Evaluación de la Usabilidad en un Entorno de Arquitecturas Orientadas a Modelos[1]

Sergio España, Inés Pederiva, José Ignacio Panach, Silvia Abrahão, Óscar Pastor

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, 46071 Valencia, España
{sergio.espana,ipederiva,jpanach,sabrahao,opastor}@dsic.upv.es
+34 96 387 7000

Resumen. Con el creciente interés en el paradigma de las arquitecturas orientadas a modelos (Model-Driven Architecture), los modelos conceptuales son la columna vertebral del proceso de desarrollo de software. En este trabajo se presenta un marco de evaluación de usabilidad en el contexto de la tecnología OlivaNova Model Execution, la implementación industrial del método de producción automática de software OO-Method. Presentamos un modelo de usabilidad que es la base para el marco de evaluación y una clasificación de problemas de usabilidad obtenida como resultado del proceso de evaluación. En consecuencia, la usabilidad de un sistema software es evaluada y mejorada en la fase de modelado conceptual basándose en las correspondencias entre la definición abstracta de la interfaz de usuario y sus implementaciones software.

1 Introducción

Con el desarrollo de aplicaciones generadas automáticamente a partir de modelos conceptuales se presenta un desafío poco explorado: la evaluación temprana de la usabilidad en productos software. Determinar cualidades de usabilidad en la etapa de análisis garantiza la calidad del producto y se adelanta a resolver problemas de interacción del usuario. La propuesta *Model Driven Architecture (MDA)* [12] es propicia para este tipo de desarrollo de aplicaciones ya que promueve el uso de modelos durante el proceso de desarrollo y permite que éstos puedan ser transformados hasta la obtención del código fuente del software final.

A pesar del crecimiento de la propuesta MDA, poco se ha considerado respecto a la evaluación temprana de usabilidad; no hay propuestas sistemáticas que descompongan el concepto de usabilidad en atributos medibles y determinen la relación de estos atributos con los patrones de interfaz. Así mismo, no se encuentran clasificaciones de las relaciones entre componentes abstractos de un modelo de interfaz, con componentes de la aplicación final.

Usando la tecnología OlivaNova™ Model Execution [4](ONME), en este trabajo mostraremos que las interfaces de usuario generadas automáticamente a partir del mé-

[1] Este trabajo está subvencionado por los proyectos DESTINO con ref. TIN2004-03534 y MAUSE *(Towards the Maturation of IT Usability Evaluation)* financiado por la COST European Action n°294.

todo de producción automático de software OO-Method [20] satisfacen un cierto nivel de usabilidad. Este nivel es alcanzado por construcción, a diferencia de la usabilidad de interfaces de usuario obtenida por métodos artesanales tradicionalmente usados.

Para realizar la evaluación de usabilidad definimos un modelo de usabilidad en base a la ISO/IEC 9126-1 [11]. Para lograr esto, sostenemos que un modelo de usabilidad debe incluir atributos de rendimiento y atributos de percepción. Ejemplo de atributos de rendimiento son la operabilidad y facilidad en el aprendizaje, mientras que los atributos de percepción, como el grado de atracción, son más subjetivos. Además, el modelo de usabilidad debería incluir: (1) el tipo de atributo de usabilidad: mediante una clasificación explícita; (2) la ponderación de los atributos de usabilidad en función del dominio de la aplicación; (3) la relación entre atributos: para detectar cómo un atributo incide positiva o negativamente en otro atributo; y (4) usabilidad vs. tipos de usuarios: para ponderar atributos según los perfiles de usuarios.

Este artículo está organizado como sigue. La sección 2 presenta los patrones conceptuales que provee el modelo de presentación ONME a partir de los cuales genera el código de las interfaces para las distintas plataformas. La sección 3 presenta el marco de usabilidad general, el modelo de usabilidad propuesto y su relación con la arquitectura MDA [12]. La sección 4 muestra la aplicación del modelo propuesto sobre un caso práctico. En la sección 5 se compara el modelo propuesto con los trabajos existentes. Finalmente, la sección 6 presenta las conclusiones y trabajos futuros.

2 Desarrollo de Interfaces de Usuario con ONME

ONME es la implementación realizada por CARE Technologies [4] del método de producción automático de software OO-Method [20]. OO-Method se basa en la separación del espacio del problema y del espacio de la solución siguiendo las directrices de MDA [12][14]. OO-Method comprende básicamente de los siguientes modelos:

El *Modelo Conceptual* que está dividido en cuatro vistas complementarias: (1) Modelo de Objetos que describe las propiedades estáticas del sistema en términos de clases y sus relaciones; (2) Modelo Dinámico que describe los aspectos relacionados con el control, las vidas válidas y la interacción entre objetos; (3) Modelo Funcional que describe la semántica asociada a los cambios de estado de los objetos provocados por la ejecución de eventos; y finalmente, el (4) Modelo de Presentación, que permite la especificación abstracta de los requisitos de interfaz de usuario.

Mediante una *Compilación de Modelos* se establecen las reglas de transformación de un Modelo Conceptual a su representación software correspondiente en una plataforma tecnológica concreta. Nos centraremos en el Modelo de Presentación, a partir del cual se genera la interfaz de usuario que determina la usabilidad de una aplicación.

2.1 El Modelo de Presentación

El Modelo de Presentación [15] permite la especificación abstracta de las interfaces de usuario por medio de un lenguaje de patrones conceptuales denominado Just-UI [16]. El modelo se estructura en tres niveles o capas de especificación.

