UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA

PROYECTO FIN DE MASTER

"Workflow para el desarrollo de software para dispositivos móviles"

DIRECTORES: Juan Manuel Cueva Lovelle

B. Cristina Pelayo García-Bustelo

AUTORA: María Macías Alonso

Vº Bº del Director del Proyecto

Agradecimientos

En primer lugar quiero agradecer a mis padres todo su apoyo y confianza desde que empecé esta carrera, sin ellos esto no sería posible. Su comprensión, paciencia y dedicación han conseguido que llegue hasta aquí y que siga avanzando y creciendo como persona. Por todo esto y mucho más, GRACIAS papás!!

A David, mi novio, por estar siempre ahí, ayudarme, comprenderme y en definitiva apoyarme para que consiga mis objetivos más fácilmente. Eres mi compañero y contigo las alegrías son más dulces y las tristezas menos duras, GRACIAS.

A mis cuatro abuelos, Trinidad y Antonio , y Manuel y Elia, que siempre intentan apoyarme y darme ánimos en todo momento. Gracias a sus consejos cada día aprendo algo nuevo que me hace llegar a ser mejor persona, por todo eso y mucho más, GRACIAS.

A mi tíos, Ana y Valen y Paco y Regi, que siempre intentan entenderme y darme buenos consejos. A mis primos Ana y Paquito que son muy especiales para mi, GRACIAS.

A mis suegros, Ana y Santi, que intentan apoyarme y ayudarme en todo lo que pueden y se preocupan por mi, GRACIAS. A mi cuñada Yle y su marido Rober que son personas muy importantes para mi junto con sus hijas Yle y Lucía, dos soles que iluminan mi cara cuando las miro y me llaman tia... GRACIAS.

A Roxana y Javi, por ser mis amigos y estar pendientes de mi cada vez que estoy sola en Oviedo, por escuchar mis presentaciones e intentar que las haga bien, porque a pesar de que este año no nos hemos visto todo lo que quisieramos siempre que os he necesitado habéis estado ahí, por todo eso GRACIAS.

A las Euitianas, Paola, Esther Álvarez, Esther Abascal, Susana, Vane, Almu, Omayra por esas cenas en las que lo pasamos tan bien, este año han sido menos, pero aún así me encanta salir con vosotras, GRACIAS.

A Bea y Hugo, porque desde que fui a vuestra boda, empezó una amistad que sólo hace que crecer día a día y que vuestro apoyo es muy importante para mí, GRACIAS.

A Weena, por toda la ayuda que me has prestado para realizar muchas partes de este proyecto, por acompañarme hasta el último día y por tu comprensión en mis días insoportables, GRACIAS.

A mis compañeros del máster de ingeniería web, en especial al grupo de tarde, que a pesar de lo duro que es este máster, con vosotros ha sido mucho más

llevadero, en especial a David Orozco, mexicanito, por todas nuestras charlas camino de casa al salir del máster, por toda la ayuda que me prestaste para realizar algunas prácticas del máster y por esos partidos de tenis en los que lo menos importante es el resultado final, a Mel circus, por todo el apoyo que me has dado durante estos ultimos meses, desde que empezamos a ser hamiguitos desde la asignatura de e-learning, tu compañia, tus consejos y tu desorden hacen que un día normal se convierta en especial, GRACIAS.

A mis compañeros del laboratorio OOTLab, que siempre que he necesitado su ayuda me la han prestado gustosamente y me he sentido como en casa desde mi primer día, en especial a Jaime por ayudarme tanto en el desarrollo de muchas partes del proyecto y a Nat por avisarme todos los días para que vaya al pincho y se alegrara tanto el primer día que me vio aparecer por la puerta, GRACIAS.

Por último y no por ello menos importante, muchas gracias a mis directores de proyecto Juan Manuel Cueva Lovelle y Cristina Pelayo García-Bustelo, por brindarme la oportunidad de pertenecer a este grupo de investigación que tanto me gusta, por ofrecerme la oportunidad de realizar un proyecto útil y atractivo y por todo el apoyo y la ayuda recibida a lo largo del desarrollo del proyecto, GRACIAS.

Muchas gracias a todos y cada uno de vosotros, todos formáis parte de mi vida y a mi me encanta ser parte de las vuestras GRACIAS!

Resumen

En este proyecto se describe la definición de un meta-modelo de manera gráfica mediante la herramienta WebDesing. A través del meta-modelo se definirá el modelo estableciendo unas reglas definidas en ficheros XML. Estos serán interpretados y junto con las plantillas de transformación y como resultado se obtendrá un sitio web académico completo que será visible desde cualquier navegador.

Palabras Clave

ASP.NET, generación automática de aplicaciones, asignatura, personalizable, MDA, MDE, DSL, DSM.

Abstract

In this proyect we introduce the definition of a graphical meta-model through the use of the WebDesign tool. Using this meta-model we will be able to define the model by a set of rules defined in XML files. These files will be interpreted and in combination with the transformation templates they will generate a complete academic website that can be visited from any web explorer.

Keywords

ASP.NET, automatic generation of applications, subject, customizable, MDA, MDE, DSL, DSM.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

Cada vez más usuarios utilizan sus dispositivos móviles para navegar por internet, por lo que desarrollar sitios web optimizados para estos dispositivos se está volviendo crítico para los desarrolladores web. HTML5 es una excelente opción para que los desarrolladores que quieren crear un sitio compatible con los dispositivos móviles más modernos y comenzar a ofrecer nuevos e innovadores servicios haciendo uso de todas las nuevas características del HTML5. La idea que se presenta en este proyecto es proporcionar herramientas que ayuden a la generación de sitios web de forma rápida, sencilla y con pocos conocimientos en programación. Las personas que se encargan de crear este tipo de sitios puede que no tengan muchos conocimientos informáticos y para ellos suele ser un quebradero de cabeza el lograr que el sitio web funcione correctamente. Pensando en este tipo de usuarios y dado el nexo común de los sitios web académicos, se ha construido una herramienta que permite a un usuario con pocos conocimientos en programación ver sus necesidades cubiertas y lograr construir un sitio web académico fácil de actualizar y de mantener. Además de ser multilenguaje, altamente personalizable y visible en cualquier dispositivo móvil, ya que el sitio web generado esta desarrollado en HTML5.

1.2. Finalidad del Proyecto

Este proyecto se centra en conseguir acercar el desarrollo y puesta en marcha de sitios web académicos accesibles desde dispositivos móviles, a personas sin conocimientos en informática. Para lograrlo, se va a construir una herramienta gráfica mediante la cual será posible construir un sitio web arrastrando elementos y de forma intuitiva. A pesar de ser un método fácil, esto no quiere decir, que el sitio web generado resultante sea simple, ya que puede disponer del comportamiento que el usuario que construye el sitio desee. Existen varias opciones, puede escoger un sitio web que solamente muestre información de ma-

nera estática y nada más, otra opción es que además de mostrar información tenga un formulario de contacto, otra opción sería que mostrara la información y tuviera un formulario de preinscripción y por último que tuviese la resolución de la preinscripción. En función de las necesidades del usuario que desarrolle el sitio web académico, éste se adaptará a ellas y le dará soporte de forma que es posible generar sitios web académicos a la carta y los cuales serán visibles desde cualquier dispositivo móvil que contenga un navegador que interprete HTML5. Por lo que la finalidad de este proyecto es adaptarse al desarrollo de un sitio web académico en función de las necesidades que tenga cada uno y proporcionando al sitio web generado la mayor visibilidad posible, ya que al generar HTML5 será interpretado por cualquier navegador.

1.3. Organización del documento

Este documento de Trabajo Fin de Máster está estructurado en capítulos, de tal forma que cada uno de estos hace referencia a una parte concreta del trabajo realizado durante la realización del proyecto. A continuación se realiza un pequeño recorrido por cada uno de los capítulos con el fin de establecer los contenidos que se repasarán en cada uno de ellos.

El capítulo segundo, titulado "Fijación de Objetivos" contiene una enumeración de los objetivos que persigue este trabajo así como el establecimiento de posibles ámbitos de aplicación donde introducir las ideas contenidas en este proyecto.

El capítulo tres, "Estado Actual de los Conocimientos Científico-Técnicos", describe la situación actual con respecto a aquellos aspectos teóricos que están directamente relacionados con el trabajo que aquí se presenta. En concreto, este apartado realiza una introducción a las tecnologías que se utilizan para el desarrollo del proyecto, empezando por MDE, MDA, DSL o XML que ayudan a que se haga realidad la propuesta que se describe en este documento. Además de ofrecer un pequeño estado del arte de las herramientas que actualmente están en voga a la hora de diseñar sitios web de forma de que se establezca un marco común en el desarrollo y comparación de este proyecto con ellas.

En el cuarto capítulo, cuyo título es "Trabajo Propuesto", se presenta un meta-modelo que permitirá definir los modelos de páginas que se utilizaran para formar el sitio web académico final. A partir de la definición del meta-modelo se presentan dos prototipos: WebDesing, una herramienta gráfica que permite definir el modelo de cada página fácilmente y de forma intuitiva, e Web Generator, un generador de código que genera aplicaciones de forma automática en base a los modelos realizados con WebDesing.

El quinto capítulo, "Resultados Obtenidos", presenta las pruebas de satisfacción de usuarios que realizaron determinadas tareas con varias herramientas y con la que aquí se presenta y en el que se muestran sus opiniones sobre ellas.

El capítulo número 6, "Conclusiones y Trabajo Futuro", realiza un recorrido por las conclusiones obtenidas a partir de este estudio y presenta las líneas de trabajo futuro que quedan pendientes.

El siguiente apartado corresponde a la "Bibliografía", presenta al lector los trabajos anteriores que han sido consultados como punto de partida para la realización de este trabajo así como aquellas referencias web que han sido utilizadas en la elaboración del mismo.

El último capítulo de esta documentación es el apartado de "Apéndices", donde se han incluido una serie de documentos relacionados con este trabajo que no tenían cabida en ninguno de los capítulos anteriores del documento.

Capítulo 2

Fijación de Objetivos

Los objetivos marcados para la realización de este proyecto son los siguientes:

- Definición de un meta-modelo que defina los modelos de páginas disponibles.
- Creación de una herramienta de modelado que soporte la utilización de los elementos web como procesos que se pueden modelar.
- Desarrollo de una herramienta que analice los modelos generados con la herramienta anterior y genere una aplicación web, de tal manera que la configuración de la aplicación sea sencilla y funcional y realice la tarea específica definida en el modelo.
- Permitir la visualización de la pagina web resultante en cualquier dispositivo móvil.
- Desarrollar un sitio web multilenguaje de forma sencilla y rápida.
- Permitir la generación de sitios web académicos personalizados.
- Comprobar la eficacia de la herramienta generada con usuarios valorando su satisfacción.

2.1. Posibles Ámbitos de Aplicación

El ámbito de aplicación de esta investigación se centra en un escenario en el que se necesite definir un sitio web académico que esté sujeto a varios cambios y que en función de las necesidades de cada usuario se pueda adaptar a ellos y que sea accesible desde cualquier dispositivo móvil.

A pesar de que esta investigación esta más centrada en el ámbito de lo académico, se podrían aplicar las mismas premisas que para este desarrollo y se podría diseñar un generador de páginas web que no tienen porque ser solamente académicas.

En este proyecto se han realizado los prototipos que desarrollan las hipótesis planteadas en la tecnología .NET, pero debido al enfoque de MDE y al nivel de abstracción que ofrece, sería muy fácil desarrollar estos prototipos en otras plataformas.

Capítulo 3

Estado Actual de los Conocimientos Científico-Técnicos

3.1. Ingeniería dirigida por modelos (MDE)

La ingeniería dirigida por modelos surge como la respuesta de la ingeniería de software a la industrialización del desarrollo de software, MDA es la propuesta de la OMG, que centra sus esfuerzos en reconocer, que la interoperabilidad es fundamental y el desarrollo de modelos permite el desarrollo de otros modelos que luego al ser juntados proveerían la solución a todo un sistema e independiza el desarrollo de las tecnologías empleadas [15] El nivel de abstracción de los diferentes lenguajes de programación se ha ido incrementando con el paso del tiempo, así antes se hablaba de lenguajes en Binario, intercambiando información muy cerca del hardware y actualmente los lenguajes dinámicos están muy lejos del hardware y más cerca de propuestas como MDE que logran la interoperabilidad por definición. Para lograr estos propósitos este paradigma combina los siguientes conceptos:

- **Lenguajes de dominio especifico.** Formalizan la estructura de la aplicación, el comportamiento y los requisitos dentro de un dominio particular. Estos lenguajes DSL son descritos usando metamodelos, los cuales definen relaciones entre elementos dentro de un dominio.
- **Motores de transformación y generadores.** Analizan algunos aspectos de los modelos y crean los utensilios para manejar la información, código fuente, entradas de simulación, descripciones de uso XML.

Las herramientas MDE usan los conceptos anteriores y hacen más fácil para los ingenieros del software el soporte a la evolución del software, tanto en su lógica como en su tecnología. Mediante los DSLs se consiguen notaciones

distintas para cada tipo de sistema, las cuales están definidas formalmente por su metamodelo. De esta forma se tienen herramientas específicas para cada tipo de sistema, lo cual permite modelarlos de una manera más detallada y de acuerdo al dominio al que pertenecen. Mediante los motores de transformación se facilita la evolución de modelos, transformando de unos modelos a otros, según las reglas de transformación entre metamodelos. En el paradigma MDE cualquier concepto debe ser modelado. De esta manera, cualquier cambio o nueva propiedad del sistema debe ser mostrado en su modelo correspondiente. Con este paradigma, la parte de código es lo menos importante.
MDA nace con la idea de separar la especificación de la lógica operacional de un sistema. Como se puede apreciar en la figura 3.2 Los objetivos de MDA son la portabilidad, la interoperabilidad y la reusabilidad a través de la separación de la arquitectura.

3.1.1. Conceptos generales de MDE

En la figura 3.1 [24] se representan los conceptos más importantes y básicos que se pueden emplear bajo la tecnología MDE.

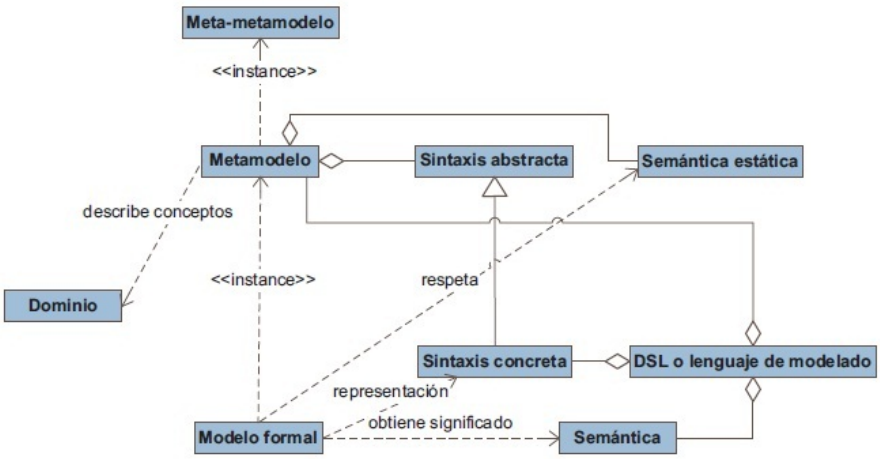


Figura 3.1: Conceptos generales de MDE

Dominio

El punto inicial de MDE siempre es un dominio, que delimita un campo de conocimiento. Hay dos tipos de dominios: los dominios tecnológicos, referentes a la tecnología de desarrollo de software y los dominios profesionales, referentes a los conceptos que manejar´a la aplicación. Los dominios pueden estar subdivididos en dominios más pequeños.

