para llevar a cabo el comportamiento requerido. Un ejemplo de este diagrama se puede ver en la siguiente figura.

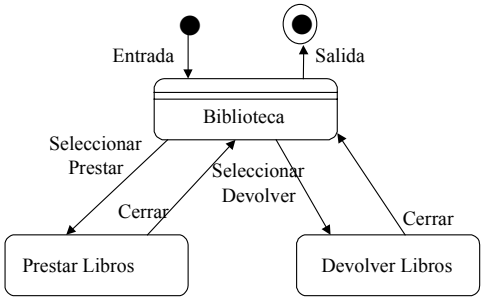

Figura 3. Diagrama de Interacción de Diálogos.

El segundo diagrama consiste en establecer, para cada una de las ventanas o diálogos resultantes del primer diagrama, el conjunto de herramientas de control y de visualización necesarias para dotarlos de la funcionalidad requerida para llevar a cabo la tarea de usuario en cuestión. Este segundo diagrama define el estado de la IU tal y como se muestra en la figura 4.

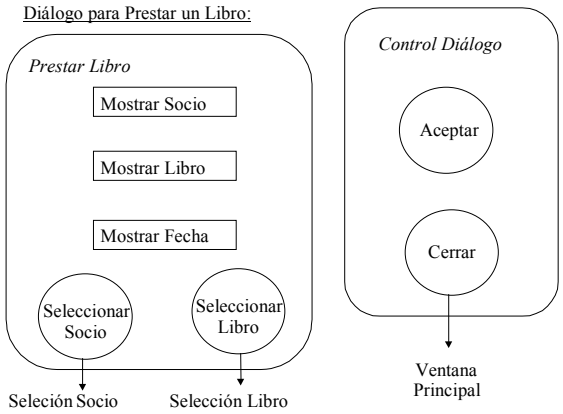

Figura 4. Diagrama de Especificación de Componentes.

A nivel de implementación se genera el Modelo de Presentación. En este caso y en función de las decisiones de implementación que se tomen, la plataforma final de implementación y de acuerdo a la guía de estilo seguida, se especifican los objetos de interacción concretos que formarán parte de la interfaz gráfica de usuario (GUI).

6. Usabilidad. Aspecto esencial en el desarrollo de interfaces de usuario

El objetivo principal a la hora de construir software es facilitar a los usuarios la consecución de objetivos concretos de forma sencilla. Esto es, conseguir que el sistema sea fácil de aprender y de recordar su manejo, sea útil, es decir, contenga las funciones que los usuarios necesitan en su trabajo, y que su manejo sea fácil y agradable. En definitiva el objetivo básico es conseguir un sistema con un alto grado de *"usabilidad"* [Gou85].

Los sistemas construidos con criterios de usabilidad tienen una serie de ventajas tales como la mejora de la productividad, la reducción del coste y del tiempo de aprendizaje, y el incremento de la autonomía de los usuarios finales.

Es más, Landauer [Lan95] observó que el 80% de los costes de mantenimiento – lo que representa el 80% del total de los costes del desarrollo de software – se deben a problemas de los usuarios con el sistema y no a problemas técnicos. Además indica que el 64% de estos costes están relacionados con problemas de usabilidad. Además, varios estudios sobre el tema de la productividad en herramientas software concluyen que la facilidad de uso y aprendizaje son unas de las razones más importante a la hora de incrementar de forma efectiva la productividad de los usuarios del software.

Esta situación resalta la importancia de la usabilidad y justifica la necesidad de incorporarla antes, durante y después del desarrollo de sistemas software. Esto significa que el desarrollo de las aplicaciones informáticas es una actividad multidisciplinar que incorpora factores humanos y técnicas y conocimientos ergonómicos con el objetivo de mejorar las condiciones de trabajo del usuario final.

Como plantea B. Shneiderman [Shn98], dentro de lo que se denomina "ingeniería de la usabilidad", una de las bases para alcanzar altas cotas de calidad es utilizar métodos de diseño iterativo que permitan realizar pruebas tempranas sobre prototipos, revisiones basadas en la retroalimentación de las observaciones y reacciones de los usuarios y los refinamientos sucesivos propuestos por los expertos en usabilidad. A su vez, J. Nielsen [Nie93] plantea que el modo más barato de aplicar con éxito los criterios de usabilidad es realizar el mayor esfuerzo posible antes de que el proceso de diseño de la presentación de la interfaz comience. En este aspecto lo más importante es realizar un estudio previo que permita "conocer a los usuarios". Este conocimiento no sólo se basa en estudiar sus características o habilidades individuales sino que también es necesario conocer las tareas que necesitan realizar. Igualmente dentro del ámbito de la ingeniería de la usabilidad es importante que los usuarios puedan realizar evaluaciones (test de usabilidad) sobre prototipos del sistema final, con lo que pueden aportar mejorar a incorporar en la siguiente versión.