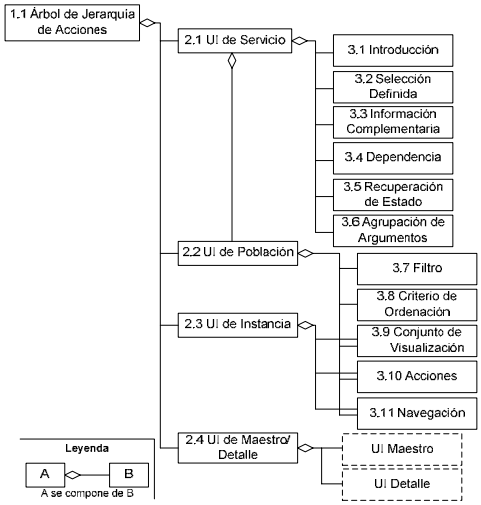


Fig. 1. Patrones de presentación en el espacio del problema

- **Nivel 1. Árbol de jerarquía de acciones (AJA):** Expresa cómo la funcionalidad será presentada al usuario.
- **Nivel 2. Unidades de interacción (UIs):** Modela abstracciones de interfaz con la-que el usuario interactuará. Existen cuatro unidades: (1)*UI de servicio:* modela la presentación de un diálogo para que un usuario lance un servicio;(2)*UI de instancia:* modela la presentación de los datos o estado de una instancia. Se define sobre una clase y se le proporciona un conjunto de visualización (qué información ha de ser mostrada), otro de acciones (qué funcionalidad debe ofertarse sobre el objeto) y por último un conjunto de navegación (qué información relacionada puede ser accedida); (3)*UI de población:* permite mostrar un conjunto de instancias de una clase. Se pueden establecer mecanismos de filtrado y ordenación para facilitar la selección y consulta de objetos; (4)*UI de maestro/detalle:* modela unidades de interacción construidas a partir de UIs más sencillas.
- **Nivel 3. Patrones elementales (PEs):** Permiten restringir y precisar el comportamiento de las diferentes unidades de interacción. La *selección definida* permite al analista proporcionar por enumeración los elementos válidos para un argumento de un servicio y el *patrón de información complementaria* permite especificar una información adicional que será mostrada al usuario.

3 Un marco de Usabilidad adecuado a MDA

La Figura 2 presenta las correspondencias entre los modelos propuestos por el paradigma MDA y los modelos que se manejan en un proceso de desarrollo con la tecnología ONME. El análisis del sistema produce el Modelo Conceptual, incluyendo el Modelo de Presentación que contiene la definición abstracta del interfaz del sistema. El Modelo Conceptual es asimilable al PIM (*Platform-Independent Model*) de MDA puesto que describe todos los aspectos funcionales de la aplicación, independiente-

mente de la plataforma software en la que vaya a implementarse y ejecutarse [14]. En una segunda fase, la compilación de modelos convierte el Modelo Conceptual en un Modelo de Aplicación que se corresponde con el PSM (*Platform-Specific Model*) de MDA. Automáticamente se vuelve a transformar el Modelo de Aplicación en el Código Fuente de la aplicación final, correspondiente al *Code Model* de MDA, en un paso que resulta transparente al analista desarrollador.

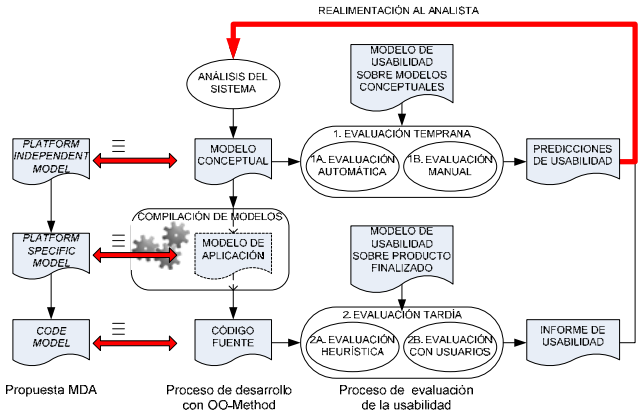


Fig. 2. Correspondencia entre MDA y OO-Method, y evaluación temprana de la usabilidad.

Habitualmente, la evaluación de la usabilidad se viene realizando sobre la aplicación ya desarrollada (sobre el *Code Model*), mediante técnicas heurísticas, testings con usuarios, etc. Esta estrategia supone una realimentación completamente fiable para el analista y el diseñador y es la más frecuentemente usada.

La aportación de este trabajo consiste en adelantar la evaluación a la fase de análisis (modelado conceptual), de manera que, como información de entrada, se toma el PIM y se obtiene como salida un informe de predicciones de usabilidad que realimentará el análisis del sistema, en particular el Modelo de Presentación. Así se asegura la *usabilidad independiente de plataforma*, como un aspecto de la calidad interna, que puede ser evaluada automática (1A) o manualmente (1B). Por otro lado el modelo propuesto también soporta la evaluación de la calidad externa (usado como pautas o *guidelines*, 2A). Estas técnicas complementan la habitual evaluación de la calidad en uso (frecuente en la Interacción Persona-Ordenador, 2B).

3.1 Usabilidad independiente de plataforma

El objetivo del marco propuesto es evaluar la usabilidad sobre un modelo independiente de plataforma y proporcionar al analista una realimentación que sugiera cambios sobre el Modelo de Presentación, con el fin de prevenir la aparición de errores y problemas de usabilidad en la aplicación que se generará. Este es un proceso iterativo que se repite hasta que el Modelo Conceptual tenga el nivel de usabilidad requerido.

El primer paso en la definición del marco fue definir un modelo de usabilidad basado en el estándar ISO/IEC 9126-1[11]. Se fue aumentando la granularidad de las subcaracterísticas de la usabilidad en una descomposición jerárquica, hasta llegar a

definir atributos medibles. Para obtener estos atributos se analizaron varias taxonomías de criterios de usabilidad propuestas en la literatura [8][9][21], entre las que destaca la de Bastien y Scapin [3] y sus criterios ergonómicos validados empíricamente.