Metamodelo

El metamodelo sirve para describir de manera formal los conceptos relevantes que tiene el dominio. Además, es fundamental para conseguir automatizar el desarrollo de software [7].

Meta-metamodelo

Para que los metamodelos puedan ser reutilizables, interoperables y portables, tiene que existir otro metamodelo en un nivel de abstracción superior, describiendo de forma única los conceptos que sirven para representar cualquier metamodelo de cualquier dominio. El meta-metamodelo es quien realiza esa función. Tiene la peculiaridad de que se define a sí mismo.

Sintaxis abstracta y sintaxis concreta

Los metamodelos están formados por una sintaxis abstracta y por una semántica estática. La sintaxis abstracta se centra en los elementos a nivel conceptual mientras que la sintaxis concreta se centra en cómo se representan los conceptos. De ahí se deduce que un metamodelo tendría una sintaxis abstracta invariable pero podrían existir varias sintaxis concretas para representar los mismos conceptos. La sintaxis abstracta de un lenguaje especifica su estructura, es decir, las construcciones, sus propiedades y los conectores que puede tener dicho lenguaje. Generalmente, también se especifican reglas del lenguaje en el metamodelo y así se evita la mala práctica de tener que validar los modelos en los generadores de código. Cuanto primero se detecten anomalías, más sencilla será la tarea de los demás componentes. La sintaxis concreta de un lenguaje es necesaria para especificar la notación que los usuarios del lenguaje deben utilizar. Idealmente, cada concepto del dominio y del lenguaje se mapearía a una representación en la notación específica.

Semántica estática

La semántica estática de los metamodelos se basa en la sintaxis abstracta y tiene como misión hacer comprobaciones semánticas en los modelos para asegurar que están bien construidos.

Lenguajes de dominio específico

Los DSLs (Domain-Specific Languages o Lenguajes de Dominio Específico) [6] tienen el objetivo de poder expresar los conceptos de un dominio. Están formados por uno o varios metamodelos, una o varias sintaxis concretas, y habitualmente una herramienta que lo soporta para facilitar la usabilidad. Un DSL es un lenguaje definido pensado específicamente para abordar un problema específico de un dominio concreto y es el elemento principal de cualquier solución de dominio específico. Los DSLs son denominados frecuentemente lenguajes de modelado.

Modelos formales

Con la infraestructura definida hasta ahora, se puede hablar ya de modelos formales, que son el punto inicial desde el que se automatizan las posibles transformaciones a entidades de menor nivel de abstracción (p.e., a partir de un modelo se podría generar automáticamente una aplicación Java o .NET). Los modelos formales son instancias de los metamodelos y se representan gracias a una sintaxis concreta. Además, tienen que respetar la semántica estática que tenga el metamodelo para poder realizar construcciones coherentes dentro de un dominio.

Semántica del espacio del problema

La semántica de un DSL hace referencia a que cada concepto de los modelos tiene un significado: cada vez que se incluye un elemento en un modelo, lo que se está añadiendo es significado. A diferencia de lo que ocurre con los lenguajes de propósito general, gracias al uso de DSLs, se consigue que los conceptos de un lenguaje se mapeen directamente a conceptos del dominio que se modela, sin posibilidad de interpretaciones erróneas. La semántica de un DSL debe estar bien documentada o ser lo suficientemente intuitiva para que los creadores de los modelos sepan qu´e conceptos se están utilizando del espacio del problema; se deben asociar fácilmente los elementos de un lenguaje con los conceptos del dominio.

3.2. Model Driven Architecture (MDA)

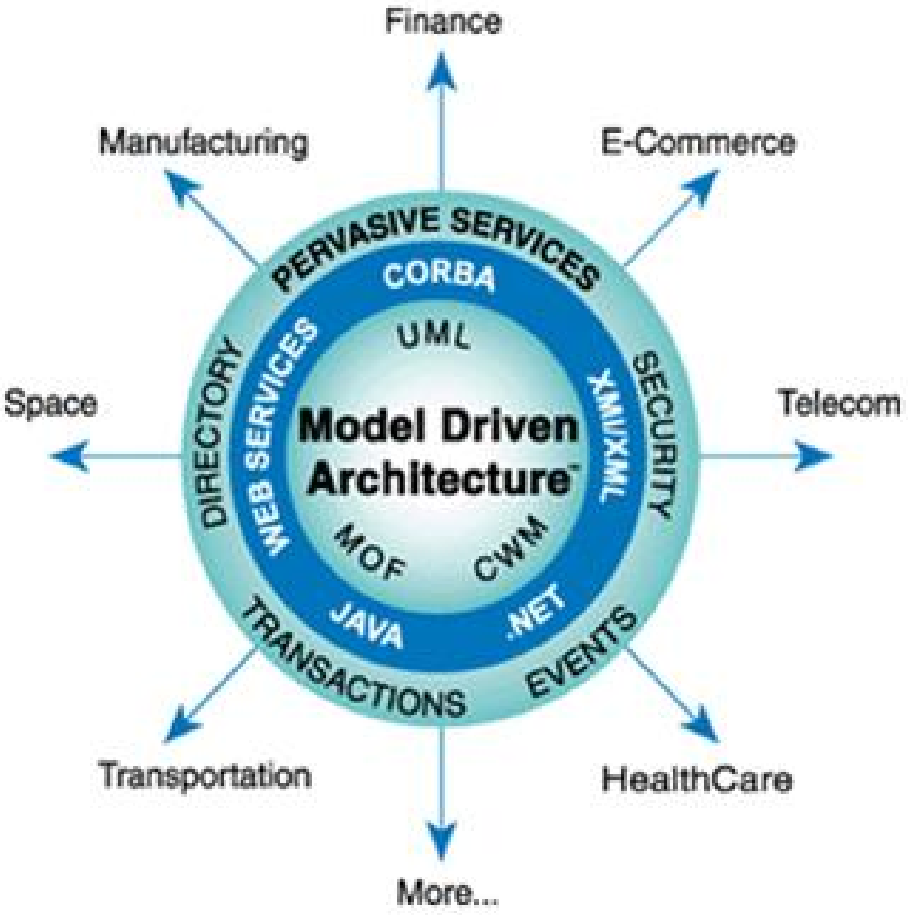


Figura 3.2: Definición de MDA

MDA es el acrónimo de Model Driven Architecture (Arquitectura Dirigida por Modelos), consiste en basar el desarrollo de software en modelos de forma que a partir del mismo modelo se puedan generar distintos programas con una base común.
Los objetivos principales de MDA son la portabilidad, interoperabilidad y la reusabilidad a través de la separación arquitectónica de conceptos [20][9]

3.2.1. Conceptos generales MDA

Sistema

Los conceptos de MDA serán expresados en términos del sistema existente. Este sistema puede incluir: un programa, un ordenador simple, una combinación de diferentes parte de sistemas, una conjunción de sistemas cada uno con un control diferente (personas, una empresa, federación de empresas)

Modelo

El modelo de un sistema es la descripción o especificación de ese sistema y su adaptación a un propósito determinado. Frecuentemente se presenta a un modelo como una combinación de dibujos y texto. El texto puede estar en un lenguaje de modelado o en lenguaje natural. En el contexto de MDA, los modelos son representaciones formales del funcionamiento, comportamiento o estructura del sistema que se está construyendo[21]. El término "formal" especifica que el lenguaje utilizado para describir el modelo debe de tener una semántica y sintaxis bien definida. Un concepto clave a la hora de construir modelos es el de "abstracción". Se debe ignorar la información que no es de interés para el dominio del problema a resolver. Los modelos representan elementos físicos o abstractos correspondientes a una realidad simplificada, adecuada al problema que se necesita resolver.

Dirigido por modelos

MDA es un acercamiento al diseño del sistema, para ello se incrementa la importancia de los modelos. Es dirigido por modelos porque aporta los medios para usar los modelos directamente para el entendimiento, diseño, construcción, despliegue, operación, mantenimiento y modificación.

Arquitectura

La arquitectura del sistema es una especificación de las partes y los conectores del sistema y las reglas para la interacción de las partes usando conectores. El MDA prescribe algunos tipos de modelos para ser utilizados, cómo tienen que ser preparados esos modelos y las relaciones entre los tipos de modelos.

Punto de vista y Vista

Un punto de vista de un sistema es la técnica para la abstracción utilizando un conjunto seleccionado de conceptos de la arquitectura y reglas estructurales, para centrarse en conceptos particulares dentro del sistema. La palabra "abstracción" es utilizada con su significado de proceso de suprimir los detalles seleccionados para establecer un modelo simplificado. Los conceptos y reglas pueden ser utilizados para formar un lenguaje de Punto de Vista. El MDA especifica tres puntos de vista sobre un sistema: un punto de vista de computación independiente, un punto de vista de plataforma independiente y un punto de

vista de plataforma específica. Una Vista o un modelo de punto de vista del sistema es una representación del sistema desde la perspectiva del punto de vista elegido.

Plataforma

Una plataforma es un conjunto de subsistemas y tecnologías que aportan un conjunto coherente de funcionalidad a través de interfaces y patrones de uso específico. En MDA, el término "plataforma" se utiliza generalmente para hacer referencia a los detalles tecnológicos y de ingeniería que no son relevantes de cara a la funcionalidad esencial del sistema. Esto uso del término se encuadra dentro de la separación fundamental que se realiza en MDA entre la especificación de un sistema y su plataforma de ejecución. Se define plataforma como la especificación de un entorno de ejecución para un conjunto de modelos [15]. Estará formada por un conjunto de subsistemas y tecnologías que proporcionan una funcionalidad determinada a través de una serie de interfaces y patrones de uso determinados [16]. Una aplicación soportada por la plataforma puede ser utilizada sin preocuparse por los detalles de cómo se suministra la funcionalidad por parte de la implementación de la plataforma.

- Tipos genéricos de plataformas: Objetos, batch, dataflow.
- Tipos de plataformas especificas del vendedor: J2EE [1], Microsoft .NET [22].

Aplicación

En la definición de MDA se utiliza la palabra aplicación para referirse a la funcionalidad que se está diseñando. Un sistema es descrito en términos de una o más aplicaciones que son soportadas por una o mas plataformas.

Independiencia de la Plataforma

Esta es una cualidad que el modelo debe tener. Esta característica es la que hace que el modelo sea independiente de las características de un tipo particular de plataforma. Como la mayoría de las características, la independencia de la plataforma es una cuestión de grado. Por esto, un modelo pudo asumir solamente la disponibilidad de características de un tipo muy general de plataforma, tal como invocación remota, mientras que otro modelo pudo asumir la disponibilidad un sistema particular de las herramientas para la plataforma de CORBA. Asimismo, un modelo pudo asumir la disponibilidad de una característica de un tipo particular de plataforma, mientras que otro modelo se pudo adaptar completamente a ese tipo de plataforma.

Puntos de Vista de los MDA

Punto de vista de computación independiente: se centra sobre la adaptación del sistema, y los requisitos para el sistema; los detalles de la estructura

y el proceso del sistema están ocultos o indeterminados. Punto de vista de independencia de plataforma: se centra en la operatividad del sistema ocultando los detalles sobre la plataforma en particular. Una vista independiente de la plataforma, muestra parte de la especificación completa que no cambia de una plataforma a otra. Esta vista puede utilizar un lenguaje de modelado de propósito general, o un lenguaje especifico al área donde el sistema será utilizado. Punto de vista de plataforma específica: combina la vista de independencia de plataforma con un enfoque adicional centrado en los detalles del uso por parte del sistema de una plataforma específica.

El modelo de computación independiente (CIM)

En inglés Computation Independent Model (CIM), es una vista del sistema desde el punto de vista de la independencia de computación. Un CIM no enseña la estructura de los sistemas. Se asume que el principal usuario del CIM, los conocedores del dominio, no conocen los modelos o artificios utilizados para realizar las funciones de los requerimientos que se articulan en el CIM. Además el CIM desempeña un importante rol en tender un puente entre los expertos del dominio y sus requisitos, y los expertos en el diseño y construcción de artefactos que juntos satisfacen los requisitos del dominio.

El modelo de plataforma independiente (PIM)

Se denomina Platform Independent Model (PIM) a una vista del sistema centrada en la operación del mismo que esconde los detalles necesarios para una determinada plataforma [16]. Un PIM muestra un determinado nivel de independencia de la plataforma de forma que sea posible utilizarlo con un número de plataformas diferentes de forma similar. Una técnica muy común para conseguir la independencia de plataforma es a través de tecnología neutral de una máquina virtual [3]. Estas máquinas virtuales son especificaciones de un procesador computacional, que pueden ser realizadas sobre múltiples plataformas computacionales, siendo típicamente su desarrollo lógico [18]. Las máquinas virtuales consiguen de este modo la independencia de de los programas que ejecutan con respecto a la plataforma de ejecución final (habitualmente un sistema operativo funcionando sobre una configuración hardware determinada). Es importante señalar que MDA propone un nivel de abstracción mayor, puesto que se buscará la independencia de la plataforma y dichas máquinas virtuales serán consideradas plataformas de ejecución. En la Figura 3.3 se recoge un escenario de utilización de MDA que ilustra diferentes niveles de abstracción y los artefactos de ejecución manejados en cada uno de ellos. En este escenario típico, una herramienta MDA recoge los diferentes PIM que especifican el sistema y los transforma en un lenguaje de alto nivel soportado por la plataforma de destino. En el caso de que la plataforma destino utilice una máquina virtual que no sea un intérprete puro [13], se requerirá un proceso de compilación que obtenga el código máquina soportado por la máquina virtual. Este es el caso de las plataformas Java y .NET. Es habitual que, por razones de eficiencia, dicha máquina virtual

sea compilada en cada sistema operativo como código máquina nativo, que será ejecutado finalmente por el hardware utilizado en cada caso.

