Soporte para el Análisis de la Arquitectura de la Información en Aplicaciones Web

Luis A. Rojas y José A. Macías

Resumen

En los equipos de desarrollo de aplicaciones web interactivas intervienen expertos no informáticos centrados en el modelado conceptual y la concreción de aspectos no funcionales de la aplicación. Este análisis de la información es luego procesado por analistas e ingenieros en las fases técnicas del proyecto. Muchas veces, este trasiego de información impide una automatización implícita, debido a la que la información procesada por los distintos profesionales tiene un nivel conceptual diferente, lo que supone un coste mayor, en tiempo y en esfuerzo, en el desarrollo del proyecto. El objetivo de este trabajo es minimizar dichos costes mediante una herramienta CASE capaz de unir puentes entre las definiciones conceptuales de los contenidos de una aplicación web –la Arquitectura de la Información, y los elementos de análisis y diseño procesables por desarrolladores en forma de representaciones UML, aumentando así la automatización e interoperabilidad en el desarrollo de aplicaciones web interactivas.

Palabras clave

Arquitectura de la Información, Desarrollo por el Usuario Final, Análisis y Diseño de Sistemas Interactivos.

1 Introducción

En nuestra época tecnológica actual, la Arquitectura de la Información (AI) es un paradigma creciente que se está incorporando paulatinamente en la mayoría de los proyectos Software. Se define como la ciencia de la estructuración, organización y gestión de la información donde el arte del etiquetado, el hallazgo y la facilidad de uso son también importantes [2]. La AI cubre indistintamente aspectos tales como el diseño de sitios web, interfaces de dispositivos móviles, interfaces de máquinas dispensadoras, interfaces de juegos electrónicos, etc. Su principal objetivo es facilitar al máximo los procesos de comprensión y asimilación de la información, así como las tareas que ejecutan los usuarios en un espacio de información definido [8]. La forma en que las personas interactúan con los entornos ricos en información digital está directamente influenciada por la Arquitectura de Información [1].

El Arquitecto de la Información trabaja sobre las fases tempranas del Software interactivo, principalmente en aplicaciones web, tratando de encontrar un puente que una los conocimientos conceptuales que plantean los usuarios, dentro del espacio del problema, y la información de diseño que luego necesitarán los ingenieros de la solución (Ingenieros del Software), que se encargarán de diseñar la interfaz web final.

Sin embargo, muchas veces, los roles del Arquitecto de la Información y del Ingeniero del Software no suelen coincidir, ya que el Arquitecto de la Información puede tener un perfil menos técnico y más orientado a tareas conceptuales o relacionadas con la ergonomía, lo cual hace necesario cierto grado de interoperabilidad y consonancia entre el output que genera el Arquitecto de la Información y el input que espera el Ingeniero del Software para comenzar con el diseño de la aplicación web interactiva. Si esta continuidad se pudiera hacer de forma automática, el tiempo y esfuerzo de realización del proyecto se reduciría, permitiendo a cada

experto concentrarse en su trabajo en función de sus conocimientos y minimizando el tiempo de trasiego de información entre uno y otro profesional [7].

El objetivo de este trabajo es atacar este problema y unir puentes entre las tareas del Arquitecto de la Información y las del Ingeniero del Software, proponiendo para ello una herramienta CASE (Computer Aided Software Engineering) denominada InterArch (Interoperable Information Architecture), la cual permite a los expertos en el dominio del problema concentrarse en el análisis de contenidos y navegación, generando la herramienta de forma automática las clases de contenido para el Ingeniero del Software en formato UML y concretando además, de forma inmediata, elementos del dominio de la solución.

La estructura del artículo que se presenta es la siguiente. A continuación, en la Sección 2, se detalla el trabajo relacionado existente. Posteriormente, la Sección 3 entra en detalle en nuestra propuesta y proporciona un caso de estudio que valida los planteamientos iníciales que dan soporte a nuestra investigación. Finalmente, en la Sección 4, se discuten las conclusiones y las líneas de trabajo futuro.