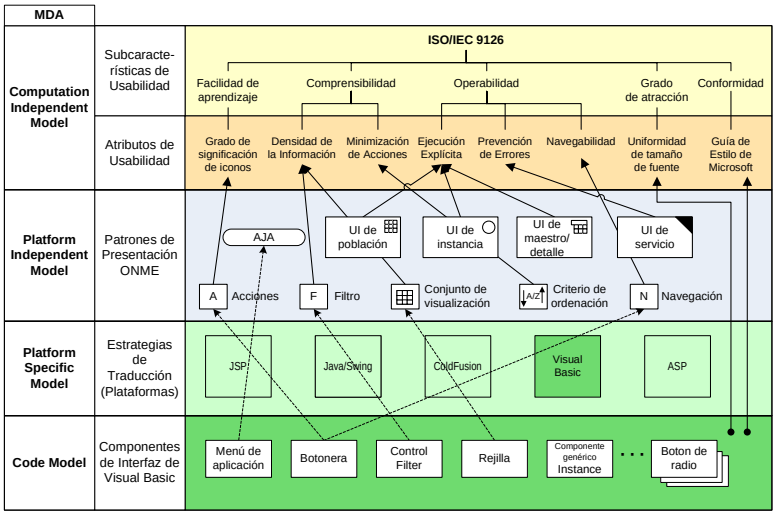


Fig. 3. Modelo de Usabilidad propuesto.

El modelo fue presentado en [19][1], y ha sido revisado y actualizado hasta convertirse en el árbol que presentamos en este trabajo. Los atributos de usabilidad, situados en las hojas del árbol se relacionan con los patrones de presentación, las abstracciones de la interfaz pertenecientes al PIM. En la Figura 3 se ilustra algunas de estas relaciones. La finalidad es determinar qué patrones de presentación contribuyen a la satisfacción de determinados atributos de usabilidad provenientes de la ergonomía de interfaces de usuario. Cabe destacar que las subcaracterísticas grado de atracción y conformidad no pueden ser evaluadas de manera objetiva en la fase de modelado conceptual y, por lo tanto, se definen relaciones con componentes del *Code Model*; solo podrán ser evaluadas tras la generación de la aplicación final.

3.2 Usabilidad dependiente de plataforma

ONME oculta el PSM al desarrollador al definir una estrategia de compilación que directamente obtiene el código fuente de la aplicación final a partir del Modelo Conceptual. En este espacio intermedio se encuentran las distintas estrategias de generación de código existentes que dan soporte a la generación automática de aplicaciones de tres capas (interfaz, lógica de negocio y base de datos) para diversas plataformas (Visual Basic, Java/Swing, ColdFusion, JSP, ASP y PocketPC). En función del generador de código seleccionado, los patrones de presentación abstractos son instanciados en la arquitectura software correspondiente, en un proceso denominado *reificación*.

Existen subcaracterísticas de la usabilidad que deben ser evaluadas sobre la concreción de los patrones de presentación para una plataforma determinada; por lo tanto

se establecen relaciones entre estas subcaracterísticas y los componentes software que conforman la interfaz gráfica resultante.

Es el caso de la subcaracterística grado de atracción, en la que hemos diferenciado aspectos sintácticos (p.e. uniformidad del color y el estilo de la fuente) y semánticos (p.e. atracción subjetiva). En este caso, los atributos sintácticos están garantizados por la estrategia de traducción que implementa la tecnología ONME. La evaluación de los atributos semánticos escapa a la evaluación temprana que aquí presentamos, puesto que debe realizarse con usuarios.

3.3 Modelo de Usabilidad

El árbol de atributos de nuestro modelo (ver Tabla 1) constituye una taxonomía de los aspectos que componen la usabilidad de una aplicación. Para desarrollar el modelo de usabilidad se realizaron las siguientes actividades: definición de los objetivos de calidad perseguidos, especificación de las características y atributos de usabilidad, especificación de relaciones y confección del modelo dotándolo de potenciales métricas.