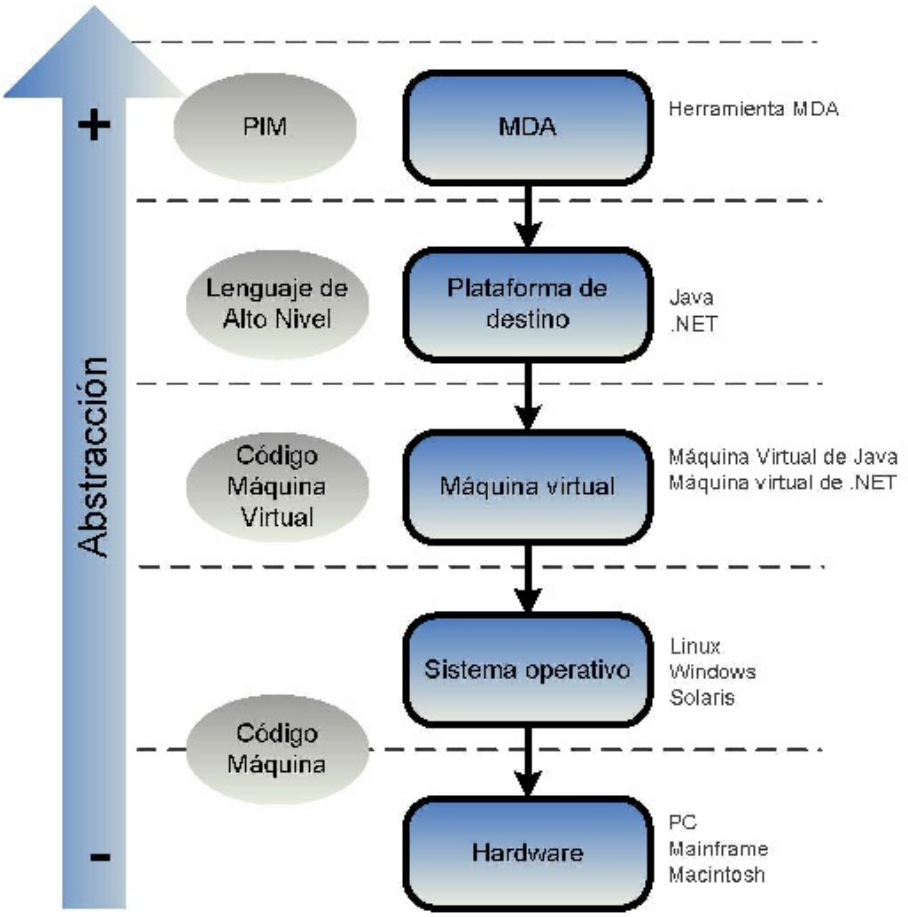


Figura 3.3: Ejemplo de niveles de abstracción e independencia con respecto a la plataforma MDA

El modelo de plataforma específica (PSM)

En Inglés Platform Specific Model (PSM) es la vista del sistema desde el punto de vista de la plataforma específica. Un PSM combina las especificaciones del PIM con los detalles que especifica cómo utiliza el sistema un tipo particular de plataforma. Un PSM es una vista del sistema desde el punto de vista de la plataforma específica de ejecución [16]. Combina los diferentes PIM que especifican el sistema con los detalles acerca de cómo utiliza un tipo concreto

de plataforma. La diferencia entre un modelo PSM y uno PIM es el conocimiento que incorpora acerca de la plataforma final de implementación. Un PIM no incorpora ninguna información acerca de la plataforma de destino. Por su parte, un PSM contiene toda la información necesaria para hacer posible su implementación sobre una plataforma concreta. Es decir, contendrá toda la información necesaria para generar el código de la aplicación a partir de él [14]. Existen autores que consideran el código fuente de la aplicación un modelo PIM [21]. No obstante, en la propuesta del OMG el código fuente de la aplicación se considera un resultado de una transformación sobre el PIM [MM03]. Esta implementación del programa en un lenguaje de programación determinado se conoce en ocasiones como Plattorm Specific Implementation (PSI). Aunque en la guía de referencia de MDA del OMG [16] se realiza una separación clara entre los dos tipos de modelos manejados en MDA: PIM y PSM, algunos autores consideran que la distinción no siempre es posible [12]. Esto es debido a que los modelos de las aplicaciones suelen contener información relacionada con la plataforma o tipo de plataforma sobre la que se ejecutarán. Por ejemplo, al especificar que un método de una clase es transaccional, puede considerarse que el modelo se convierte en específico a una plataforma tecnológica que soporte transacciones.

El modelo de plataforma

Suministra un conjunto de conceptos técnicos que representan los diferentes tipos de elementos que conforman la plataforma y los servidos ofrecidos por la misma. También especifica, para la utilización de un modelo especifico de plataforma, los conceptos que representan los diferentes tipos de elementos utilizados para especificar el uso de la plataforma por una aplicación. Un modelo de plataforma especifica además los requisitos de conexión y uso de las partes de la plataforma y la conexión de una aplicación a la plataforma. Un modelo de plataforma genérica puede establecer una especificación de un estilo particular de arquitectura.

Transformación

La transformación del modelo es el proceso de convertir un modelo en otro del mismo sistema. En MDA se contemplan dos tipos fundamentales de transformaciones (Figura 3.4):

- La transformación de un PIM en un PSM.
- La transformación de un PSM en el código fuente de la aplicación.

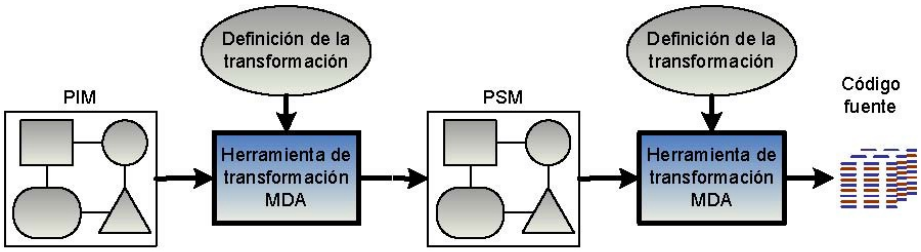


Figura 3.4: Transformaciones y herramientas de transformación

En la Figura 3.4 se representa esta doble transformación. Una herramienta de transformación MDA recogerá el modelo independiente de la plataforma PIM y lo transformará en un modelo específico a la plataforma PSM. Para realizar esta transformación, hará uso de una metainformación que especifique cómo realizar la transformación. Esta metainformación se denomina "definición de la transformación" [12] o "mapping rules" (reglas de mapeo) [16]. En general, suele utilizarse el término "mapping" (mapeo) para denominar "transformación". Además de las definiciones de transformación, es habitual etiquetar o marcar los modelos para guiar los procesos de transformación. Por ejemplo, para diferenciar datos persistentes de datos temporales. O para distinguir entre procesos remotos o locales. Esta metainformación se denomina genéricamente como ?marcas? (marks) [15]. Las marcas se utilizan tanto en los PIM como en los PSM. Las dos transformaciones pueden ser realizadas por la misma herramienta de transformación o por herramientas distintas. Es precisamente cómo se aborda la construcción de dichas herramientas de transformación, junto con las definiciones de cómo se realizan las transformaciones, uno de los aspectos claves que diferencian las diferentes líneas de investigación en torno al paradigma MDA. De hecho, en la Guía de Referencia de MDA [16], la información que especifica la transformación se representa con una caja blanca vacía (Figura 3.3).

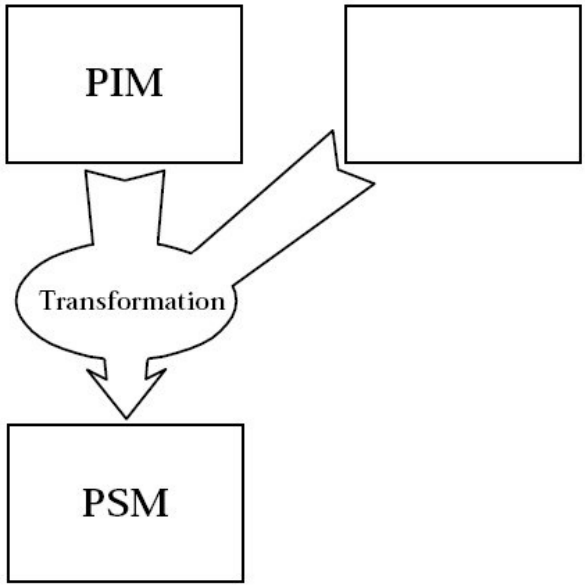


Figura 3.5: Definición de la información de la transformación

Servicios difundidos

Son los servicios disponibles en una amplia gama de plataformas. MDA suministra habitualmente, un modelo independiente de la plataforma de servicios difundidos. Facilitará el mapeado para la transformación de los modelos, dibujando estos servicios difundido PIMs, a los modelos de plataforma especifica utilizando los servicios suministrados por la plataforma en concreto.

Implementación

Una implementación es una especificación, que suministra toda la información necesaria para la construcción del sistema y ponerlo operativo.

3.3. XML Metadata Interchange

MOF XMI (XML Metadata Interchange) [4] es una especificación basada en XML para compartir metadatos MDA. Actualmente XMI es la base para conseguir la interoperabilidad entre herramientas MDA. Como ventaja principal de XMI se puede citar que se pueden almacenar todos los modelos basados en MOF utilizando un mismo esquema. Sin embargo, tiene inconvenientes tales como que su formato es muy difícil que sea leído por humanos o que muchas de las herramientas existentes que trabajan con XMI no se ajustan a las especificaciones de la recomendación.

3.4. Lenguaje de dominio específico (DSL)

DSL, o "Domain Specific Language" (Lenguaje Específico de Dominio), es un lenguaje de programación dedicado a un problema de dominio en particular, o una técnica de representación o resolución de problemas específica. [5].

Los DSL se crean específicamente para resolver problemas de un determinado dominio, con lo cual no están pensados para resolver problemas fuera de este dominio. Son lenguajes con objetivos muy específicos, tanto en diseño como en implementación. Puede ser un lenguaje de diagramación visual así como textual. Son por lo general pequeños, ofreciendo un conjunto limitado de notaciones y abstracciones. Son, por lo general, lenguajes declarativos., por lo que pueden verse como lenguajes de especificación, así como lenguajes de programación. [8].

Estos DSL no tienen funciones de bajo nivel (acceso al sistema de ficheros, control de procesos internos, etc.) y además no se compilan a código ejecutable. Tienen metas de diseño importantes que contrastan con aquellas de los lenguajes de propósito general:

- Tienen menos alcance.
- Son mucho más expresivos dentro de su dominio.

El uso de DSL trae consigo una serie de ventajas, como son permitir expresar soluciones usando los términos y el nivel de abstracción apropiado para el dominio de un problema concreto, así como una mejora de la calidad, productividad, mantenibilidad, portabilidad y reutilización de las aplicaciones. Si por el contrario se habla de desventajas, una que podemos remarcar es el hecho de encontrar, establecer y mantener el alcance adecuado de nuestro problema. Aunque, en definitiva, aporta muchas más ventajas que desventajas en el desarrollo de aplicaciones.

3.5. Windows Workflow Foundation (WWF)

Windows Workflow Foundation (WWF) [2] es un entorno de programación de Microsoft que permite construir un flujo de trabajo, el cual es un conjunto de unidades elementales llamadas actividades que están almacenadas como un modelo que describe un proceso real. Los flujos de trabajo proporcionan una manera de describir el orden de ejecución y las relaciones de dependencia entre las partes de trabajo de ejecución corta o prolongada. Este trabajo pasa a través del modelo desde el principio hasta el final y las actividades pueden ser ejecutadas por personas o por funciones de sistema.

3.5.1. Motor de tiempo de ejecución del flujo de trabajo

Las instancias de flujo de trabajo en ejecución se crean y se mantienen por medio de un motor de tiempo de ejecución en curso con el que el proceso de host interactúa a través de alguna de las clases siguientes:

- Una clase WorkflowInvoker, que invoca el flujo de trabajo como si se tratara de un método.

- Una clase WorkflowApplication para el control explícito de la ejecución de una única instancia de flujo de trabajo.

- Una clase WorkflowServiceHost para las interacciones basadas en mensajes en escenarios de varias instancias.

Cada una de estas clases ajusta el tiempo de ejecución de la actividad principal, representado como una clase ActivityInstance responsable de la ejecución de la actividad. Puede haber varios objetos ActivityInstance ejecutándose simultáneamente dentro de un dominio de aplicación.

Los tres objetos de interacción de host anteriores se crean a partir de un árbol de actividades al que se hace referencia como programa de flujo de trabajo. Con estos tipos o con un host personalizado que ajuste la clase ActivityInstance, los flujos de trabajo se pueden ejecutar dentro de cualquier proceso de Windows, incluidos las aplicaciones de consola, las aplicaciones basadas en formularios, los Servicios de Windows, los sitios web de ASP.NET y los servicios de Windows Communication Foundation (WCF).

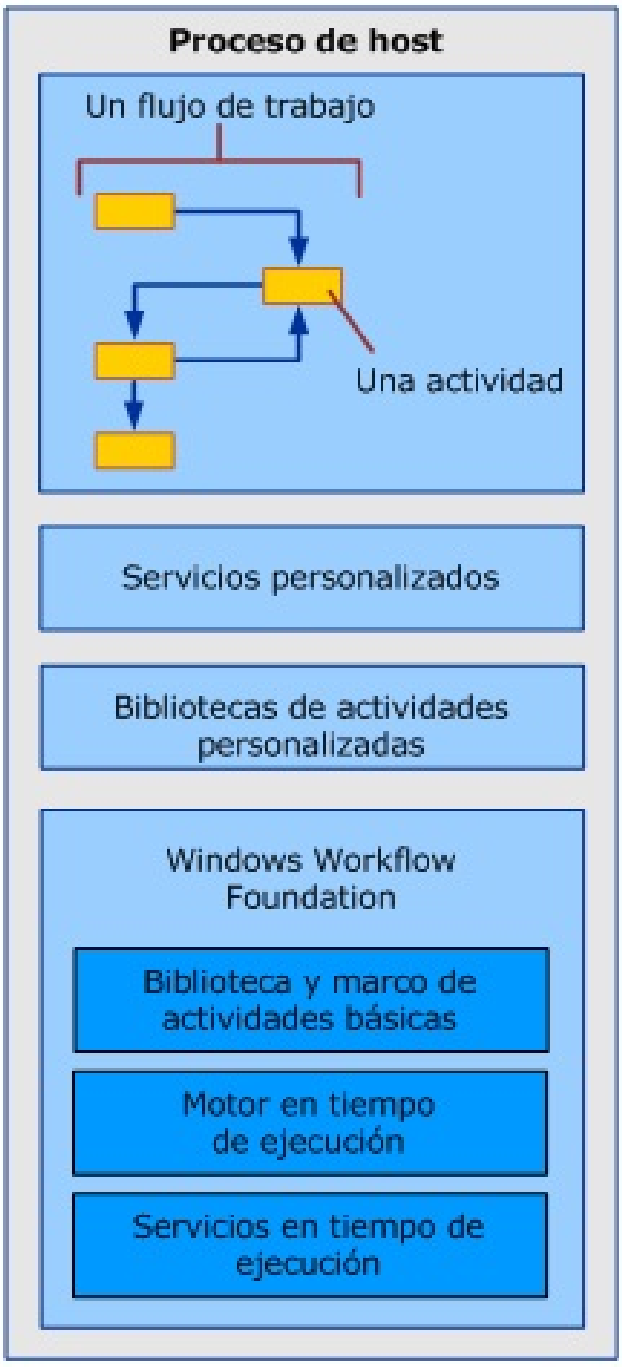


Figura 3.6: Proceso de host

3.5.2. Interacción entre los componentes de flujo de trabajo

El siguiente diagrama muestra cómo los componentes de flujo de trabajo interactúan unos con otros.