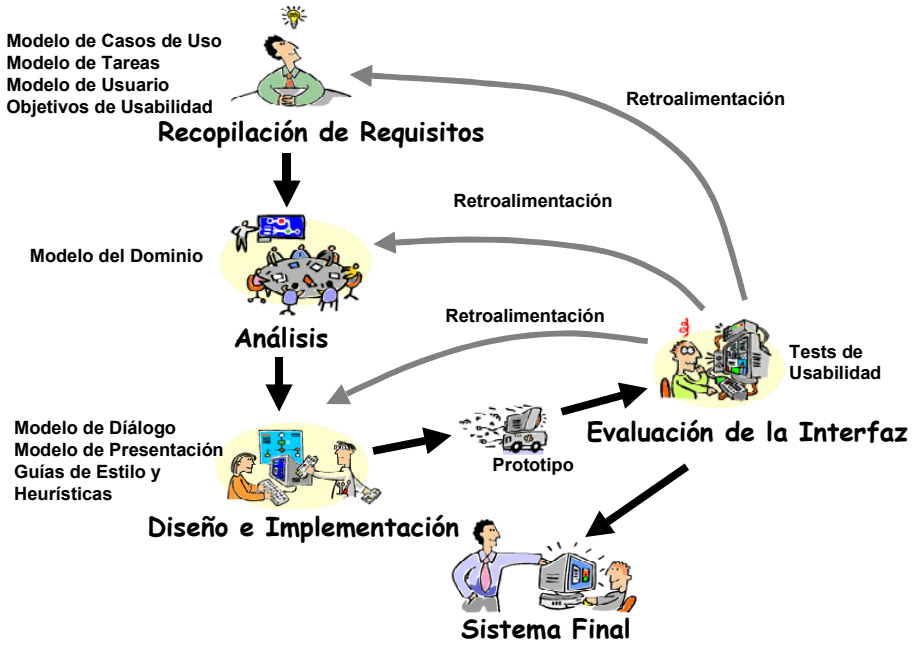

Figura 5. Proceso de Desarrollo Iterativo Incorporando Criterios y Test de Usabilidad.

Por ello, nuestra propuesta incluye modelos de alto nivel que permitan "conocer a los usuarios". En este punto, IDEAS propone la utilización de modelos de casos de uso, modelos de tareas y modelos de usuario. Los primeros permiten describir de manera precisa las tareas que realiza cada tipo de usuario, y el último nos ayuda a especificar las características individuales de los diferentes usuarios.

Tras conocer a los usuarios y las tareas que realizan, se propone un estudio que permita establecer las metas o criterios básicos de usabilidad que el sistema debe alcanzar. Esta definición permitirá más adelante realizar test de usabilidad asociadas a dichos criterios.

En las últimas etapas del desarrollo, a la hora de diseñar el modelo de presentación, el diseñador debe tener presente diferentes guías de estilo y heurísticas que le permitan mejorar la usabilidad de sus diseños. A su vez, a partir de dichos modelos de presentación se generan los prototipos de las interfaces. Estos prototipos son evaluados (test de usabilidad) por los usuarios finales de la aplicación.

Finalmente, todo lo anterior se integra dentro de un proceso de diseño iterativo. Este permite realizar el proceso de retroalimentación introduciendo en el desarrollo de la interfaz la información recogida en las pruebas realizadas por los usuarios, mejorando notablemente la usabilidad de las interfaces diseñadas tal y como se refleja en la figura 5.

7. Conclusiones.

En este trabajo se ha tratado el tema de la generación de interfaces de usuario dentro del proceso de desarrollo de software orientado a objetos. Este desarrollo se lleva a cabo aplicando técnicas de casos de uso y análisis de tareas, que nos llevan a la construcción de los diferentes modelos que constituyen el modelo de interfaz de usuario, y que representan cada uno de ellos los diferentes aspectos involucrados en la construcción de la interfaz.

Junto a lo anterior se ha descrito el modo en que se incorporan los criterios y tareas encaminadas a conseguir interfaces de alta usabilidad. Se han descrito qué modelos aportan información relevante dentro del desarrollo de sistemas usables y se han introducido ciertas tareas encaminadas a mejorar dicha usabilidad. Finalmente se ha planteado un proceso de diseño iterativo que permite incorporar las sucesivas mejoras que los usuarios introducen en el proceso de diseño.

Referencias

[Bal96] H. Balzert et al.: The JANUS application Development Environment Generating More than the User Interface. In: Computer-Aided Design of User Interfaces. Namur: Namur University Press, 1996, 183-205.

[Bod95] F. Bodart et al.: Towards a Systematic Building of Software Architectures: teh TRIDENT Methodological Guide. In: Design, Specification and Verification of Interactive Systems. Wien: Springer, 1995, 262-278.

[Bod96]F. Bodart et al.: Key Activities for a Development Methodology of Interactive Applications. In : Critical Issues in User Interface Systems Engineering. London: Springer, 1996, 109-134.