2 Trabajo Relacionado

El trabajo del Arquitecto de la Información se basa principalmente en la generación de un conjunto de material entregable y en su posterior evaluación. Para realizar este trabajo, en función de las necesidades, existen distintos tipos de herramientas que permiten desde la creación de diagramas para representar la Arquitectura de Información y concretar la interacción en equipos de desarrollo [3], hasta herramientas que permiten realizar el análisis y evaluación de la Arquitectura de la Información de un sitio web [4]. En [10] se presentan algunas de estas técnicas y herramientas generales más usadas por el profesional de la información en el proceso de análisis de productos electrónicos (ver Tabla 1).

Dentro de las técnicas de representación de información, en [9] se realiza un análisis y evaluación de una serie de herramientas diseñadas para facilitar la creación de prototipos web. Las herramientas analizadas fueron: Axure, Visio Profesional, OmniGraffle, Denim, Conceptdraw Pro, Smartdraw y Pencil Project, Mockflow, iPlotz, Pidoco, Lovely Chart, MockingBird y Lumzy. En este trabajo, los autores utilizan características de operatividad, funcionalidad, plantillas y soporte como parámetros para evaluar las herramientas, obteniendo como resultado que las herramientas mejor evaluadas son: iPlotz, Smartdraw, Conceptdraw y Pencil Project.

Las soluciones analizadas están formadas por un conjunto de herramientas de escritorio y en línea utilizadas por los profesionales dedicados al diseño de la interacción en general, las cuales incorporan librerías formadas por una veintena de elementos gráficos para el prototipado web que permiten la gestión y edición de sus elementos y la incorporación de nuevos componentes gráficos externos. Adicionalmente, permiten incluir anotaciones y notas de pie de página, la edición colaborativa del prototipado y la creación de prototipos dinámicos.

Tabla 1. Herramientas más habituales en la Arquitectura de Información

Técnica	**Herramientas**
Interacción con el usuario	Reunión, entrevista y encuesta, diseño de escenarios y diseño participativo
Interacción con el context	Evaluación de productos similares y análisis de la competencia
Matemáticas (co-ocurrencia)	Clasificación de tarjetas (cardsorting) y análisis de secuencia
Representación de información	Representación de etiquetas y prototipado (creación de maquetas)

En general, las herramientas en línea existentes suelen ser menos expresivas y completas que las soluciones de escritorio. Por otro lado, en las herramientas más comunes existen dificultades para conectar el output del Arquitecto de la Información y el input que espera el Ingeniero del Software. Esto se intenta resolver generando distintos formatos exportables de imágenes y HTML. No obstante, esta solución hace desaparecer detalles semánticos importantes relativos al análisis, y dificulta la interoperabilidad y seguimiento si se utilizan herramientas posteriores más precisas.

Por otro lado, ninguna de las herramientas existentes, ni siquiera las comentadas anteriormente, generan de forma automática o semi-automática información que permita obtener, a partir de los contenidos de información, los diagramas de clase y objetos de contenido que puedan ser utilizados por ingenieros o analistas software para dar continuidad al resto de fases del ciclo de vida del proyecto.

3 Solución Propuesta

A modo general, es difícil estipular los límites operacionales de la Arquitectura de la Información, lo que hace necesario muchas veces el uso de diversos tipos de herramientas y estándares, como se ha visto en la sección anterior. No obstante, es posible hacer un resumen de los productos más comunes que el Arquitecto de la Información debe crear para el análisis de la Arquitectura de la Información de una aplicación web interactiva. De todos estos productos, los más importantes se corresponden con los blueprints (planos), los wireframes (bosquejos o prototipos de baja fidelidad), los modelos de contenido y los vocabularios controlados [8, 2]. Estos productos representan un conocimiento importante para los diferentes profesionales que participan en los proyectos de construcción de sitios web, y se hace indispensable compartirlos en diferentes formatos y plataformas para su posterior utilización por los demás profesionales que integran el equipo de trabajo.

Sin embargo, algunos de estos productos, en especial los blueprints junto con los modelos de contenido, son especialmente trascendentales para analistas e ingenieros software, e incluso son susceptibles de un tratamiento automático que permita, a partir de ellos, generar de forma automática los diagramas de clase de contenido y los objetos de contenido que definirán la aplicación en el dominio de la solución. Esta es la razón por la cual en nuestro trabajo nos hemos concentrado en estos elementos esenciales de cara a una automatización de las salidas del proceso de análisis de la Arquitectura de la Información.