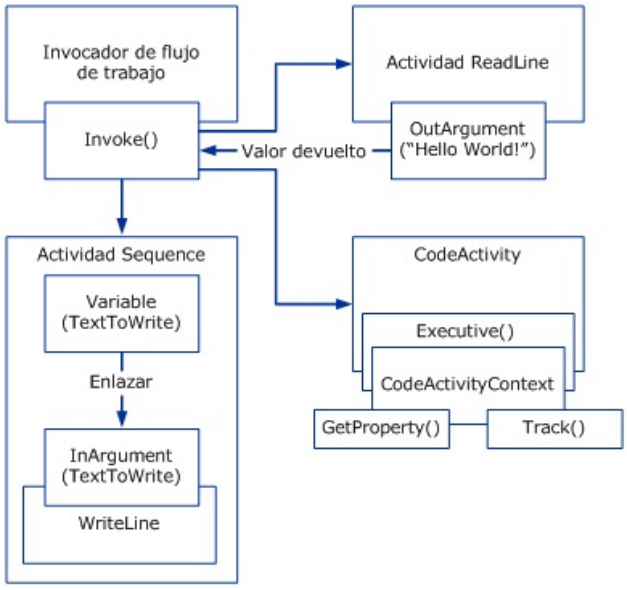


Figura 3.7: Flujos de trabajo interactuando unos con otros

En el diagrama anterior, el método Invoke de la clase WorkflowInvoker se usa para invocar varias instancias de flujo de trabajo. WorkflowInvoker se usa para los flujos de trabajo ligeros que no necesitan la administración del host. Los flujos de trabajo que necesitan la administración del host (como la reanudación de Bookmark) se deben ejecutar usando el método Run. Antes de invocar una nueva instancia de flujo de trabajo no es necesario esperar a que se complete la instancia de flujo de trabajo en ejecución. El motor de tiempo de ejecución permite ejecutar varias instancias de flujo de trabajo simultáneamente. Los flujos de trabajo invocados son los siguientes:

- Una actividad Sequence que contiene una actividad secundaria WriteLine. Un objeto Variable de la actividad principal está vinculado a un objeto InArgument de la actividad secundaria. Para obtener más información sobre variables, argumentos y enlaces, vea Variables y argumentos.
- Una actividad personalizada llamada ReadLine. Un argumento OutArgument de la actividad ReadLine se devuelve al método Invoke de llamada.
- Una actividad personalizada que se deriva de la clase abstracta CodeActivity. La clase CodeActivity puede tener acceso a las características de

tiempo de ejecución (como el seguimiento y las propiedades) usando CodeActivityContext, que está disponible como parámetro del método Execute. Para obtener más información sobre estas características de tiempo de ejecución, vea Seguimiento y traza del flujo de trabajo y Propiedades de ejecución del flujo de trabajo.

3.5.3. Entorno de trabajo WWF

El entorno de trabajo de Windows Workflow Foundation esta incluido dentro del entorno de desarrollo integrado en Visual Studio 2010. Su aspecto es el que se muestra en la siguiente imagen.

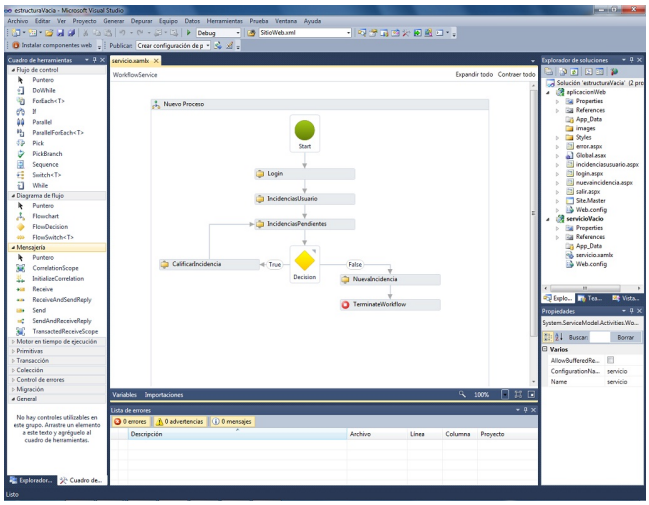


Figura 3.8: Entorno de trabajo de Windows Workflow Foundation

De este espacio es posible diseñar las actividades necesarias para construir los flujos de trabajo, así como definir el orden de su ejecución. La sección izquierda es la paleta de herramientas que contiene las entidades que pueden ser utilizadas en el modelado de los procesos; esta área es independiente en el entorno de trabajo de WWF. La sección central está dividida en dos: el área de mayor tamaño se corresponde con la zona de modelado del proceso mientras que la zona inferior contiene el área de notificación de errores y avisos; a diferencia del caso anterior, la utilización de las entidades de la paleta de herramientas se hace a través de la selección de la entidad en la paleta y el arrastre de la misma a la zona de diseño. La zona de la derecha se encuentra distribuida en dos secciones: la sección superior muestra la estructura de ficheros del proyecto o solución que está siendo utilizado y la parte inferior es la zona donde se muestran las propiedades de los distintos elementos; a través de la utilización de esta sección inferior es posible realizar cambios en el modelo que se está diseñando sin necesidad de tocar el fichero xamlx asociado al proceso.

3.6. HTML5

HTML5 será el nuevo estándar para el HTML. La versión anterior de HTML, HTML 4.01, se produjo en 1999. La web ha cambiado mucho desde entonces.

HTML5 es todavía un trabajo en progreso. Sin embargo, los principales navegadores apoyan muchos de los elementos nuevos y API de HTML5.

HTML5 es una colaboración entre el Consorcio de la World Wide Web (W3C) y la web de hipertexto Aplicación de la Tecnología del Grupo de Trabajo (WHATWG).

WHATWG estaba trabajando con los formularios web y aplicaciones, y el W3C está trabajando con XHTML 2.0. En 2006, decidieron cooperar y crear una nueva versión de HTML. Algunas reglas para HTML5 se establecieron:

- Las nuevas características debe basarse en HTML, CSS, DOM y JavaScript
- Reducir la necesidad de plugins externos (como Flash)
- Mejor manejo de errores
- Más marcado para reemplazar secuencias de comandos
- HTML5 debe ser independiente del dispositivo
- El proceso de desarrollo debe ser visible para el público

En HTML5 no es sólo una declaración, y es muy simple <Doctype!>

```
<!DOCTYPE html>
```

Figura 3.9: HTML5

A continuación se muestra un documento HTML5 simple, con el mínimo de etiquetas requeridas:

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>

<body>
The content of the document......
</body>

</html>
```

Figura 3.10: HTML5 Simple

Algunas de las novedades más interesantes en HTML5:

- El elemento <canvas> para dibujo en 2D

- Los elementos <video> y <audio> para la reproducción de medios
- Apoyo para el almacenamiento local
- Nuevos contenidos de elementos específicos, como <article>, <footer> y <header> y <nav>, <section>
- Los nuevos controles de formulario, como el calendario, fecha, hora, correo electrónico, URL, búsqueda

HTML5 aún no es una norma oficial, y no los navegadores tienen soporte HTML5 completo. Sin embargo, todos los principales navegadores (Safari, Chrome, Firefox, Opera, Internet Explorer) continúan agregando nuevas características HTML5 a sus últimas versiones.

3.7. Dispositivos móviles

Figura 3.11: Dispositivos Móviles

Existen multitud de tipos de dispositivos móviles, desde teléfonos móviles, smartphones y tablets que ofrecen nuevas posibilidades que hasta ahora estaban asociadas a ordenadores de mesa o a portátiles. La comodidad de estos dispositivos y la potencia que pueden llegar a tener, son dos de los secretos que permite que estos aparatos hayan tenido tanto éxito.

Existen varias plataformas que compiten entre ellas para conseguir tener más usuarios, en el siguiente enlace (http://www.wayerless.com/2011/11/todos-contra-todos-ios-vs-android-vs-windows-phone-vs-blackberry/) está disponible una comparativa entre los 4 principales sistemas operativos móviles más usados del mundo.

A pesar de existir esta rivalidad entre plataformas y sobre todo entre el desarrollo de software y los usuarios que tiene cada una, HTML5 es un estándar que es compatible en los dispositivos móviles y por ello desarrollar software en esta tecnología permitirá el acceso de cualquier usuario a los sitios web generados.

3.8. Wix

Figura 3.12: Wix: Herramienta gráfica para diseñar una web

Permite crear páginas web dinámicas con material enriquecido como vídeos, imágenes, animaciones, enlaces, etc. Para diseñar una página es necesario elegir la categoría a la que pertenece el ámbito de la página que se va a desarrollar, así como una subcategoría que la defina un poco más. Esta clasificación está pensada y enfocada al aspecto final de la página, no es lo mismo un buffet de abogados que una página de un bar. Es una herramienta gráfica, con lo cual, está pensada tanto para personas principiantes en el mundo de la informática, como para profesionales que ya tengan experiencia en el tema. Dispone de un editor gráfico en el que puedes añadir elementos a partir de una paleta de opciones y una serie de herramientas que se muestran juntas en la siguiente imagen para que se vea rápidamente todo lo que ofrece esta herramienta.

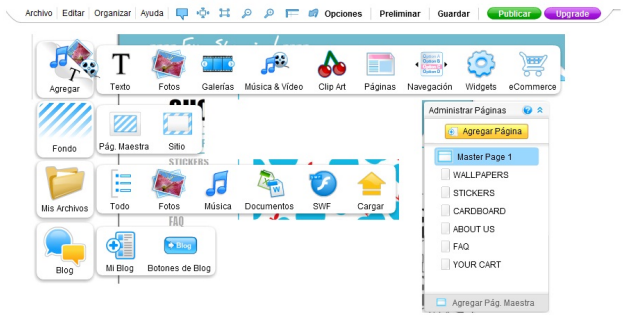


Figura 3.13: Wix: Opciones del editor para diseñar una web

Todos los elementos que aparecen en la página son objetos. Wix ofrece un tratamiento común para ellos, se pueden escalar rotar y también se puede cambiar su posición con respecto a otros objetos existentes en la página. También se puede añadir una galería de fotos, efectos a los objetos de la página, animaciones, comportamientos,etc. Wix permite publicar material mediante distintas páginas. Las páginas pueden ser:

- **Maestras:** se sustituyen unas a otras.
- **Convencionales:** dentro de una página maestra, podemos insertar distintas páginas. Cada página tiene su propio contenido.

Cuando usamos distintas páginas maestras hay que poner controles de navegación en cada una de ellas, que pueden ser:

- **Menus:** muestran la lista de páginas maestras para que el usuario escoja a cuál ir.
- **Botones:** al hacer click sobre ellos, se irá a una página maestra determinada.
- **Controladores:** permiten navegar entre páginas como si fuera una galería.

Las páginas pueden contener código HTML, mediante un widget HTML.

3.9. WordPress

Figura 3.14: Wordpress: Crea tu blog

WordPress es un sistema de gestión de contenidos (CMS) orientado específicamente a blogs creado a partir de la antigua aplicación. Su desarrollador principal, Matt Mullenweg se enfocó desde un principio en crear una aplicación web libre bajo licencia GPL, lo cual le permitió desde un principio crear una gran comunidad, fomentando su desarrollo y crecimiento.

Actualmente WordPress es el sistema más utilizado al rededor del mundo, no solamente por la facilidad de su uso, sino también porque tanto el software como sus herramientas son casi en su totalidad gratuitas.

Dentro de las características principales de WordPress, encontramos: fácil instalación, actualización, personalisación y uso en general, posee herramientas de comunicación entre blogs, sistema de enlaces, calendario, fácil integración con el foro bbPress y Vanilla de Lussumo factible, Importación desde Movable Type y Typepad, Textpattern, Greymatter, Blogger, Dotclear, Livejournal, Blogware y desde cualquier RSS, entre otras.

Actualmente WordPress es una de las mejores aplicaciones para administrar blogs, no solamente debido a las características que nombramos anteriormente como su gran facilidad de uso y su licencia GPL, entre otras, sino principalmente debido a la gran cantidad de plugins y themes que existen, que permiten modificar el aspecto y funcionalidad de la página al 100

Otra gran ventaja es que WordPress está optimizado para que los buscadores lo indexen mejor que a otros CMS, por este hecho está siendo adoptado masivamente por muchos webmasters hoy por hoy.

Por otra parte, permite que sea administrado por múltiples autores, permite comentarios que pueden ser publicados en el sitio, borrados o marcados como spam, Widgets para los Themes, admite pluggins, gestión y distribución de enlaces, y además permite ordenar artículos y páginas en categorías, subcategorías y etiquetas, entre otras prestaciones.

Si deseas lanzar webs rápidamente, con poco trabajo y solo tienes tiempo de trabajar en el material (información) y no en el diseño o funcionalidades, solo basta escribir el texto, descargar unos cuantos plugins, un lindo theme y ya tienes tu web funcional.

Como en toda aplicación web, las actualizaciones son siempre importantes, no solo para tener la última versión del producto, tener siempre tus plugins y themes compatibles, sino también para poder evitar los baches de seguridad y vulnerabilidades comunes que se ven hoy en día en las aplicaciones webs, de este modo evitamos problemas de hackeo o pérdida de información en nuestros sitios.

3.10. Trabajo Relacionado

La metodología MDA ofrece un soporte que permite definir el meta-modelo que definirá el resto de modelos, en [11] se utiliza la metodología MDA para generar artefactos incrementalmente a partir de modelos, de forma que se optimicen los recursos computacionales, se minimice el impacto sobre aplicaciones que ya podrían estar en producción y se permita la realización de trazabilidad de la evolución de los sistemas.

El ámbito de aplicación de MDA es muy variado en [10] se muestran las ventajas de TALISMAN MDE [19], que desarrolla un sistema que permite generar y ser usado para controlar la trazabilidad alimentaria. En el mundo del e-learning, MDE y MDA junto con DSL ofrecen el soporte idóneo para que en trabajos como este [17] se pueda generar un meta-modelo basado en el estudio de las plataformas Learning Management System (LMS) para desarrollar una herramienta que genere módulos que puedan ser desplegados en tres gestores de

contenido o CMS diferentes, como pueden ser Moodle, Claroline o Atutor. La tecnología MDA también resolvió el trabajo de definir una notación de modelado denominada BPMN muy simple (BPMN MUSIM) que permita realizar el modelado de los procesos de negocio a través de un conjunto mínimo de símbolos, intentando reducir la complejidad de uso existente en BPMN[23].

En los trabajos anteriores la tecnología MDE y MDA resolvieron los problemas y agilizaron el proceso de desarrollo de las propuestas. Por eso, y dada la eficacia y el éxito de estas tecnologías, se han utilizado en esta propuesta para desarrollar una herramienta que permita definir una web académica desde un entorno gráfico.

Capítulo 4

Trabajo Propuesto

Se ha construido un meta-modelo mediante el cual se definirá un modelo que permitirá generar sitios web académicos. Este meta-modelo será definido mediante una herramienta gráfica, llamada Webdesing, que generará ficheros XML en los que estarán descritas las reglas del modelo. A través de un proceso de transformación en el que se interpretarán los ficheros XML y mediante plantillas de transformación (http://msdn.microsoft.com/es-es/library/bb126445.aspx) se generará un sitio web académico. Con la utilización de las plantillas de transformación se consigue desarrollar un sitio web en varios idiomas de forma fácil y rápida. Como se muestra en la imagen, a través de una herramienta gráfica y sin conocimientos de programación es posible generar sitios web académicos.

Figura 4.1: Propuesta

4.1. Meta-modelo

Mediante la herramienta gráfica Webdesing se va a construir el meta-modelo. Para ello, Webdesing generará archivos XML en los que se van a describir las reglas del meta-modelo. Existen varios modelos dentro del meta-modelo:

- Modelo de enumeración: en este modelo se enumerarán todos los elementos que va a tener el sitio web final.