Tabla 1. Árbol de aributos de usabilidad

1. **Facilidad de aprendizaje**
   - 1.1. Facilidades de Ayuda
     - 1.1.1. Completitud de Documentación
     - 1.1.2. Documentación Multiusuario
       - 1.1.2.1. Distinción por Capacitación
       - 1.1.2.2. Distinción por Rol
   - 1.2. Predecibilidad
     - 1.2.1. Grado de Significación de Iconos
     - 1.2.2. Título de Iconos o Enlaces
     - 1.2.3. Determinación de la Acción
   - 1.3. Realimentación Informativa
   - 1.4. Facilidad de Memorización
2. **Comprensibilidad**
   - 2.1. Legibilidad
     - 2.1.1. Tamaño de Fuente
     - 2.1.2. Contraste
     - 2.1.3. Disposición
   - 2.2. Facilidad de Lectura
     - 2.2.1. Agrupación Cohesiva de la Información
       - 2.2.1.1. Agrupación por Disposición
       - 2.2.1.2. Agrupación por Formato
     - 2.2.2. Densidad de la Información
   - 2.3. Familiaridad de conceptos
     - 2.3.1. Significación del Etiquetado
     - 2.3.2. Internacionalización
   - 2.4. Reducción de la Carga de Trabajo
     - 2.4.1. Brevedad
       - 2.4.1.1. Inicialización de Valores
         - 2.4.1.1.1. Completitud de Valores Iniciales
         - 2.4.1.1.2. Modificabilidad de Valores Iniciales
       - 2.4.1.2. Minimización de acciones
     - 2.4.2. Autodescripción
   - 2.3. Orientación al usuario
     - 2.3.1. Calidad de los Mensajes
     - 2.3.2. Navegabilidad
3. **Operabilidad**
   - 3.1. Capacidades de Instalación
     - 3.1.1. Facilidad de Instalación
     - 3.1.2. Multiplicidad de Instalación
     - 3.1.3. Capacidad de Actualización
     - 3.1.4. Transparencia de Actualización
   - 3.2. Validación de datos
   - 3.3. Capacidad de Control
     - 3.3.1. Aplazamiento de Edición
     - 3.3.2. Soporte de Cancelación
     - 3.3.3. Ejecución Explícita
     - 3.3.4. Soporte de Interrupción
     - 3.3.5. Soporte de Deshacimiento
     - 3.3.6. Soporte de Rehacimiento
   - 3.4. Capacidad de Adaptación
     - 3.4.1. Adaptabilidad
     - 3.4.2. Adaptatividad
   - 3.5. Consistencia
     - 3.5.1. Comportamiento Constante de Controles
     - 3.5.2. Permanencia de Controles
     - 3.5.3. Estabilidad de los Controles
     - 3.5.4. Consistencia de Orden
     - 3.5.5. Consistencia de Etiquetado
   - 3.6. Gestión de Errores
     - 3.6.1. Prevención de Errores
     - 3.6.2. Recuperación de Errores
   - 3.7. Monitorización del Estado del Sistema
4. **Grado de atracción**
   - 4.1. Metáfora
   - 4.2. Uniformidad del Color del Fondo
   - 4.3. Uniformidad del Color de la Fuente
   - 4.4. Uniformidad del Estilo de la Fuente
   - 4.5. Uniformidad del Tamaño de la Fuente
   - 4.6. Uniformidad en Disposición de Elementos
   - 4.7. Atracción Subjetiva
5. **Conformidad**
   - 5.1. Cumplimiento con ISO/IEC 9126 (partes 1 y 3)
   - 5.2. Cumplimiento con ISO 9241-10
   - 5.3. Cumplimiento con Guía de Estilo de Microsoft
   - 5.4. Cumplimiento con Guía de Estilo de Java

El objetivo de nuestro modelo de usabilidad es analizar el Modelo de Presentación con el propósito de evaluar su usabilidad desde la perspectiva del investigador y en el contexto de la iniciativa MDA instanciada sobre ONME. Considerando las subcaracterísticas como una forma de agrupar aspectos de la usabilidad con relación entre sí, estructuramos jerárquicamente los atributos de usabilidad basándonos en ISO/IIEC 9126-1 [11]. La descomposición es necesaria para poder aplicar métodos de inspección de la usabilidad y para identificar y arreglar los problemas de usabilidad detectados.

3.3.1 Facilidad de Aprendizaje

La facilidad de aprendizaje referencia a los atributos que dotan a un producto software de permitir al usuario aprender su manejo. Esta subcaracterística consiste en **facilidades de ayuda** (ayuda en línea y documentación, en sus aspectos de completitud y adaptación a distintos perfiles de usuario), así como la **predecibilidad**, que se refiere a la facilidad con que un usuario puede determinar el resultado de sus acciones futuras (p.e. grado de significación de los iconos y determinación de la acción), la **realimentación informativa** en respuesta a sus acciones y la **facilidad de memorización**.

3.3.2 Comprensibilidad

La comprensibilidad se refiere a atributos que facilitan entender si el software resulta adecuado y cómo puede usarse. Descomponemos esta subcaracterística para diferenciar la propia **legibilidad** óptica de los textos e imágenes de la **facilidad de lectura,** que a su vez consta de la **agrupación cohesiva de información** presentada y **densidad de la información**. La **familiaridad de conceptos** es un aspecto relativo al contexto social, cultural y laboral del usuario. La **brevedad** está principalmente relacionada con patrones que reducen el esfuerzo cognitivo del usuario (dotar de valores iniciales a los campos de los formularios, evitar interacciones innecesarias). Respecto a la **orientación al usuario,** pasa por asegurar la **calidad de los mensajes** y **navegabilidad**. La navegabilidad está relacionada con guiar al usuario proveyendo mecanismos para informar de la ruta y la posición actual. En nuestro modelo, la navegabilidad puede ser medida en términos de la ausencia de navegaciones cíclicas entre Unidades de Interacción abstractas (UIs).

3.3.3 Operabilidad

La operabilidad se refiere a los atributos que facilitan el control del usuario y la operación. Esta subcaracterística referencia a la capacidad del sistema para proporcionar **capacidades de instalación** (facilidad de instalación, transparencia de actualización, etc), realizar **validación de datos** y ofrecer **capacidad de control** del usuario en la ejecución de servicios. Sobre la **capacidad de adaptación** de la interfaz de usuario hacemos una diferenciación entre *adaptabilidad*, capacidad del software de ser adaptado por el usuario, y *adaptatividad*, capacidad del software de adaptarse a las necesidades del usuario (la diferencia está en el agente de la adaptación) [18]. La **consistencia** operacional en la ejecución de servicios y controles es otra subcaracterística de la operabilidad, al igual que la **gestión de errores** y la **capacidad de monitorizar el estado del sistema** por la cual el sistema comunica información al usuario sin ser el resultado de una interacción con éste.

3.3.4 Grado de Atracción

Algunos aspectos del grado de atracción relacionados con el diseño estético pueden ser cuantificados desde una perspectiva sintáctica midiendo la **uniformidad de la interfaz** en términos del tamaño y estilo de la fuente utilizada, del uso que se hace del color y de la disposición de los elementos en la pantalla. Estas uniformidades son garantizadas por las estrategias de generación de código de ONME. Otros aspectos son de carácter semántico y proponemos evaluarlos subjetivamente: un adecuado uso de la **metáfora** y el grado de **atracción subjetiva**, medible mediante cuestionarios.