Para que la generación de información automática sea posible, hemos diseñado una herramienta CASE denominada InterArch. Esta herramienta está basada en dos principios esenciales. Primero, dado que habitualmente el Arquitecto de la Información posee un perfil menos técnico y más orientado a aspectos de diseño y organización de la información, InterArch le permitirá concentrarse en sus tareas de análisis conceptual dentro del dominio del problema. Esto facilitará que el profesional de la información pueda elaborar sus productos de forma habitual. Segundo, en base al análisis inicial realizado por el Arquitecto de la Información, InterArch generará la información UML para analistas e ingenieros software de forma automática, concretando los elementos que tienen su correspondencia con los diagramas de clase y objetos de contenido que utilizan los profesionales del software. Para ello, dicha información se generará en un formato textual y transportable en XMI[1] , de forma que sea procesable por cualquier herramienta CASE existente con objeto de dar continuidad al resto de fases y actividades del proyecto.

3.1 Herramienta Case para el Arquitecto de la Información

La herramienta de autor InterArch está compuesta principalmente por una serie de procesos que se encargan de la gestión y transformación de modelos en un entorno visual orientado al Arquitecto de la Información. Como se especifica en la Figura 1, estos procesos comprenden el modelado visual de los elementos conceptuales requeridos por el profesional de la información, la transformación del modelo visual en un modelo intermedio, y la generación textual transportable del análisis de

[1] XMI – XML Metadata Interchange, es el estándar definido por la OMG para el intercambio de diagramas UML: www.omg. org

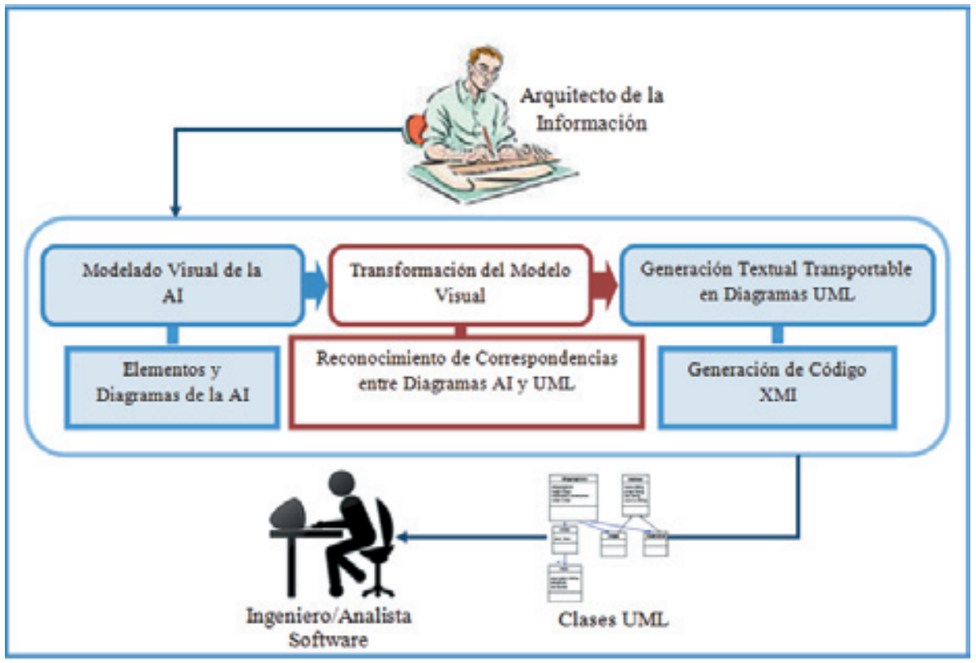

Fig. 1. Detalle Arquitectónico de la Herramienta CASE InterArch

la información en diagramas UML. Dichos procesos tienen por objetivo tomar como entrada el diseño visual de diagramas del Arquitecto de la Información y generar como salida diagramas UML para el analista e ingeniero software.