```
<sitioWeb titulo="Master de ingeniería Web">
<Site>
<Inicio/>
<Programa/>
<Introduccion/>
<Preinscripcion/>
<ResolverPreinscripcion/>
<Contacto/>
</Site>
</sitioWeb>
```

Figura 4.2: Meta-modelo

- Modelo de página: en este modelo se definirá el contenido que tendrán las diferentes páginas que formarán el sitio web final.

```
<titulo3>Dos especialidades: Profesional e Investigadora </titulo3>
<texto>
El Máster se oferta de carácter mixto, lo
que significa que el alumno puede optar por una u otra
especialización.
Durante su segundo curso, el alumno decidirá si quiere obtener
un título de postgrado que le capacite para ejercer la
profesión,  según la nueva normativa europea,
o si por el contrario desea obtener un título equivalente al
actual doctorado de cara a la realización de
su tesis doctoral.
<enlace direccion="programa/">Más sobre especialidades</enlace>
</texto>
</bloque>
<bloque col="12">
<titulo3>Organizadores</titulo3>
<imagen ruta="Img/empresas/uniovi.gif"/><texto>Universidad de Oviedo</texto>
<imagen ruta="Img/empresas/euitio.gif"/><texto>Escuela de Ingeniería Informática</texto>
</bloque>
```

Figura 4.3: Modelo de Página

- Modelo de formulario: modelo en el que se definirán los contenidos que va a tener un formulario.