3.3.5 Conformidad

C**onformidad** se mide evaluando la concordancia del PIM con los siguientes estándares: ISO/IEC 9126-1 y -3 e ISO 9241-10 y la concordancia del *Code Model* con las guía de estilo de Microsoft [13] y de Java [22]. Las dos primeras valoraciones se centran en el espacio del problema y las dos últimas en el espacio de la solución, pues analizan el cumplimiento de la guía de estilo de Microsoft y del *look and feel* de Java por parte de los componentes software empleados en las diferentes estrategias de generación automática de código.

Para cada atributo se ha estudiado cómo afecta positiva o negativamente a otros y se ha definido explícitamente las diferencias con otros atributos con los cuales parecía solaparse. Además, se han definido relaciones entre los atributos de usabilidad y los patrones de interfaz del PIM y sus propiedades, estableciendo correspondencias entre los niveles CIM y el PIM. La estrategia de traducción automática proyecta estas correspondencias hasta el CM (*Code Model*), logrando que el modelo de usabilidad tenga efecto en todas las capas del paradigma MDA.

Igualmente importante es definir métricas para cuantificar cada atributo. Dada la correspondencia entre los atributos de usabilidad y los componentes del PIM, las métricas se calcularán sobre este modelo con la salvedad de las métricas semánticas, cuya valoración se realizará contra la interfaz final.

La siguiente tabla muestra las relaciones establecidas entre algunos atributos de usabilidad y los patrones de interfaz del PIM. En la tercera columna se explica la razón fundamental de esta relación, mientras que en la última se proponen métricas.

Tabla 2. Subcaracterísticas, atributos, patrones de presentación y métricas.

Subcaract. / Atributo	**Elementos del PIM afectados**	**Razón fundamental**	**Métricas**
Facilidad de Aprendizaje / Completitud de Documentación	HAT, UI de Instancia, UI de Población, UI de Servicio	La existencia de una ayuda en línea facilita al usuario el aprendizaje de la herramienta; por lo tanto hay que comprobar que el analista ha definido mensajes de ayuda en los patrones de interfaz.	Proporción de acciones de una UI que tienen definidas mensajes de ayuda. Índice de completitud de mensajes de ayuda definidos para una UI. Índice de completitud de mensajes de ayuda para el modelo.
Comprensibilidad / Densidad de la Información	Acciones, Conjunto de visualización, UI de Servicio (argumentos), Filtro	La densidad de la información afecta a la facilidad de lectura de la interfaz, por lo tanto debe evitarse presentar al usuario información innecesaria para la tarea que está realizando.	Proporción de acciones invocables en una UI de Servicio. Proporción de UIs de Población en las que se definen filtros.
Comprensibili-	Introducción,	Los valores por defecto redu-	Proporción de patrones de Introduc-

Subcaract. / Atributo	Elementos del PIM afectados	Razón fundamental	Métricas
dad / Completitud de Valores Iniciales	Selección Definida	cen la carga de trabajo del usuario y facilitan la comprensión.	ción y Selección Definida que definen un valor inicial.
Operabilidad / Consistencia de Etiquetado	UI de Servicio, UI de Instancia, UI de Población, Navegación	Las etiquetas relativas a argumentos, campos y navegaciones deben ser consistentes a lo largo de las UIs para facilitar su reconocimiento.	Proporción de campos relativos al mismo dato con etiquetas consistentes a lo largo de UIs relacionadas. Proporción de navegaciones iguales con etiquetas consistentes a lo largo de UIs relacionadas.
Operabilidad / Significación del Etiquetado	Alias en los patrones: Introducción, Información Complementaria, Selección Definida, Filtros, Navegación, Agrupación de Argumentos y Acciones	Una etiqueta es significativa para los usuarios cuando hay una fuerte relación semántica entre aquellas etiquetas y los ítems o acciones a los que se refiere.	Proporción de elementos con un alias definido explícitamente por el analista Proporción de elementos con nombres significativos para cada patrón (propiedad). Proporción total de elementos con nombres significativos.

En algunos casos es posible y conveniente definir métricas con distinta granularidad. Por ejemplo, para el atributo **completitud de documentación**, podríamos verificar la existencia de mensajes de ayuda definidos para todos y cada uno de los elementos de una UI de Población determinada: campos del conjunto de visualización, acciones ofertadas, filtros, navegaciones, etc. Por ejemplo <proporción de acciones de una UI que tienen definidas mensajes de ayuda>. Ahora bien, a la hora de computar una métrica genérica para la UI (un índice de completitud), podemos realizar una media ponderada de las métricas anteriores, pues no es igual de relevante el mensaje de ayuda en los campos de un filtro como lo es en las acciones ofertadas al usuario. Los índices particulares de cada UI se pueden combinar en un único índice de completitud de documentación relativo al modelo PIM valorado.

Se puede observar que el atributo **significación del etiquetado** está relacionado con prácticamente todos los patrones, pues la propiedad de *alias* (la etiqueta que se presenta al usuario) se puede definir para todas las abstracciones de la interfaz. Por ejemplo, en la UI de Servicio cada argumento tiene una etiqueta, lo mismo ocurre con los campos de las UIs de Instancia y de Población. Respecto a las métricas, podemos diferenciar las de carácter sintáctico, verificables sobre el PIM en una evaluación temprana, y las de carácter semántico, que se calcularán sobre el *Code Model*. Entre las métricas sintácticas se propone obtener la <proporción de elementos con un alias definido explícitamente por el analista>, lo cual es más relativo a la completitud porque no asegura la significación de la etiqueta escogida por el analista y su evaluación puede ser automatizada.