[Coc97]A. Cockburn: Structuring Use Cases with Goals. Journal of Object-Oriented Programming, 10(5) y 10(7), 1997. http://members.aol.com/acockburn/papers/usecases.htm

[Elw95] T. Elwert, E. Schlungbaum: Modelling and Generation of Graphical User Interfaces in the TADEUS Approach. In: Designing, Specification and Verification of Interactive Systems. Wien: Springer, 1995, 193-208.

[Fol95] J. Foley, P. Sukaviriya: History, Results and Bibliography of the User Interface Design Environment (UIDE), an Early Model-based System for User Interface Design and Implementation. In F. Paterno (ed.): Interactive Systems: Design, Specification and Verification. Berlín: Springer, 1995, 3-14.

[Gou85] J.D. Gould, C. Lewis. Designing for usability: key principles and what designers think. Communications of the ACM, (1985)28, 300-311.

[Har84] D. Harel: Dynamic Logic, In Handbook of Philosophical Logic II, editors D.M. Gabbay, F; Guenter ; pags, 497-694. Reidel 1984.

[ISO98]Human-Centered Design Processes for Interactive Systems. Draft of the ISO-DIS 13407 International Standard. 1998.

[Jac92] I.Jacobson, M. Christerson, et al.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, 1992.

[Jan93]C. Janssen: Generating User Interfaces from Data Models and Dialog Net Specifications. In: Bridges between Worlds. Proceedings InterCHI'93 (Amsterdam, April 1993). New York: ACM Press, 1993, 418-423.

[Joh95] P. Johnson, H. Johnson, S. Wilson: Rapid Prototyping of User Interfaces Driven by Task Models. In: J. Carroll (ed.) Scenario-Based Design.London: John Wiley & Son, 1995, 209-246.

[Lan95] Landauer, T.K. “The Trouble with Computers: Usefulness, Usability and Productivity. MIT Press, 1995.

[Let98] P. Letelier, I. Ramos, P. Sanchez, O. Pastor. OASIS versión 3.0: A Formal Approach for Object Oriented Conceptual Modelling. SPUPV-98.4011. Edited by the Technical University of Valencia, Spain, 1998.

[Lon96] F. Lonczewski: The FUSE System: An Integrated User Interface Design Environment. In: Computer-Aided Design of User Interfaces. Namur: Namur University Press, 1996, 37-56.

[Loz00] M.Lozano, I. Ramos, P. González. User Interface Specification and Modeling in an Object Oriented Environment for Automatic Software Development. 34th International Conference on Technology of Objetct-Oriented Languages and Systems. TOOLS-USA 2000. Santa Barbara, CA. 30 Julio - 3 Agosto, 2000

[Loz99a] M.Lozano, I. Ramos: An Object Oriented Approach for the Specification of User Interfaces. PACRIM'99, Victoria -Canadá- Agosto, 1999.

[Loz99b] M. Lozano, I. Ramos: Integration of the User Model and Human Factors in an Object Oriented Software Production Environment. ECOOP'99, Lisboa -Portugal-, Junio, 1999.

[Mär96] C. Märtin: Software Life Cycle Automation for Interactive Applications: The AME Design Environment. In: Computer-Aided Design of User Interfaces. Namur: Namur University Press, 1996, 57-74.

[Nie93] J. Nielsen. Usability Engineering. Morgan Kaufmann, 1993.

[Pas92] O. Pastor. Diseño y Desarrollo de un Entorno de Producción Automática de Software Basado en el Modelo Orientado a Objetos. Tesis Doctoral. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. 1992.

[Pue94a] A. Puerta et al.: Beyond Data Models for Automated User Interface Generation. In: People and Computers IX. Proceedings British HCI'94 (Glasgow UK, Agosto 1994). Cambridge: Cambridge University Press, 1994, 353-366.

[Pue94b] A. Puerta, P. Szekely: Model-based Interface Development. CHI'94 Tutorial Notes, 1994.

[Pue96] A. Puerta: The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development. In: Computer-Aided Design of User Interfaces. Namur: Namur University Press, 1996, 19-36.

[Ram93] Ramos, I. et. al., Objects as observable processes, in Proc. of the 4th International Workshop on the Deductive Approach to Information Systems and Databases, Tech. Report, DLSI, U. P. Catalunya, 1993.

[Shn98] B. Shneiderman. Designing the User Interface. Strategies for Effective Human-Computer Interaction. Addison Wesley, 1998.

[Sze93] P. Szekely et al: Beyond Interface Builders: Model-Based Interface Tools. In: Bridges between Worlds. Proceedings InterCHI'93 (Amsterdam, April 1993). New York: ACM Press, 1993, 383-390.

[Sze96] P. Szekely, et al.: Declarative Interface Models for User Interface Construction Tools: The MASTERMIND Approach. In: Engineering for Human-Computer Interaction. Proceedings of the IFIP Conference on Engineering for Human-Computer Interaction (Yellowstone Park, Agosto 1995). London: Chapman &Hall, 1996, 120-150

[Wie90] C. Wiecha et al.: ITS: A Tool for Rapidly Developing Interactive Applications. ACM Transactions on Information Systems 8 (1990), 3, 204-236.