```
<Contacto>
    <Campos>
      <campo Id="Nombre" Text="Nombre" validacion="Introduzca su nombre, por favor!"/>
      <campo Id="Email" Text="E-Mail" validacion="Introduzca su e-mail, por favor!" formato="Formato de e-mail inválido"/>
      <campo Id="Asunto" Text="Asunto" validacion="Introduzca el asunto, por favor!"/>
      <campo Id="Mensaje" Text="Mensaje" validacion="Introduzca el mensaje, por favor!"/>
    </Campos>
    <botonEnviar Id="Enviar" Text="Enviar"/>
  </Contacto>
```

Figura 4.4: Modelo de Formulario

Estos archivos XML anteriormente descritos se van a generar a través de la herramienta gráfica Webdesign que es una herramienta de modelado que ofrece la posibilidad de generar el tipo de archivo correspondiente en cada caso a través de las características aportadas al proyecto por Windows Workflow Foundation. Esta herramienta ha sido desarrollada aprovechando la posibilidad de realizar un rehosting del entorno de construcción de los diagramas de flujo existentes en Visual Studio 2010.

4.1.1. Características de la Herramienta

La característica más destacable de esta herramienta es su sencillez debido a la intención de ponerla a disposición de usuarios con pocos conocimientos informáticos y de modelado. Así, se ha optado por construir un interfaz con tres zonas delimitadas y que ofrezca las acciones más comunes de cualquier software.

Otra característica destacable de este software de modelado es el alto nivel de adaptación y la sencillez con la que se logra ajustar su contenido a las necesidades existentes. Los mecanismos ofrecidos por la Windows Workflow Foundation para construir este tipo de herramientas permiten configurar, a través de sencillos cambios en las líneas de código de la aplicación, las entidades que aparecerán en el interfaz gráfico a disposición de los usuarios. Esto supone una ventaja en entornos donde se quiera disponer de distintas versiones de la herramienta de modelado para satisfacer las necesidades de un grupo de expertos para un dominio concreto.

4.1.2. Estructura de la Herramienta

La figura que se incluye a continuación muestra la estructura general de la herramienta de modelado. Tal y como se puede observar el interfaz gráfico está dividido en tres zonas: paleta de opciones, la zona de diseño y el área de propiedades. Además, se incluye un menú superior en el que están presentes los siguientes contenidos: el menú Archivo, que permite acceder a las opciones de uso de la herramienta; el menú Ayuda, que muestra unas indicaciones sobre la forma de uso de la herramienta; y el menú Acerca de, que establece la versión de la herramienta que se está utilizando y su fecha de publicación.

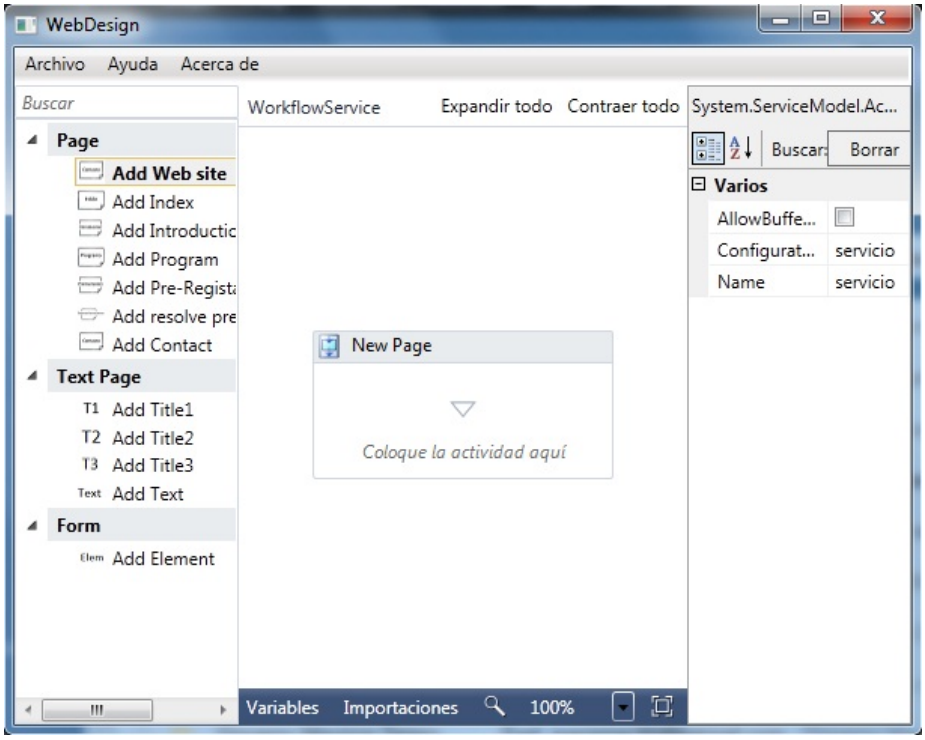


Figura 4.5: WebDesing

Paleta de opciones

La paleta de opciones es la barra lateral situada más a la izquierda en el interfaz gráfico. Esta área contiene todos aquellos elementos de los que disponen los usuarios del software para realizar los modelados de sus páginas web. En concreto se pueden observar tres secciones dentro de esta zona: la seccion Page, Text Page y Form que contienen las opciones necesarias para modelar paginas web de ese tipo.

Zona de diseño

La zona de diseño es la parte central de esta aplicación de modelado y en ella se encuentra el modelo del proceso que irán componiendo los usuarios cuando utilicen la herramienta. Dicha zona está compuesta por una secuencia en la que se irán añadiendo los diferentes elementos que van a conformar la página web.

Área de Propiedades

La zona situada en el extremo derecho del interfaz gráfico de esta herramienta es la zona donde se muestran las propiedades de la entidad o del elemento que esté seleccionado en el modelo del diseño que se esté realizando. Las propiedades mostradas comprenden tanto el contenido de la entidad seleccionada así como posibles variables añadidas necesarias en cada elemento en particular.

4.1.3. Programación de la herramienta

Después de haber expuesto la estructura del interfaz de WebDesing es necesario establecer cómo se construye una herramienta de este tipo utilizando Visual Studio 2010.

Definición de la Estructura

El primer paso a dar es la configuración de los componentes que aparecerán en la pantalla de la aplicación, de tal forma que será necesario dividir la pantalla en las secciones que vaya a contener. Para ello, Visual Studio 2010 ofrece la posibilidad de definir la apariencia de la aplicación a través de un fichero xaml.

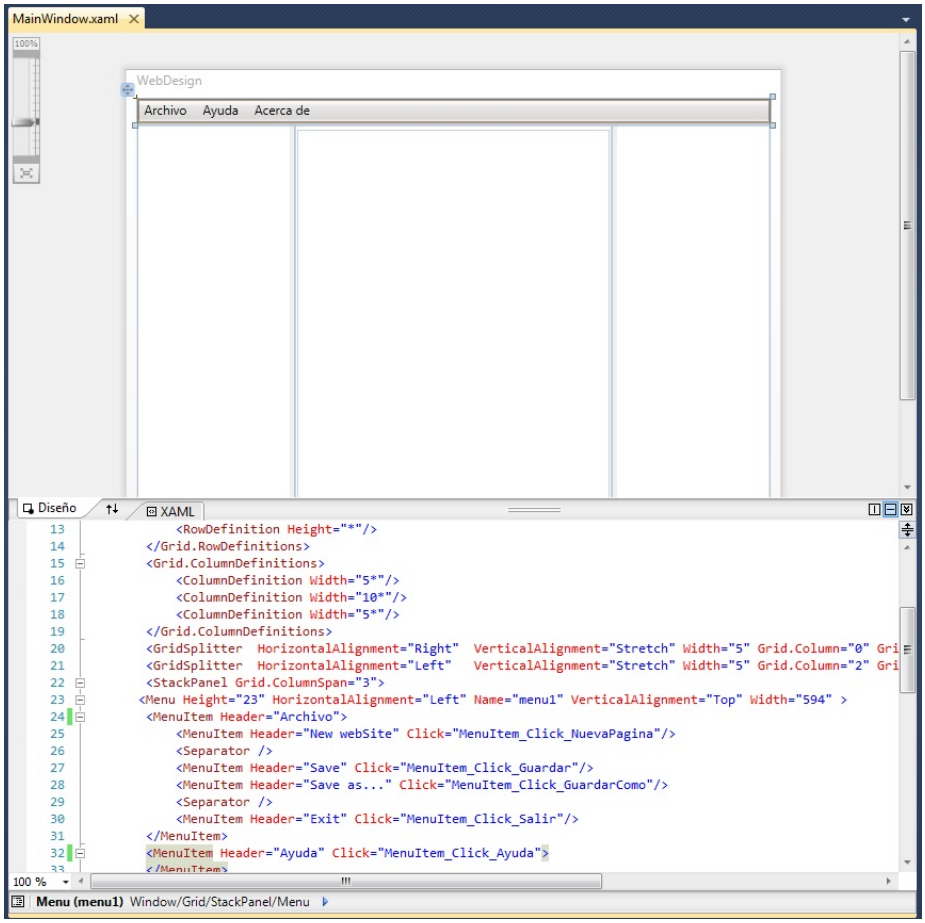


Figura 4.6: Entorno de definición de la interfaz

Según se observa en la captura anterior, el fichero xaml que representa a la aplicación es un fichero de configuración en el que se indica, entre otros muchos aspectos, la disposición de filas y columnas de la pantalla de la herramienta. En este caso la pantalla ha sido dividida en dos filas: una para albergar el menú que permite tener acceso a las acciones que se pueden realizar y otra para albergar la paleta de opciones, modelado y propiedades. En la imagen anterior también se puede observar que además de definir el número de filas y columnas también se puede establecer el ancho de las mismas; en el caso de WebDesing se ha optado por ofrecer el doble de ancho a la zona de modelado ya que se la considera la parte más importante de la aplicación.

Definición del Menú

Una vez definidas cada una de las filas y columnas de nuestro interfaz es necesario establecer sus contenidos. En primer lugar se establece el contenido de la barra del menú, donde se establece que existirán varias acciones disponibles en uno de sus componentes y en los otros dos se configura la aparición de unos cuadros de diálogo con los mensajes pertinentes. Como se muestra en la captura incluida a continuación.

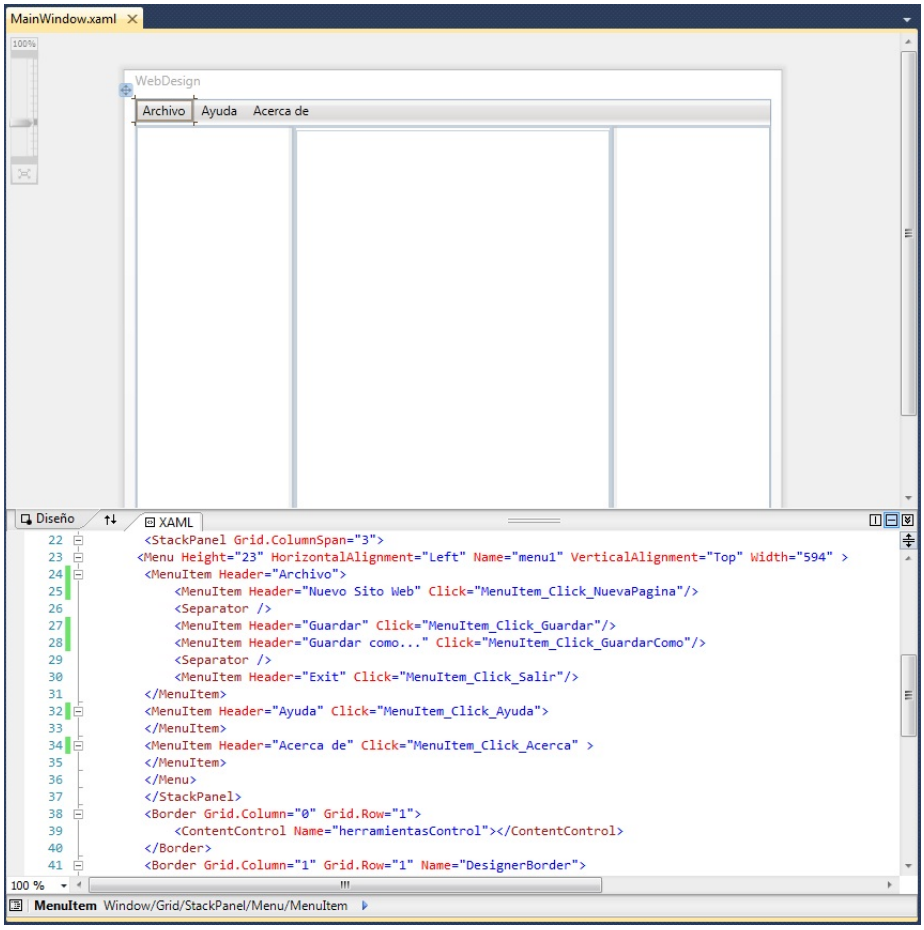


Figura 4.7: Definición de elementos del Menú

La definición de los elementos del menú, tal y como se muestra en la imagen superior, conlleva la existencia de una serie de métodos en el fichero de código asociado a esta definición del interfaz. Así, cada vez que se selecciona un elemento del menú, se produce una llamada al método correspondiente, que ejecuta la

acción asociada al elemento seleccionado. Se puede observar también como se utilizan unos separadores para diferenciar las opciones del menú Archivo, de tal forma que aparezcan separadas de una manera similar a la que aparecen en las aplicaciones informáticas comúnmente utilizadas.

Definición de las Zonas Principales

la zona central del interfaz se encuentra dividida en tres columnas que darán soporte a cada una de las zonas descritas en el apartado de definición de la estructura de la herramienta. Así, al igual que en el caso de los elementos de menú, será necesario definir cuáles serán los contenidos de cada una de estas zonas.

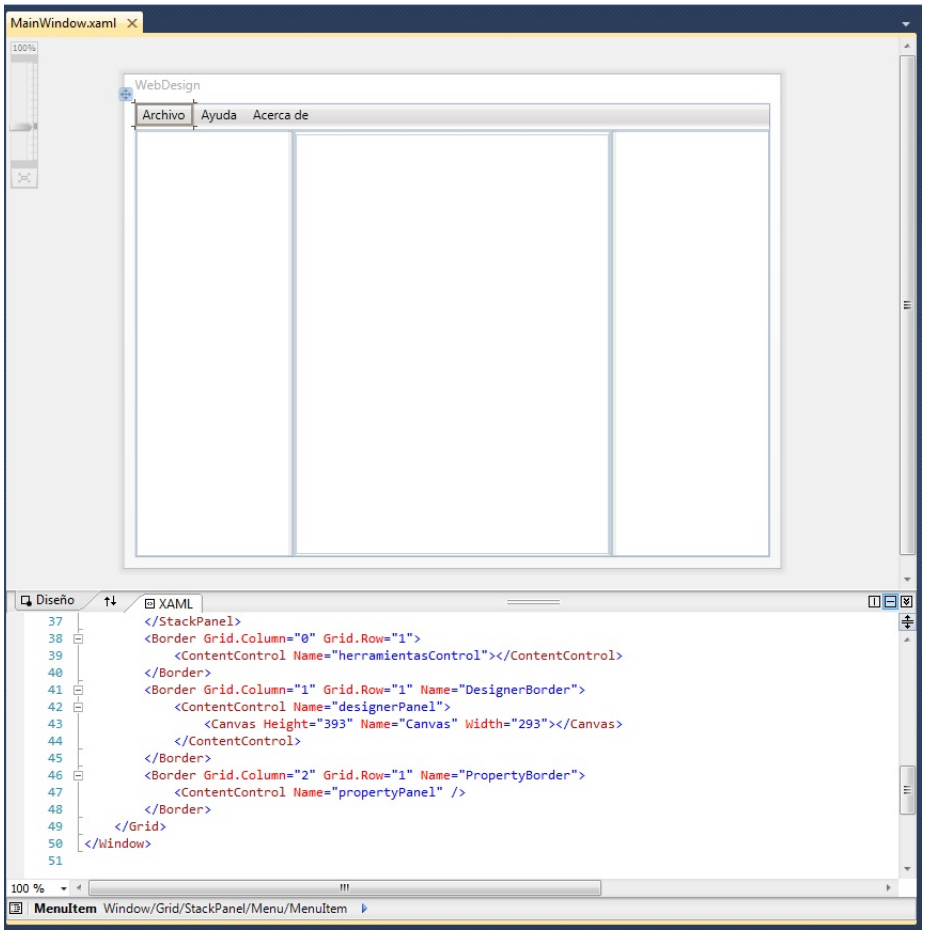


Figura 4.8: Definición de las Zonas Principales

La captura superior muestra la definición de los contenidos de control que se asocian a cada una de las zonas. En este caso los contenidos se cargarán en el archivo de código asociado ya que es necesario declarar la construcción de objetos y hacer otras tareas que no es posible realizar desde este fichero de configuración.

Archivo de Código Asociado

La definición del interfaz, que se realiza en el fichero xaml mostrado en varias capturas anteriores, se ve complementada con la existencia de un fichero de código en el que se establecen los detalles de ejecución de las distintas características de la herramienta; por ejemplo, todo lo relativo a la ejecución de las acciones existentes en el menú de la aplicación.

4.1.4. Ejemplo de Utilización de WebDesing

El comienzo de la utilización de la herramienta muestra la pantalla que se visualiza en la imagen que se incluye a continuación. Tal y como se observa en dicha figura, la zona de diseño de la herramienta aparece vacía al comienzo por lo que será necesario seleccionar la opción Nueva Página dentro del menú Archivo de la herramienta para iniciar un nuevo modelado de la página web.

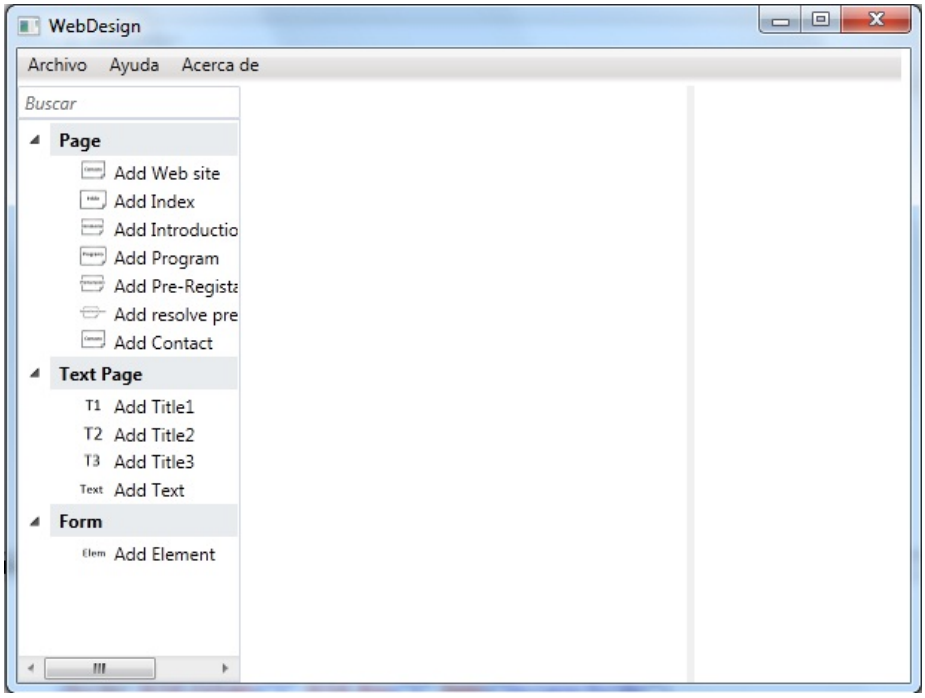


Figura 4.9: Herramienta de diseño web

Una vez que el usuario ha seleccionado la opción de iniciar una nueva página, la zona de diseño de la herramienta se carga con la estructura de la secuencia que permitirá realizar el modelado. Esta situación se ilustra en la figura que sigue a este párrafo, donde se observa que la zona central de la herramienta muestra una actividad secuencia dentro de la cual se pueden ir añadiendo los elementos que estime oportuno el usuario según el diseño deseado.

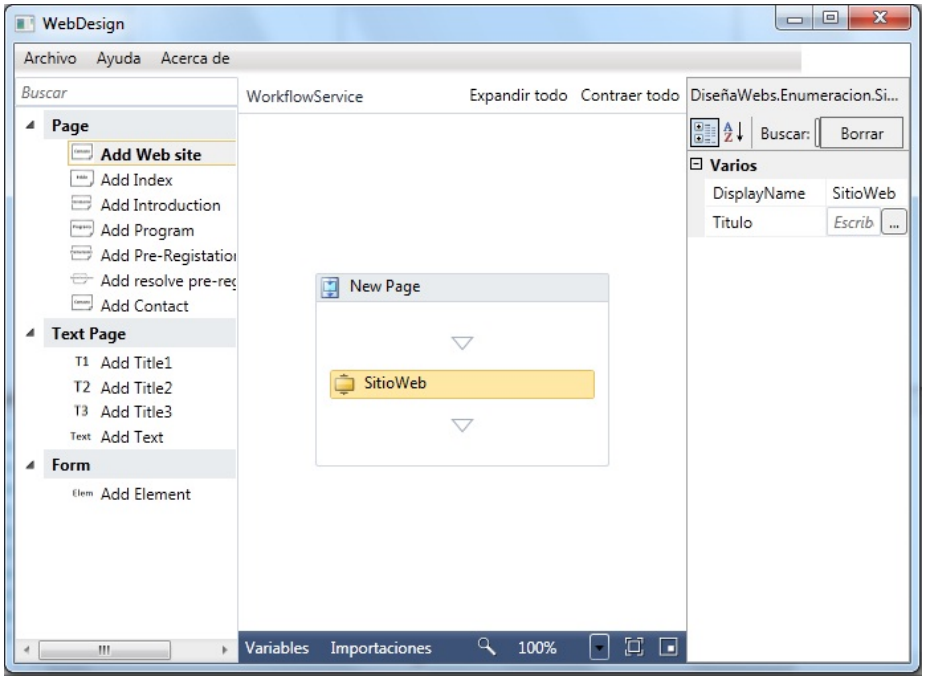


Figura 4.10: Herramienta de diseño web con un ejemplo

Al término del proceso de modelado el usuario debe de acceder de nuevo al menú Archivo para poder almacenar su diseño de la página. De esta forma se procederá a almacenar el diseño en fichero XML que será utilizado por el generador que se presentará en el siguiente apartado para crear los artefactos necesarios para poder ejecutar el proceso: un servicio de workflow y una aplicación web.

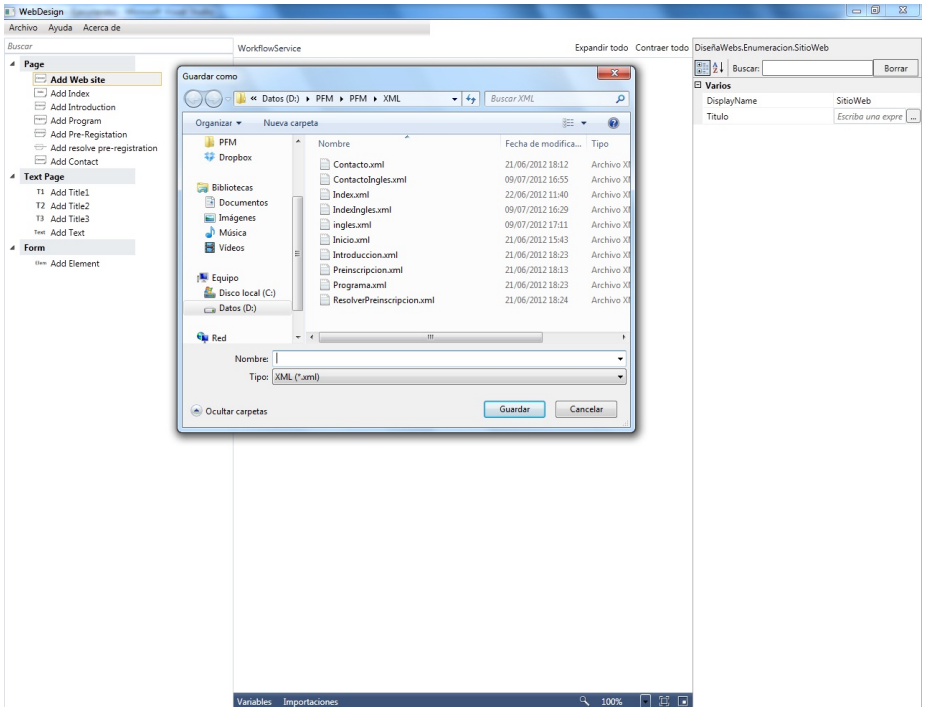


Figura 4.11: Herramienta de diseño web guardando el ejemplo

4.2. Generación de Código a partir de Modelos de Páginas Web

Una vez modeladas las distintas Páginas web que van a formar el sitio web académico final es necesario procesar los ficheros XML que genera Webdesing. Para facilitar y simplificar la configuración del entorno para la ejecución de los procesos modelados con WebDesing se ha desarrollado un pequeño prototipo que crea una solución de Visual Studio 2010 en la que se incluye el contenido que se eligió a partir del modelo descrito mediante la herramienta gráfica.

4.2.1. Estructura de la Herramienta

Web Generator es una aplicación de escritorio desarrollada en C dentro del entorno Visual Studio 2010. Esta herramienta ha sido desarrollada con el fin de ofrecer a los usuarios una forma sencilla de utilizar los modelados realizados por ellos para crear el sitio web académico personalizado.

Esta aplicación cuenta con un interfaz gráfico sencillo en el que se pide al usuario que complete dos campos de información necesarios para crear la

estructura del futuro sitio web académico: la ruta de la carpeta que contiene los archivos de proceso generados con WebDesing y la ruta donde desea que se genere la solución de Visual Studio 2010 que contendrá el sitio Web Académico completo.

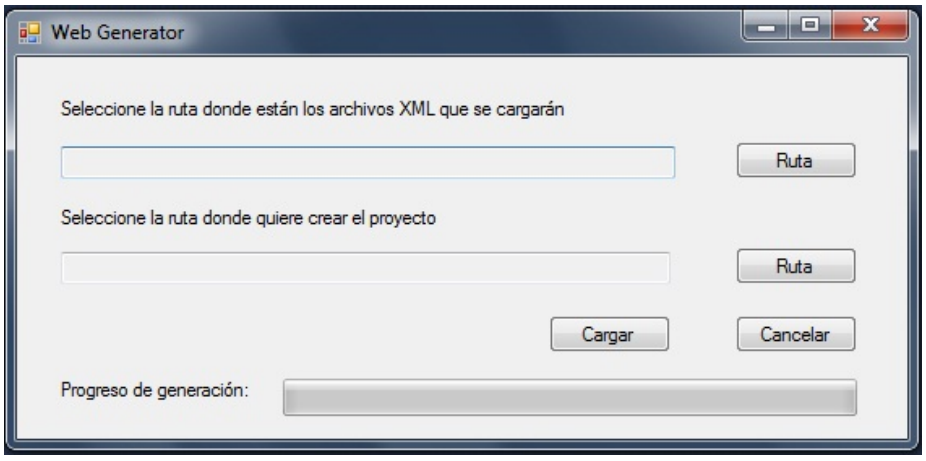


Figura 4.12: Generador Web

4.2.2. Programación de la Herramienta

Web Generator realiza tres acciones para generar la estructura que ofrece como resultado a partir de los ficheros XML que se le introduce como uno de sus parámetros. Estas tres acciones son las siguientes: copiar una estructura de proyecto vacío sobre el que se trabaja para generar el proyecto asociado a los ficheros XML, introducir los ficheros XML y añadir a la estructura los formularios web necesarios para conseguir el sitio web académico completo y generar un fichero XML global.

Copia de la Estructura Vacía

El primer paso que realiza Web Generator es utilizar la ruta de generación que el usuario le indica para situar en ella una estructura vacía de Visual Studio 2010 que se irá configurando para cumplir con las necesidades del modelo de proceso. Esta estructura vacía ha sido configurada a mano previamente de tal forma que incluye dos proyectos de la plataforma .NET: uno para una aplicación web y otro para un servicio de Workflow.

Para realizar esta tarea se ha programado un método que, a partir de la ruta indicada como base para la estructura vacía copia todo el contenido de esta en la ruta especificada por el usuario como destino de la solución en Web Generator. La imagen incluida a continuación muestra el código del método en cuestión, pudiendo observarse como se realiza la navegación por todos los

directorios y ficheros de la estructura vacía para realizar la copia de los mismos en la ruta escogida por el usuario como destino.