La idea principal que subyace detrás de estos componentes arquitecturales es permitir al Arquitecto de la Información trabajar en la elaboración visual de sus diagramas de forma transparente, pero incorporando por debajo una capa de interpretación capaz de reconocer las distintas correspondencias entre estos diagramas de análisis de la información y las clases UML requeridas por los Ingenieros del Software. La transformación del modelado visual se realiza en base a una serie de reglas de relación y asociación que se aplican al modelo conceptual inicial realizado por el Arquitecto de la Información, generando como elemento de salida un conjunto de diagramas UML en formato transportable XMI.

En las siguientes secciones, se explicará en detalle cada uno de los componentes arquitecturales de la herramienta InterArch.

3.1.1 Modelado Visual de la Arquitectura de la Información

El modelado visual-conceptual de los elementos arquitectónicos de la información se lleva a cabo mediante la interfaz de usuario principal de la herramienta InterArch. Esta interfaz es el entorno principal de trabajo del Arquitecto de la Información, y consta de elementos visuales para elaborar diagramas en un entorno funcional para manipular e interactuar con los elementos visuales a través de distintas opciones de formato y edición. En la Figura 2 se muestra esta interfaz de usuario, cuyas partes principales están marcadas con letras mayúsculas (A, B y C) en la Figura 2.

En la parte A de la Figura 2, se muestran las opciones de formato y edición habituales de una aplicación de estas características para manipular elementos dentro del entorno de trabajo: archivos, edición, formato y estilos, vistas del entorno de trabajo, etc. En la parte B de la Figura