Entre las métricas semánticas hay algunas que, si bien requieren intervención humana, pueden todavía evaluarse sobre el PIM porque se corresponden con propiedades de patrones de interfaz: <proporción de elementos con nombres significativos para cada patrón>. Por último, hay otras métricas que implican necesariamente una evaluación con usuario, sobre el *Code Model*. Es el caso de <proporción de elementos cuyo nombre considera significativo el usuario>. El leitmotiv del marco propuesto es precisamente que la evaluación temprana sobre el PIM minimice la aparición de problemas de usabilidad en las evaluaciones tardías sobre el CM.

4 Caso de estudio

Hemos tomado un caso práctico de gestión de campings generado con OlivaNova para ilustrar el uso del modelo de usabilidad propuesto para la evaluación de la calidad externa (ver Figura 2, 2A). El negocio principal del camping es el alquiler de *Bungalows* o *Parcelas*. Los usuarios pueden también utilizar las instalaciones deportivas (*ServiciosUsados*) sin recargo o por una cierta cantidad de dinero. También pueden concertar *ServiciosExtra* como excursiones, etc. Al final de la *Estancia* del cliente se emite una *Factura*. Además, las principales funciones que el sistema debe proporcionar son: control de la ocupación de las parcelas y bungalows, seguimiento del cliente y facturación. A continuación se muestra el Modelo de Objetos del caso de estudio, a partir del cual se especifica el Modelo de Presentación.

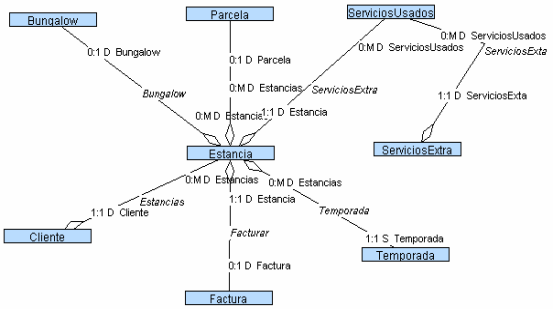
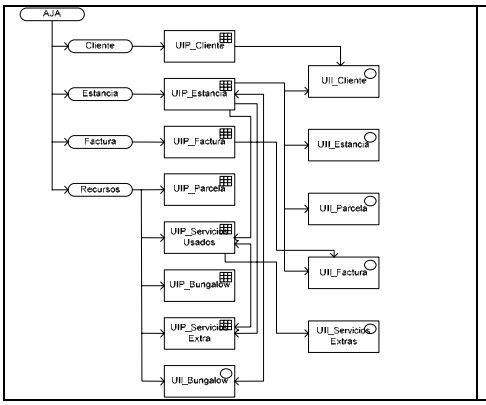


Fig. 4. Modelo de Objetos



UI de servicio:
Nombre:UIS_Crear
Argumentos: Estancia, Total, ModoPago, NumFactura, FechaPago, Pagada, FechaEmision
UI de instancia:
Nombre: UII_Temporada
Conjunto de visualización: Descripción, FechaFin, FechaInicio, Id, Nombre, Recargo
UI de población:
Nombre: UIP_Factura
Conjunto de visualización: FechaEmision, FechaPago, ModoPago, Nímero, Pagada, Total
Acciones: Pagar, Crear, Cambiar, Destruir
Navegación: Estancia, Cliente, Factura
Criterio de ordenación: <Número, Asc>
Filtro: Estancia.Cliente = VCliente

Fig. 5. Modelo de Presentación

La Figura 5 muestra en la primera columna el árbol de jerarquía de acciones para la aplicación estudiada. En la segunda columna se muestra una descripción detallada de cada uno de los patrones con los que se generan las interfaces que utilizaremos para detectar problemas de usabilidad.

Para registrar cada uno de los problemas de usabilidad, nos hemos basado en el método de clasificación CUP [23] (*Classification of Usability Problems*). En cual-

quier caso, hemos extendido esta plantilla para reflejar a qué **patrón del Modelo de Presentación** afecta cada uno de los problemas detectados y qué **atributo de usabilidad** se ve implicado. A continuación, presentamos dos problemas de usabilidad a modo de ejemplo.

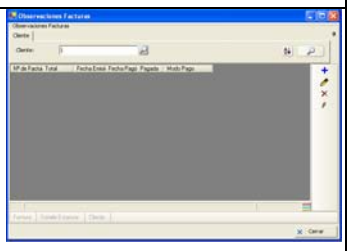
Identificador: 1

Descripción: la aplicación no facilita la entrada del dato *Modo Pago*. Debería aparecer una lista desplegable para que los valores introducidos sean homogéneos, ya que el conjunto de valores posibles es pequeño. Además, podría aparecer por defecto la forma de pago más utilizada comúnmente.

Actividad en la que se encontró el defecto: evaluación usabilidad
Detonante: al crear una *Factura*
Impacto: moderado
Fase en la que se debe reparar: diseño de la presentación
Clasificación del error: mejoras
Causa: Personal
Atributo de usabilidad afectado: minimización de acciones y valores por defecto
Patrón de presentación afectado: ausencia de *Selección Definida*, dentro de la *UI de Servicio*
Prevención del error: identificar aquellos argumentos que tengan pocos valores posibles
Solución: introducir una lista desplegable mediante el patrón de *Selección Definida* y establecer como valor por defecto el modo de pago más utilizado.

Identificador: 2

Descripción: al listar las facturas solo aparece un filtro por cliente, pero no existe un filtro para ver por fecha de pago o ver solo los que estén sin pagar. Esta ventana va a presentar mucha información y hay que facilitarle la búsqueda al usuario.