```
private bool copiarEstructura(string source, string destination)
{
    if (!Directory.Exists(destination))
        Directory.CreateDirectory(destination);
    string[] files = Directory.GetFiles(source);
    foreach (string file in files)
    {
        string name = Path.GetFileName(file);
        string dest = Path.Combine(destination, name);
        File.Copy(file, dest);
    }
    string[] folders = Directory.GetDirectories(source);
    foreach (string folder in folders)
    {
        string name = Path.GetFileName(folder);
        string dest = Path.Combine(destination, name);
        copiarEstructura(folder, dest);
    }
    return true;
}
```

Figura 4.13: Contenido del Método de Copia de la Estructura Vacía

Procesar ficheros XML

La segunda acción que realiza Web Generator es la adicción de los formularios necesarios a la estructura de la aplicación web generada. A través de un análisis de los elementos que han sido incluidos en el modelo del indice del sitio web que es uno de los parámetros de entrada se determina qué formularios serán necesarios para componer el sitio web académico generado. El contenido del fichero de modelado del indice es un fichero xml, tal y como se muestra en la captura que se incluye a continuación.

```
<sitioWeb titulo="Master de ingeniería Web">
<Site>
<Inicio/>
<Programa/>
<Introduccion/>
<Preinscripcion/>
<ResolverPreinscripcion/>
<Contacto/>
</Site>
</sitioWeb>
```

Figura 4.14: Contenido del fichero index

A partir de esta característica y teniendo en cuenta el deseo de construir un generador que precise del menor mantenimiento posible, se ha decidido utilizar el lector de xml incluido en la plataforma .NET como el punto de partida para la realización de la inspección del fichero. Así,el primer paso necesario para la realización de este análisis es convertir el fichero en un archivo xml estándar; para conseguir este objetivo sin necesidad de realizar cambios en el fichero original, se ha optado por traspasar el contenido del archivo a una cadena de texto que comienza con la cabecera xml estándar. Una vez que el fichero es accesible a través del lector de xml de .Net es necesario comenzar el análisis del mismo. En el caso de un fichero xml como el que nos ocupa, el tipo de nodos xml que nos interesan para la inspección son los nodos elementos del fichero sin embargo, no nos interesan todos los elementos que pueda haber en el fichero sino solamente aquellos que se correspondan con los ficheros que se hayan elegido. Para lograr la distinción de estos elementos frente a los demás se realizará un recorrido por todos los elementos del fichero y se inspeccionarán en detalle aquellos cuyo nombre se corresponda con el utilizado por los archivos elegidos. Para realizar esta operación se utiliza una estructura de tipo switch en la que se busca los elementos xml que coincidan con las especificaciones citadas anteriormente, tal y como queda reflejado en la captura siguiente.

```
private bool copiarInterfaces()
{
    String contenido = tratamientoFichero(rutaFichero.Text + "\\Index.xml");
    using (XmlReader reader = XmlReader.Create(new StringReader(contenido)))
    {
        while (reader.Read())
        {
            switch (reader.NodeType)
            {
                case XmlNodeType.Element:
                    if (reader.Name.StartsWith("Inicio") || reader.Name.StartsWith("Introduccion")
                        || reader.Name.StartsWith("Programa") || reader.Name.StartsWith("Preinscripcion")
                        || reader.Name.StartsWith("Contacto") || reader.Name.StartsWith("ResolverPreinscripcion"))
                    {
                        String nombreFormulario=reader.Name;
                        FileInfo formulario;
                        formulario = new FileInfo(rutaFicheros + nombreFormulario);
                        if (!formulario.Exists)
                        {
                            //copiar la estructura de cada fichero elegido
                            File.Copy(rutaFicheros + nombreFormulario + ".tt", this.nombreRuta.Text + carpetaFormularios + nombreFormulario + ".tt");
                            File.Copy(rutaFicheros + nombreFormulario + "C.tt", this.nombreRuta.Text + carpetaFormularios + nombreFormulario + "C.tt");
                            File.Copy(rutaFicheros + nombreFormulario + ".aspx", this.nombreRuta.Text + carpetaFormularios + nombreFormulario + ".aspx");
                            File.Copy(rutaFicheros + nombreFormulario + "C.aspx.cs", this.nombreRuta.Text + carpetaFormularios + nombreFormulario + "C.aspx.cs"
                            //agregar ficheros al fichero de configuración del proyecto
                            indice.Add(nombreFormulario);
                        }
                    }
                    break;
            }
        }
    }
    addFicheros();
    return true;
}
```

Figura 4.15: Estructura switch de Análisis del fichero index

Cada vez que se encuentra un elemento en el contenido del fichero que cumple la condición de tener el mismo nombre, se determina si ese elemento dispone de un formulario aspx asociado. Para ello, Web Generator busca en la ruta que contiene los formularios desarrollados que tenga por nombre el mismo que el del elemento asociado; en tal caso el generador copia los archivos relacionados con el formulario en la estructura del proyecto de la aplicacion web de tal formal que al final del análisis del fichero todos los formularios necesarios hayan sido incluidos en la estructura de la aplicación web.

Generación del fichero XML global

El usuario mediante la herramienta gráfica genera archivos xml que definen el contenido de cada página que va a formar el sitio web final. Web Generator une todos los ficheros xml para formar un único xml que será el fichero a partir del cuál y utilizando plantillas generarán las páginas web finales. Para ello se van a ir leyendo uno a uno y se formará el fichero xml final, el cuál establecerá el contenido del sitio web totalmente personalizado por parte del usuario cuando definió a través de la herramienta gráfica el contenido de los mismos.

```
public void addFicheros()
{
    //Lectura de archivos xml para formar el xml final
    string[] ficherosDisco = Directory.GetFiles(rutaFichero.Text);
    StreamWriter file = new System.IO.StreamWriter(rutaXmlDestino + "/SitioWeb.xml");
    file.WriteLine("<?xml version=\"1.0\" encoding=\"utf-8\" ?>");
    string[] ficheros = ordenarFicheros(ficherosDisco);
    for (int i = 0; i < ficheros.Length; i++)
    {
        //El fichero de index tiene un tratamiento un poco especial y hay que añadirle
        //el menu construido a partir de las paginas dadas en él.
        if (ficheros[i].Contains("Index.xml"))
        {
            String[] lineas = File.ReadAllLines(rutaFichero.Text + "\\Index.xml");
            String fichero = "";
            List<String> resultado = new List<String>();
            for (int j = 0; j < lineas.Length; j++)
            {
                fichero = lineas[j];
                file.WriteLine(fichero.ToString());
                if (lineas[j].StartsWith("<Site>"))
                {
                    addIndice(resultado, indice);
                    fichero = "<Items>";
                    for (int k = 0; k < resultado.Count; k++)
                    {
                        fichero += resultado.ElementAt(k) + "\n";
                    }
                    fichero += "</Items>";
                    file.WriteLine(fichero.ToString());
                    fichero += "</Site>";
                    j = lineas.Length;
                }