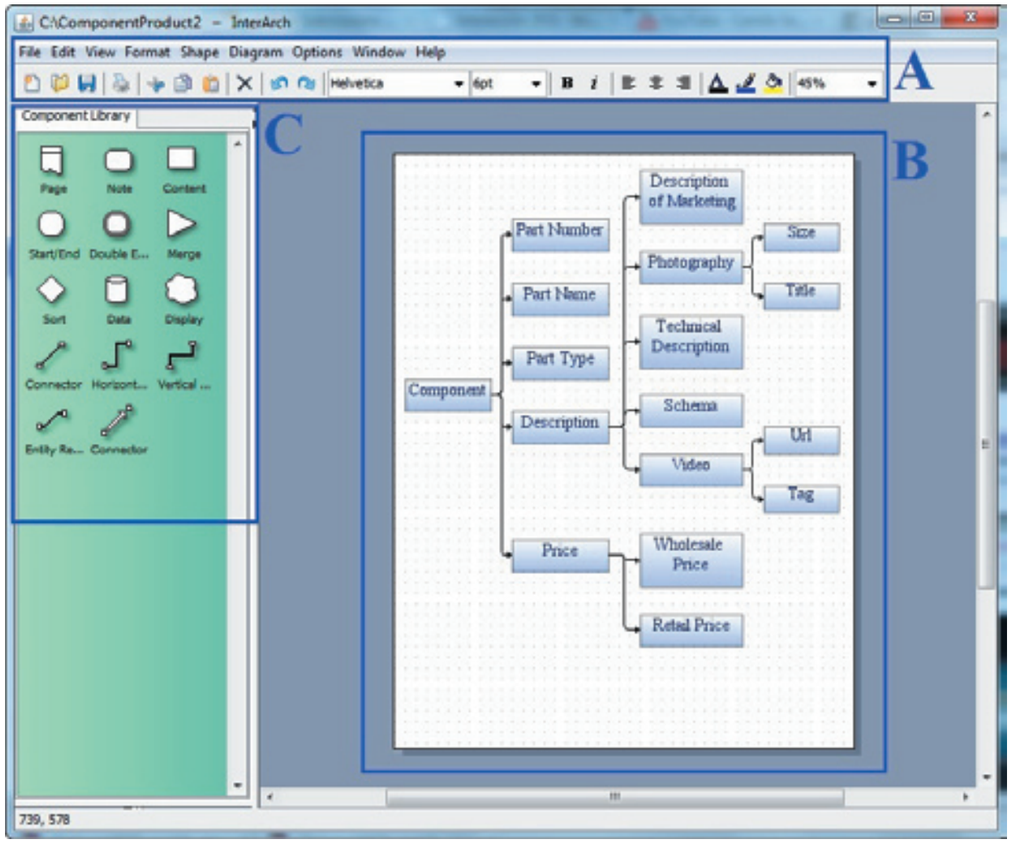

Fig. 2. Interfaz de Usuario de InterArch dividida en tres partes: A, B y C

2, se detalla el entorno de trabajo principal que permite manipular y relacionar los distintos elementos visuales de que dispone la herramienta. En el ejemplo que aparece, se pueden apreciar las relaciones entre los elementos de contenido representadas, las cuales describen la estructuración y precios de los componentes de cada producto en una tienda on-line. En la parte C de la Figura 2, se muestran los iconos de trabajo de los distintos elementos visuales que permiten enriquecer la interpretación visual de los diagramas elaborados por el Arquitecto de la Información. Existen tres tipos de elementos principales para el modelado visual: los elementos para la interpretación de contenido del modelado visual-conceptual de la AI, los elementos de enlace que permiten definir los tipos de asociaciones y relaciones entre los elementos de contenido, y los elementos de etiquetado que permiten incorporar información adicional (semántica) al modelado.

3.1.2 Transformación del Modelo Visual

La transformación del modelo visual comprende la identificación de cada elemento visual elaborado por el Arquitecto de la Información para componer, posteriormente, los diagramas UML utilizados por el ingeniero o analista software. Esto se lleva a cabo a través de reglas de asociación y relación de los elementos visuales tanto de forma individual como grupal. Hay dos tipos de reglas: las reglas fijas que generan clases de análisis de acuerdo a un conjunto de criterios respecto a la jerarquía y relación entre los elementos visuales elaborados por el Arquitecto de la Información, y las reglas configurables que permiten generar información de diseño y especificar en detalle el tipo y nivel de dependencia entre relaciones, además de la cardinalidad y navegabilidad a partir de los esquemas conceptuales elaborados por el Arquitecto de la Información. Algunas de estas reglas configurables toman valores por defecto en la generación de clases, los cuales pueden ser modificados por analistas e ingenieros para adaptar, con mínimo esfuerzo, la información de diseño correspondiente.

3.1.3 Generación Textual Transportable de Diagramas UML

Como paso final, InterArch genera los diagramas de clases de contenido UML en formato XMI a partir del análisis de la información llevado a cabo por el Arquitecto de la información, condicionado además por las reglas de asociación y relación descritas anteriormente. El XMI proporciona un estándar de facto que permite la edición y personalización de diagramas UML, por parte de analistas e ingenieros, para ser incorporados y reutilizados como documentación del proyecto software. El fin es poder continuar con el análisis y diseño del proyecto en curso y conjugar dichos diagramas con la parte funcional de la aplicación web interactiva que se desea crear mediante otro tipo de herramientas CASE utilizadas durante el ciclo de vida.

3.2 Caso de Estudio

Con objeto de describir más en profundidad el funcionamiento de InterArch, se describe un caso de estudio específico bajo nuestra herramienta.

Supongamos que el Arquitecto de la Información desea trabajar sobre un modelo de contenido de la información que describe la composición, descripción y precios de los componentes de cada producto en una tienda on-line. Este ejemplo se corresponde con el diagrama que aparece en la Figura 2 presentada anteriormente.

En este caso, el objeto de contenido Component, más general, se describe a su vez por medio de cinco objetos de contenido (Part Number, Part Name, Part Type, Description y Price), donde Description y Price presentan a su vez datos compuestos que se definen en base a los objetos de contenido que descienden de ellos de forma jerárquica. El objeto Description se compone de objetos que representan distintos tipos de descripciones del producto (Video, Schema, Technical Description, Photography y Description

of Marketing), de los cuales Video (con Url y Tag) y Photography (con Size y Title) presentan también datos compuestos. El objeto Price, indica los precios al por mayor y al por menor que puede tener el componente en cuestión: Wholesale Price y Retail Price.

3.2.1 Reglas de asociación y relación para los elementos visuales

Una vez que el Arquitecto de la Información ha creado el diagrama de contenidos, cuyo resultado se visualiza en la parte B de la Figura 2, lo siguiente sería generar las clases de contenido, aprovechables por el Ingeniero Software, en formato UML. Este paso es completamente trasparente para el Arquitecto de la Información.