Actividad en la que se encontró el defecto: evaluación usabilidad
Detonante: al listar todas las instancias de la clase *Factura*
Impacto: moderado
Fase en la que se debe reparar: diseño de la presentación
Clasificación del error: mejoras
Causa: Personal
Atributo de usabilidad afectado: densidad de la información
Patrón de presentación afectado: afecta a *Filtro* dentro de *Unidad de Interacción de Población*
Prevención del error: utilizar distintos filtros para una interfaz que muestre gran cantidad de información
Solución: introducir un filtro por *fecha de pago* y otro por el atributo *pagado*.

5 Comparación con trabajos relacionados

En los últimos años varios modelos de calidad han sido propuestos en la literatura. Estos modelos fueron definidos basándose en estándares existentes como la ISO/IEC 9241-11 [10] y la ISO/IEC 9126-1 [11] o partiendo desde cero. La ISO/IEC 9126-1 es un modelo de calidad genérico de propósito general. La ISO 9241-11 es un estándar que define la usabilidad como la medida en la que un producto pueden usarlo determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado. Aunque estos modelos son útiles, deben ser adaptados para su utilización.

Dromey et al. [6], Dix et al. [5], y Nielsen [17] proponen modelos propios para descomponer la usabilidad. La propuesta de Dromey se basa en definir propiedades de

comportamientos y *usos*. El *comportamiento* es algo que el software presenta cuando se ejecuta bajo la influencia de un conjunto de entradas. El *uso* es algo que diferentes usuarios hacen con o para el software. Las propiedades que plantea Dromey, son un subconjunto de los atributos que incluimos en nuestro modelo. Por ejemplo: la propiedad *transparencia*, que indica la facilidad de comprensión y de recordar la funcionalidad, se podría incluir en nuestra subcaracterística *comprensibilidad y facilidad de memorización*.

Dix et al. [5] proponen un modelo de usabilidad centrado en tres categorías principales: *facilidad de aprendizaje, flexibilidad* y *robustez.* Cada una de estas categorías se subdivide en principios más específicos que la soportan. Este modelo es similar a nuestra propuesta. Nosotros partimos de subcaracterísticas de usabilidad para llegar a atributos medibles y Dix parte de categorías parar llega hasta principios específicos. Los principios propuestos por Dix están recogidos en uno o más de nuestros atributos. Por ejemplo, la *capacidad de personalizar* que propone Dix, está reflejada en los atributos *adaptatividad* y *adaptabilidad*.

Nielsen [17] modela la usabilidad como parte de un modelo más amplio que parte de la *aceptación del sistema*. Dentro de la *aceptación del sistema* está la *aceptación práctica* que incluye la *utilidad* y, dentro de esta última, la usabilidad. Nielsen descompone la usabilidad en los siguientes atributos: *facilidad de aprendizaje, eficiencia, capacidad de memorizar, errores* y *satisfacción*. Hay ciertos atributos existentes en nuestro modelo que no aparecen modelados en la propuesta de Nielsen, como por ejemplo la *compresibilidad* y la *operabilidad.* También plantea algunos componentes, como por ejemplo la *satisfacción*, que engloban a atributos de nuestro modelo, el *grado de atracción* en este caso.

Por otro lado, otros trabajos se basan en la extensión o composición de estándares existentes. Este es el caso de las propuestas de Van Welie et al.[24], Fitzpatrick et al. [7], Abran et al.[2], y Shneiderman [21].

Van Welie et al. [24]. proponen un modelo con tres capas y relaciones entre ellas. La capa superior es la de *Usabilidad* (ISO/IEC 9241) y está dividida en tres aspectos: *eficiencia, efectividad* y *satisfacción*. La siguiente capa contiene un número de *indicadores de uso* que son índices del nivel de usabilidad que se puede observar en la práctica, cuando los usuarios están trabajando. Cada uno de estos *indicadores* contribuye a ciertos aspectos abstractos de la capa superior. La capa inferior se conoce como *significados*, usados como heurísticas para mejorar los indicadores de uso. Van Welie se centra más en aspectos medibles a través de pruebas de usuario dejando de lado aquellos aspectos que proponemos que sean medidos en la fase de modelado conceptual, como por ejemplo la *operabilidad*.

Fitzpatrick et al. [7] proponen tres ramas a examinar para identificar los atributos de un producto software usable. La primera rama es la de *calidad del software,* es decir, el grado en el que el software posee una combinación de atributos deseada. La segunda rama son las *obligaciones legales*. Son las obligaciones relacionadas con los mínimos requisitos de salud y seguridad que debe tener un equipo. La última rama es la *interacción persona ordenador*, que se describe como el estudio de las personas, la tecnología de los ordenadores y la forma en que unos influyen en los otros. Fitzpatrick considera como factores de calidad solo aquellos que dan soporte al usuario final. Distingue entre factores de calidad externos e internos. En los internos incluye ciertos atributos como la *facilidad en el mantenimiento, la reusabilidad y la portabilidad.*

Abran et al. [2] propusieron un modelo apoyándose en la ISO 9126 y sobre todo en la 9241-11, tomándola como base. Además, incluye las subcaracterísticas *facilidad de aprendizaje* y *seguridad*. El modelo propuesto por Abran es similar a nuestra propuesta. Una de las subcaracterísticas contempladas en el modelo propuesto y que no aparece en el de Abran, es la *operabilidad*, al igual que en la propuesta de Van Welie.

Shneiderman [21] propone que objetivos medibles y precisos guíen al diseñador, al evaluador, al comprador y al administrador. Esto se debe hacer para cada usuario y cada tarea. Shneiderman propone cinco factores humanos que son centrales para la evaluación de usabilidad: *tiempo de aprendizaje, rendimiento, tasa de errores por usuario, mantenimiento del conocimiento adquirido de la aplicación* y *satisfacción subjetiva.* En nuestro modelo aparecen ciertas subcaracterísticas de la usabilidad, como por ejemplo la *comprensibilidad* y la *operabilidad*, que no aparecen reflejados en los objetivos de diseño de la interfaz de usuario que propone Shneiderman.