Para la generación de diagramas UML a partir de los diagramas AI, se aplican las reglas referidas en secciones anteriores. En este caso concreto, a partir del elemento principal, Component, se generaría una clase principal y sus elementos descendientes se evaluarían de la siguiente forma: los elementos terminales, y descendientes directamente del elemento principal, pasan a ser atributos de la clase Component. Si estos elementos tienen a su vez descendientes, estos pasan a ser nuevas clases relacionadas directamente con el elemento Component. Si los elementos descendientes generan atributos y si son elementos compuestos, de forma recursiva generarían nuevas clases relacionadas con el elemento del que descienden. La aplicación de esta regla genera una clase Component con cinco atributos: PartNumber, PartName, PartType, Description y Price. Siguiendo con la aplicación de estas reglas, el elemento Price, al descender directamente del elemento principal y contener a su vez elementos descendientes Wholesale Price y Retail Price se transforma en una nueva clase, y sus elementos Wholesale Price y Retail Price en atributos para la clase Price. En cuanto a los métodos de clase, se generan por defecto tres métodos por cada atributo que representa una clase agregada (get, set y new).

En la Figura 3 se puede apreciar la visualización final de las clases UML conceptuales que se generan. El diagrama de clases está compuesto por una clase principal Component que tiene una relación de agregación con las clases Description y Price. A su vez, la clase Description tienen relación de agregación con las clases Video y Photography. Esta generación se hace en base a las reglas fijas anteriores que facilitan la automatización del proceso. Sin embargo, podría ser necesario, para una solución de diseño más concreta, establecer la cardinalidad entre relaciones, o incluso establecer relaciones de composición (agregación fuerte) entre ellas. Esto puede suceder en relaciones entre clases como Price y Description con Component, ya que probablemente no tenga sentido que exista el precio sin el componente (es decir, no tiene sentido que exista la parte sin el todo). El tipo y nivel de dependencia entre relaciones, además de la cardinalidad y navegabilidad, pueden ser modificados por el ingeniero o analista software mediante las reglas configurables de InterArch.

La aplicación de estas reglas genera como salida un diagrama de clases UML que puede ser almacenado en formato textual XMI. A modo de ejemplo ilustrativo, en el siguiente fragmento de código se puede apreciar la representación XMI de la clase Component con parte de sus atributos, operaciones, y su la relación con la clase Price, según el ejemplo anterior y el diagrama de salida de la Figura 3.

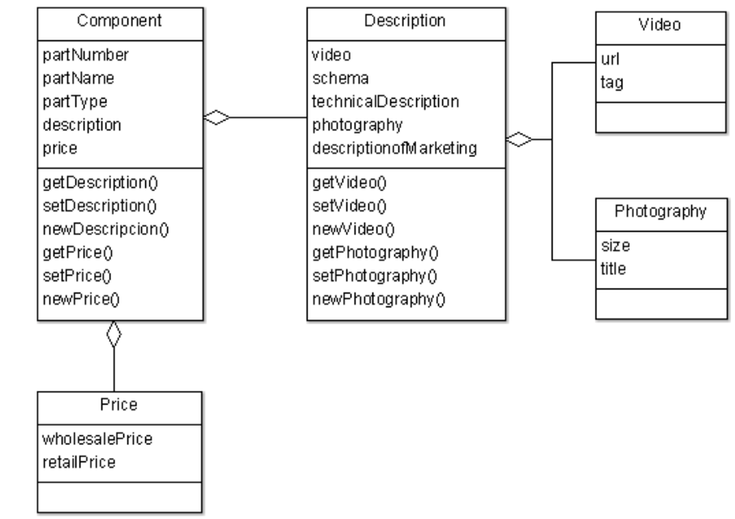

Fig. 3. Diagrama de clases UML resultante del análisis de la AI del ejemplo propuesto

<UML:Class xmi.id = ‘x232’ name = ‘Component’> <UML:Classifier.
feature>

<UML:Attribute xmi.id = ‘x232:87B’ name = ‘Description’ visibility =
‘public’></UML:Attribute>...