6 Conclusiones

En este trabajo se ha presentado una estrategia de evaluación de usabilidad en la fase de modelado conceptual de un método de producción automática de software que sigue los principios de MDA. Esta estrategia de evaluación ha sido llevada a cabo a través de la definición de un modelo que descompone el concepto de usabilidad en atributos medibles y que clarifica las relaciones existentes entre estos atributos y los elementos del modelo independiente y dependiente de plataforma. Como consecuencia, la usabilidad de un sistema software puede ser evaluada y mejorada en el espacio del problema. Al mejorar la calidad interna, parece sensato suponer que se reducirá el número de defectos de usabilidad que aparecerán durante las evaluaciones de calidad externa y calidad en uso. Pretendemos demostrar esto empíricamente en trabajos futuros.

El modelo de usabilidad propuesto ha sido aplicado en la evaluación de varios casos de estudio reales desarrollados con la tecnología ONME, aunque este modelo puede ser usado para evaluar cualquier aplicación desarrollada siguiendo las directrices de MDA.

Como trabajo futuro se plantea definir métricas para el Modelo de Presentación que nos permitan automatizar la evaluación de la usabilidad. También tenemos previsto realizar pruebas con usuarios para comprobar empíricamente la mejora de la usabilidad a nivel conceptual y medir las subcaracterísticas subjetivas.

Referencias

[1] Abrahão, S., Ó. Pastor and L. Olsina (2005). A Quality Model for Early Usability Evaluation. INTERACT05 International COST 294 Workshop on User Interface Quality Models (UIQM), Rome, Italy.

[2] Abran A., Khelifi A., Suryn W., Seffah A. (2003) Usability Meanings and Interpretations in ISO Standards, Software Quality Journal, 11 (4) pp. 325-338.

[3] Bastien, J. M. C., & Scapin, D. L. (1993). Ergonomic criteria for the evaluation of human-computer interfaces (Report No. 156). Rocquencourt, France: Institut National de Recherche en Informatique et en Automatique..
[4] Care Technologies: http://www.care-t.com Última visita: Nov-2005
[5] Dix, A., Abowd, G., Beale, R. and Finlay, J. (1998), Human-Computer Interaction, Prentice Hall Europe.
[6] Dromey R.G. (1998) Software Product Quality: theory, Model and Practice, Technical report, Software Quality Institute, Griffith University, Nathan, Brisbane, Australia.
[7] Fitzpatrick, R. and C. Higgins (1998). Usable Software and Its Attributes: A Synthesis of Software Quality, European Community Law and Human-Computer Interaction Proceedings of HCI on People and Computers XIII.Springer-Verlag: 3-21.
[8] Folmer E., Bosch J. (2004) Architecting for usability: A Survey, Journal of Systems and Software, 70(1) 61-78.
[9] Hix, Deborah and H. Rex Hartson (1993), Developing User Interfaces: Ensuring Usability Through Product & Process. New York, NY: John Wiley & Sons.
[10] ISO 9241-11 (1998) Ergonomic requirements for office work with visual display terminals - Part 11: Guidance on Usability.
[11] ISO/IEC 9126-1 (2001), Software engineering - Product quality - 1: Quality model.
[12] MDA: http://www.omg.org/mda Última visita: Nov-2005
[13] Microsoft, "Guía de Estilo para el diseño de aplicaciones .NET". 2004. http://www.microsoft.com/spanish/msdn/arquitectura/DesignGuidelines/DesignGuidelines.asp Última visita: Nov-2005
[14] Molina J.C., Pastor O. “MDA, OO-Method y la tecnología OlivaNova Model Execution”. I Taller sobre desarrollos dirigidos por modelos, MDA y aplicaciones. Málaga, 2004.
[15] Molina P., Especificación de interfaz de usuario: De los requisitos a la generación automática, PhD Thesis, DSIC, Universidad Politécnica de Valencia, Marzo de 2003.
[16] Molina, P. J. Meliá, S. and Pastor, O. (2002), Just-UI: A User Interface Specification Model. In Ch. Kolski and J. Vanderdonckt (Eds.), Computer-Aided Design of User Interfaces III, Kluwer Academics Publisher, 63–74.
[17] Nielsen, J. (1993). Usability Engineering. Morgan Kaufmann Publishers Inc.
[18] Oppermann R., Rossen R., Kinshuk, "Adaptability and Adaptivity in Learning Systems", Knowledge Transfer (Volume II). A. Behrooz (Ed), Pace, London, UK, 1997, pp. 173-179.
[19] Ortiz, J. C., Condori-Fernandez, N., Abrahão S., A (2005) Un Marco de Usabilidad para las Aplicaciones Generadas con la Tecnología OlivaNova Model Execution, VI Congreso Español de Interacción Persona Ordenador (Interacción 2005), Granada, España,.
[20] Pastor, O., Gómez, J., Insfrán, E. Pelechano, V. (2001) The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming. Information Systems, 26(7) 507–534.
[21] Shneiderman, B. (1998), Designing the User Interface: Strategies for Effective Human-Computer Interaction, Addison-Wesley, Reading, Massachusetts, USA.
[22] Sun Microsystems. "Java Look and Feel Design Guidelines", 2001. http://java.sun.com/products/jlf/ Última visita: Nov-2005
[23] Hvannberg, E. T. and L.-C. Law (2003). Classification of Usability Problems (CUP) Scheme. M. Rauterberg, M. Menozzi and J. Wesson, Human-Computer Interaction INTERACT 2003, Zurich, Switzerland, pp. 655-662.
[24] Welie, M. v., G. C. v. d. Veer and A. Eliëns (1999). Breaking Down Usability. NTERACT 99, Edinburgh. IOS Press, pp. 613-620