<UML:Operation xmi.id = ‘x232:02C’ name = ‘getDescription’ visibility =
‘public’></UML:Operation>...

<UML:Class xmi.id = ‘x235’ name = ‘Price’>...

<UML:Association xmi.id=’868’> <UML:Association.connection>
<UML:AssociationEnd xmi.id=’889’ aggregation=’aggregate’>...

<UML:AssociationEnd.participant><UML:Class xmi.idref=’x232’/></
UML:AssociationEnd.participant></UML:AssociationEnd><UML:Asso
ciationEnd xmi.id = ‘874’ aggregation=’none’><UML:AssociationEnd.
participant><UML:Class xmi.idref=’x235’>...

El archivo generado en formato XMI es transportable, y puede ser utilizado en cualquier herramienta de diagramación UML que soporte el uso de importaciones XMI, como por ejemplo: ArgoUML, StarUML, BOUML, Visual Paradigm, Circa y Mia-Generation, entre otras.

4 Conclusiones y Trabajo Futuro

En este artículo, proponemos una solución automatizada válida para el análisis y diseño de aplicaciones web interactivas consistente en una herramienta CASE para el Arquitecto de la Información con pocos o inexistentes conocimientos en ingeniería del Software. Nuestra hipótesis de partida se basa en que las personas que llevan a cabo el análisis de la Arquitectura de la Información de una aplicación web no tienen por qué tener el mismo nivel de conocimientos técnicos que los analistas e ingenieros software que acabarán diseñando la aplicación, aunque debería haber una continuidad entre la información de análisis conceptual que generan estos profesionales y la información de análisis y diseño que necesitan los analistas informáticos para llevar a cabo el modelado de la aplicación. Esta información se materializa en forma de diagramas UML, ampliamente utilizados para detallar aspectos de análisis y diseño.

Nuestra solución pretende unir puentes entre las representaciones conceptuales de alto nivel de la Arquitectura de la Información y la representación no funcional de clases de análisis y diseño necesarias para implementar dichos contenidos, los cuales se unirán, utilizando cualquier tipo de herramienta de modelado del proyecto, a las clases funcionales del dominio de la solución. Para ello, proponemos una herramienta de autor, llamada InterArch, que permite generar automáticamente diagramas de clases UML a partir de la definición conceptual de la Arquitectura de la Información de un sitio web interactivo, utilizando XMI como lenguaje intermedio de representación que además puede ser procesado por un gran número de herramientas CASE, lo que permite una mayor interoperabilidad para integrar clases UML funcionales y no funcionales durante el ciclo de vida de una aplicación web interactiva.

La investigación que aquí se presenta está basada en el EUD o paradigma del Desarrollo por el Usuario Final (del inglés End-User Deveopment) [5, 7]. El EUD incentiva la creación de artefactos software que permitan a usuarios no expertos en informática crear o customizar aplicaciones fácilmente. Más en concreto, nuestro trabajo se acerca a paradigmas relacionados como el EUSE (End-User Software Engineering), cuyo objetivo es involucrar a usuarios no expertos en computación, como sucede muchas veces con los Arquitectos de la Información, en aspectos concretos del desarrollo de un proyecto informático relacionados con el análisis y desarrollo de Software [6].

Como trabajo futuro, pretendemos mejorar la herramienta y dotarla de nuevas funcionalidades para la generación de elementos enriquecidos de diseño: componentes y objetos de diseño de contenido detallados. También, tenemos pensado aumentar la expresividad en la parte de reglas para la transformación del modelo visual, de forma que analistas e ingenieros software puedan adaptar aún más la generación de información para el proyecto de acuerdo a sus necesidades. Por último, se pretende realizar una evaluación de la usabilidad de la herramienta con Arquitectos de la Información no expertos en ingeniería del software.

Agradecimientos

Esta investigación ha sido subvencionada por el Ministerio de Educación, proyectoTIN2008-02081/TIN, así como por la DGUI de la Comunidad de Madrid, proyecto S2009/TIC-1650 y, junto con la UAM, a través del proyecto CCG10-UAM/TIC-5772.

Bibliografía

1. **Elaine G. T**.: Information interaction: Providing a framework for information architecture. Journal of the American Society for Information Science and Technology, Vol. 53, Iss. 10, 855-862 (2002)

2. **Erlin,Yunus, Y. A., Rahman, A. Abdul.**: The evolution of Information Architecture. Information Technology, ITSim 2008. International Symposiumon, vol.4, no., pp.1-6 (2008)

3. **Garrett, J. J.**: A visual vocabulary for describing information architecture and interaction design. www.jjg.net/ia/visvocab (2002)

4. **Katsanos, C., Tselios, N., Avouris, N.**: AutoCardSorter: designing the information architecture of a web site using latent semantic analysis. In Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI ‘08). ACM, New York, NY, USA, 875-878 (2008)

5. **Klann, M., Fit, F.**: End-User Development. D1.1 Roadmap. Proceedings of the End User Development Workshop at CHI’2003 Conference. Ft. Lauderdale, Florida, USA. April 5-10 (2003)

6. **Lieberman, H., Paternò, F., Wulf, V.**: End-User Development. Human Computer Interaction Series. Springer Verlag (2006)

7. **Macías, J.A.**: Aspectos Pragmáticos en el Desarrollo por el Usuario Final. Revista Novática, Nº 175, año XXXI, pp. 45-47, mayo-junio (2005)

8. **Morville, P., Rosenfeld, L.**: Information Architecture for the World Wide Web. O’Reilly Media”, O’Reilly Media Inc., Third Edition, (2006)

9. **Pérez, M., Codina, L.**: Sotfware de prototipado para la arquitectura de la información: funcionalidad y evaluación. El profesional de la información, 417-424 (2010)

10. **Ronda, R.**: Revisión de técnicas de arquitectura de información. No Solo Usabilidad, nº 6, 2007. <nosolousabilidad.com>. (2007)

Sobre los autores

Luis A. Rojas Finalizó sus estudios de Máster en Ingeniería Informática y de Telecomunicación en la Universidad Autónoma de Madrid. Actualmente, realiza la Tesis Doctoral en la Escuela Politécnica Superior de dicha universidad, trabajando en temas de Interacción Persona-Ordenador en el Grupo GHIA (Grupo de Herramientas Interactivas Avanzadas), bajo la dirección de Dr. José Antonio Macías. Desde 2001, trabaja en el Departamento de Ingeniería de Sistemas en la ComisiónChilena de Energía Nuclear en temas relacionados con la implementación de tecnologías de información en los distintos procesos productivos, de gestión y de investigación.

luisalberto.rojas@estudiante.uam.es

José A. Macías Doctor en Ingeniería Informática y profesor permanente en el Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid.Su línea principal de investigación se centra en la Interacción Persona-Ordenador. Es Vicepresidente de AIPO (Asociación Interacción Persona-Ordenador) y Co-Chair del SIGCHI (Grupo de Interés en Interacción Persona-Ordenador) de ACM en España. Ha sido investigador en el CNR de Italia. Además, ha participado en diferentes proyectos relacionados con la Interacción Persona-Ordenador y la Ingeniería del Software, contando con gran cantidad de artículos en diferentes libros, revistas y congresos de prestigio sobre dicha temática.

j.macias@uam.es

© Gregorio G. Reche

Soporte para el Análisis de la Arquitectura de la Información en Aplicaciones Web

En los equipos de desarrollo de aplicaciones web interactivas intervienen expertos no informáticos centrados en el modelado conceptual y la concreción de aspectos no funcionales de la aplicación. Este análisis de la información es luego procesado por analistas e ingenieros en las fases técnicas del proyecto. Muchas veces, este trasiego de información impide una automatización implícita, debido a la que la información procesada por los distintos profesionales tiene un nivel conceptual diferente, lo que supone un coste mayor, en tiempo y en esfuerzo, en el desarrollo del proyecto. El objetivo de este trabajo es minimizar dichos costes mediante una herramienta CASE capaz de unir puentes entre las definiciones conceptuales de los contenidos de una aplicación web –la Arquitectura de la Información, y los elementos de análisis y diseño procesables por desarrolladores en forma de representaciones UML, aumentando así la automatización e interoperabilidad en el desarrollo de aplicaciones web interactivas.