            }
        }
        else
        {
            String[] lineas=File.ReadAllLines(ficheros[i]);
            for (int j = 0; j < lineas.Length; j++)
            {
                file.WriteLine(lineas[j].ToString());
            }
        }
    }
    file.WriteLine("</sitioWeb>");
    file.Close();
}
```

Figura 4.16: Generación del Fichero XML global

4.2.3. Ejemplo de utilización de Web Generator

Una vez realizado el recorrido por la programación del generador se desea mostrar su funcionamiento a través de un ejemplo. Tomando como punto de partida los ficheros XML que forman un sitio web académico para que se vea el

resultado de la herramienta desarrollada.

Generación de Código

Dentro del interfaz gráfico de Web Generator se selecciona la carpeta donde se encuentran los ficheros XML de partida y se especifica el directorio de salida para la generación del proyecto web final.

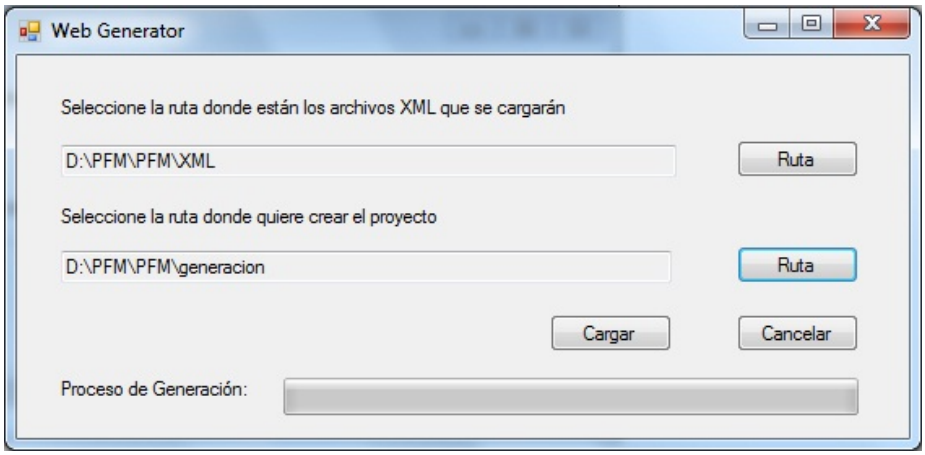


Figura 4.17: Web Generator Verificando los datos de generación

Una vez seleccionados estos datos, se produce a iniciar el proceso de generación pulsando en el botón disponible a tal efecto en la herramienta, tras lo que nos pregunta si es correcta la ruta de generación. Una vez establecida la corrección de la ruta de generación, el generador realiza su función.

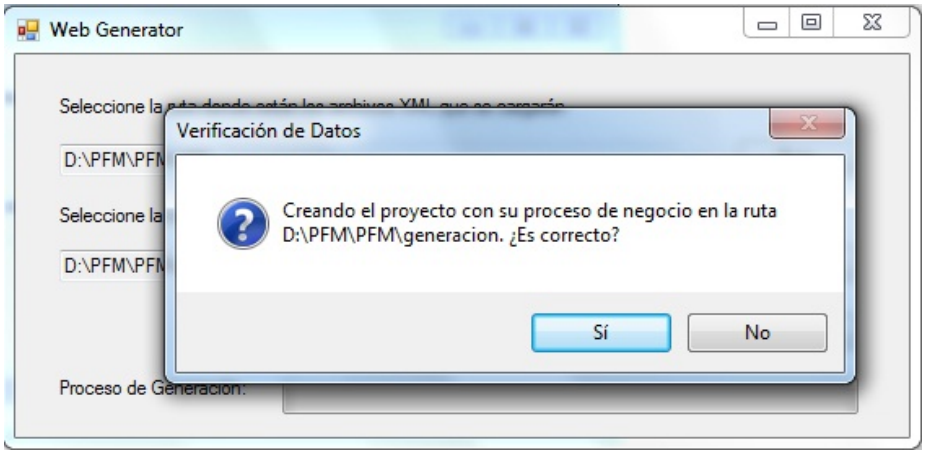


Figura 4.18: Interfaz del Generador con los datos necesarios

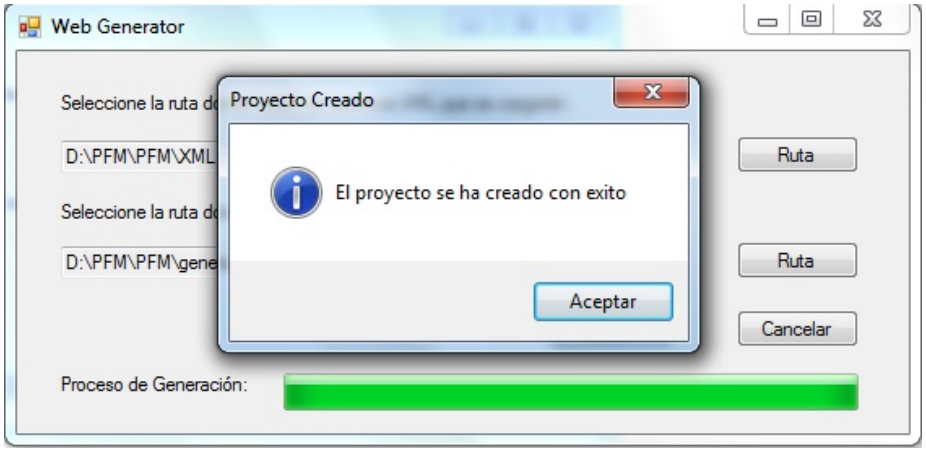


Figura 4.19: Fin de generación del proyecto

Al término del proceso de generación, dentro de la carpeta seleccionada como destino de la operación se puede observar la estructura generada. Dentro de un directorio llamado estructuraVacia se encuentran otros dos directorios correspondientes a la aplicación web generada y al servicio que se construye a partir del archivo de modelado, tal y como se muestra en la siguiente imagen.

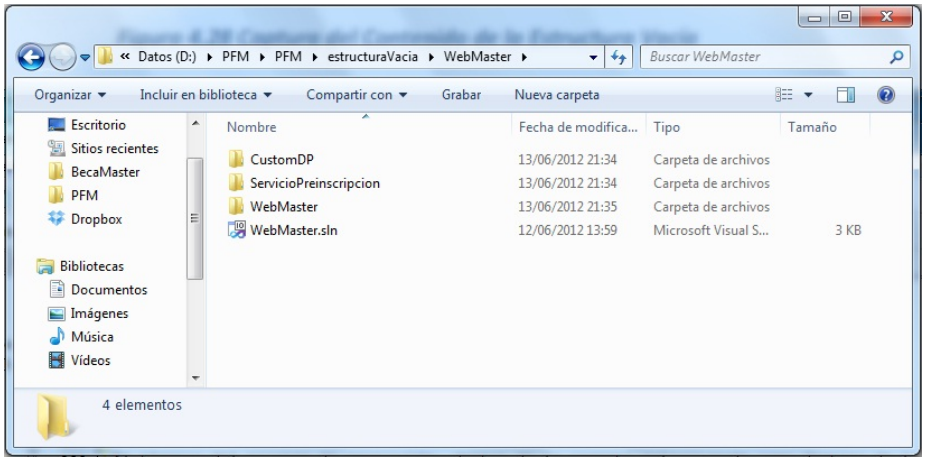


Figura 4.20: Captura del contenido de la Estructura Vacía

Al finalizar todo este proceso de generación se obtiene un proyecto de Visual Studio 2010 en el que mediante plantillas de transformación y el fichero XML global generarán un proyecto web final, que se detallará en el siguiente apartado.

4.3. Proyecto web Generado

Gracias a todas las herramientas anteriores, hemos logrado generar un sitio web en visual studio 2010. Mediante la generación anterior se ha conseguido construir el sitio web personalizado, en función del meta-modelo introducido tendremos unas páginas u otras. Además se ha construido el fichero XML que contendrá toda la definición que se realizó mediante la herramienta gráfica que se utilizará mediante las plantillas de transformación para lograr construir el sitio web personalizado y de forma automática para el usuario.

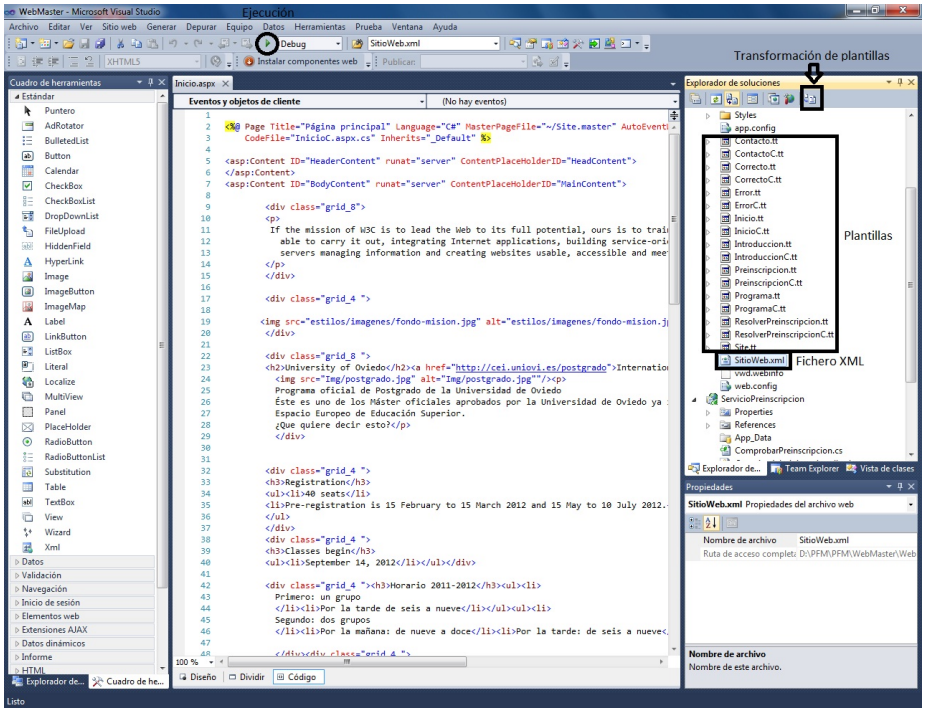


Figura 4.21: Proyecto Web generado

Como se puede ver en la figura anterior, las plantillas y el fichero XML junto con el botón de transformaciónd e plantillas, logran obtener formularios personalizados en función del contenido del fichero XML sin que el usuario toque una línea de código.

Al pulsar en la ejecución señalada se va generar un sitio web académico que se podría publica en cualquier servidor y podría estar funcionando correctamente.

En este caso se ha realizado el desarrollo del sitio web del Máster de ingeniería Web, que es un máster oficial de postgrado de la universidad de Oviedo, que se va a utilizar como muestra de la capacidad de la propuesta presentada.

Para ver un ejemplo concreto de esta transformación, en la siguiente imagen, se aprecia como a partir de código XML es posible generar código HTML5 interpretado por cualquier navegador.

Figura 4.22: Proceso de transformación

La utilización de HTML5 junto con el framework960 de CSS y CSS3, permiten que el sitio web generado sea visible en cualquier tamaño de pantalla desde la cual se consulte dicha web.

Capítulo 5

Resultados Obtenidos

5.1. Herramientas evaluadas

Para comprobar la eficacia de Webdesing se ha realizado una serie de pruebas comparándola con otras dos herramientas similares Wix y Wordpress. Existen herramientas similares a la propuesta que se describe en este proyecto como Wix (http://es.wix.com/) y Jimbo (http://es.jimdo.com/). En estas herramientas se permite realizar un diseño personalizado de la página web de forma gráfica y sin tener muchos conocimientos de programación.

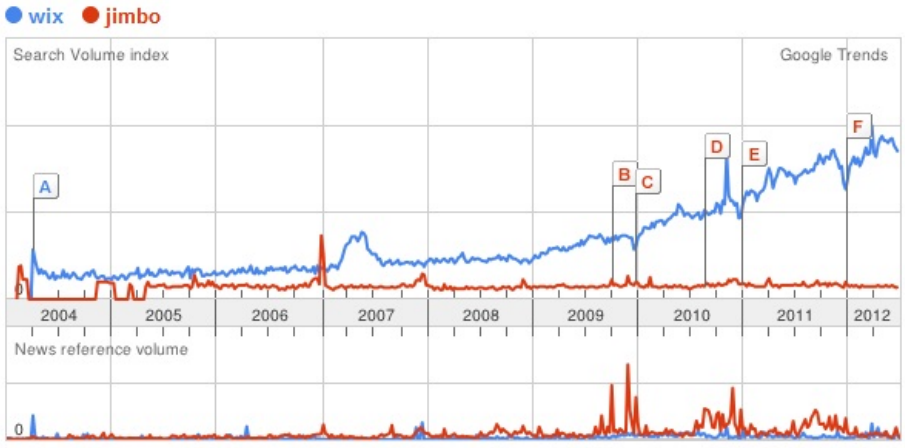


Figura 5.1: Google Trends Wix vs Jimbo

Según Google Trends como muestra la imagen 5.1 wix se utiliza más que Jimbo, por eso se ha elegido Wix para realizar la comparación con Webdesign. La desventaja de Wix es que a la hora de publicar el sitio web es necesario albergar

publicidad, además sólo tiene soporte para páginas en las que no se necesite interactuar con el servidor, es decir, no contempla (en la versión gratuita) un soporte de base de datos ni la posibilidad de almacenar información. A pesar de que Wix tiene una herramienta gráfica muy completa y sencilla de utilizar, las carencias de almacenamiento y tratamiento de datos, no permitirían realizar una web académica como la que se podría realizar con la herramienta propuesta en este proyecto.

Otro grupo de herramientas que permiten diseñar un sitio web más completo son: jommla, drupal o wordpress, en este caso, según google trends, wordpress es el más utilizado como se puede ver en la siguiente ilustración.

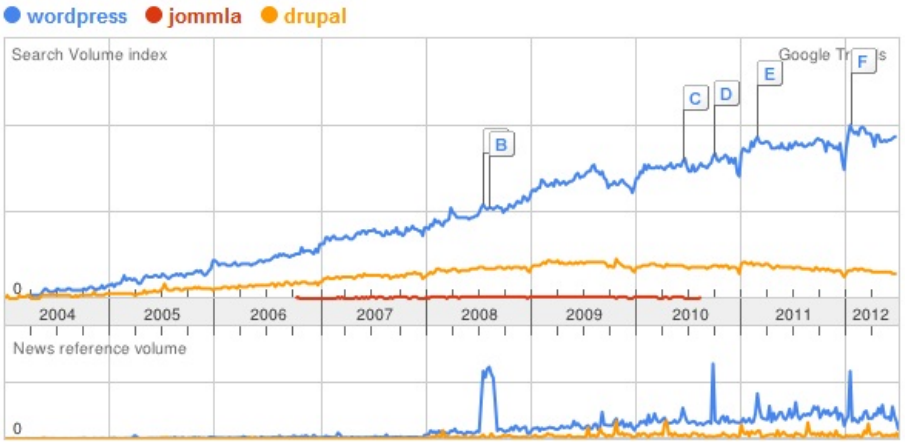


Figura 5.2: Google Trends WordPress vs Joomla vs Drupal

Para utilizar wordpress es necesario instalar apache y mysql y configurarlos para que funcione todo correctamente. La filosofía de wordpress consiste en diseñar sus páginas web como si fueran blogs. En algunos casos, esta filosofía se adaptará más o menos, concretamente, en nuestro caso no encajaría mucho ese diseño, porque lo que se persigue es informar de los contenidos académicos que se publiquen además de permitir resolver dudas en línea mediante un formulario de contacto, así como permitir la preinscripción online en los estudios ofertados. Por lo que, en este caso, a pesar de ser una herramienta muy utilizada en el mundo de generación de páginas, no encaja mucho en nuestro modelo. Otra desventaja que presenta wordpress es que para añadirle interacción con una base de datos se necesitan conocimientos de programación y más concretamente conocimientos en php, que es el lenguaje que utiliza wordpress para sus pluggins y para su configuración interna. Estas herramientas descritas generan páginas web, pero de una forma que es difícil de adaptar a las necesidades que pueden surgir en un ámbito como puede ser el académico. Webdesing, logra la sencillez de wix y la complejidad de wordpress de forma que no le afecte dichas complejidades al

administrador que va a tener que desarrollar la página web. De forma que sea sencillo agregar contenido y organizarlo, además de disponer de un proceso de preinscripción y de contacto sin tener conocimientos previos de programación.

5.2. Interpretación de los resultados

Tanto Wix como Wordpress no tienen un enfoque académico por lo que las pruebas realizadas son de carácter aislado para que la evaluación de los resultados sea más equiparable. Para el desarrollo de las pruebas se utilizaron a 20 usuarios de nivel medio para la realización de cada prueba. A cada prueba se le podía dar una puntuación de entre 0 y 10, significando 0 nada satisfecho y 10 muy satisfecho. Los resultados que se muestran en la gráfica son la media de todas las puntuaciones en cada prueba para cada herramienta.

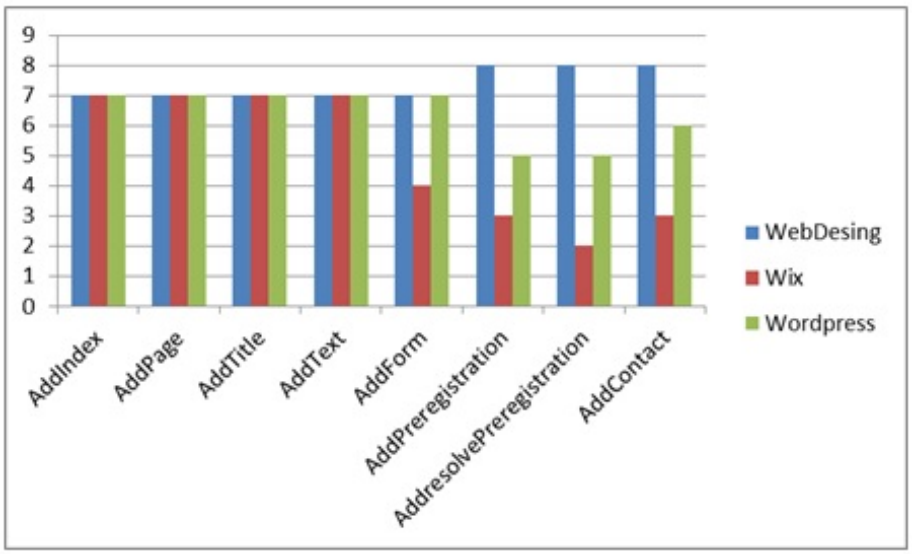


Figura 5.3: Gráfica comparativa

Los resultados muestran que WebDesign es mejor en los casos en los que es necesario añadir componentes con una carga de programación por detrás importante, como lo son un formulario, una preinscripción, resolver esa preinscripción o añadir el formulario de contacto. Es en esos casos en los que WebDesing logra satisfacer más al usuario porque consigue que esos procesos que con las otras herramientas son más costosos con nuestra propuesta sean automáticos. Los resultados muestran que Webdesing ha tenido muy buena aceptación, y mejores resultados que Wix o Wordpress, y por tanto es más adecuada a la hora de desarrollar un sitio web académico.

Capítulo 6

Conclusiones y Trabajo Futuro

El diseño web mediante herramientas automáticas es una técnica que esta aumentando actualmente, cada vez hay más personas que necesitan un sitio web porque internet es un escaparate mundial y abre nuevos mercados a coste relativamente bajo.

Mediante la definición del meta-modelo a través de Webdesing, se ha conseguido generar un sitio web académico de forma fácil y sencilla.

A través de la utilización de plantillas de transformación se ha logrado obtener un sitio web vivo, en el que las actualizaciones son sencillas de realizar, así como la traducción del sitio web a otros idiomas. El uso de tecnologías actuales y bastante estandarizadas como son HTML5 o CSS3 permiten que la web generada este accesible desde móviles, tablets y pcs.

El trabajo futuro pasará por seguir madurando la generación del meta-modelo para abstraerse un poco más y defnir un meta-metamodelo que permita desarrollar páginas sin ningún ámbito en concreto, en las que los diferentes metamodelos definan un escenario de utilización.

Otra aportación sería desarrollar las plantillas de transformación en otra plataforma.

Otra ampliación podría ser generar una aplicación web para Android e iOS, con el fin de que se aprovecharan todas las características de estos sistemas y se agilizará la consulta y utilización de la herramienta en estas plataformas.

Bibliografía

[1] Javabeans. jsr 220: Enterprise javabeanstm, version 3.0. sun microsystems, inc. technical report. `http://java.sun.com/products/ejb/docs.html`.

[2] Windows workflow foundation. `http://msdn.microsoft.com/es-es/library/dd489441.aspx`.

[3] The free online dictionary. 'virtual machine'. `www.foldoc.org`, 2002.

[4] *Meta object facility 2.0 xmi mapping specification, v2.1. Technical report, Object Management Group*, 2005.

[5] Los lenguajes específicos de domino. `http://www.dosideas.com/actualidad/487-los-lenguajes-especificos-de-dominio.html`, 2009.

[6] A Van Deursen. Domain-specific languages versus object-oriented frameworks: A financial engineering case study. *Language*, pages 35–39, 1997.

[7] David S. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing*. John Wiley & Sons, 2003.

[8] André W B Furtado and André L De M Santos. Tutorial : Applying domain-specific modeling to game development with the microsoft dsl tools. *In 3rd Brazilian Symposium on Computer Games and Digital Entertainment SBGames2006*, 2006.

[9] B. Cristina Pelayo García-Bustelo, Juan Manuel Cueva Lovelle, and Aquilino A. Juan Fuente. C3net: Smart environment for .net code generation using mda. In Hamid R. Arabnia, editor, *IC-AI*, pages 682–688. CSREA Press, 2006.

[10] Vicente García-Díaz, Héctor Fernández-Fernández, Elías Palacios-González, B Cristina Pelayo G-Bustelo, Oscar Sanjuán-Martínez, and Juan Manuel Cueva Lovelle. Talisman mde: Mixing mde principles. *Journal of Systems and Software*, 83(7):1179–1191, 2010.

[11] Vicente García-diaz, Edward Rolando Núnez-valdéz, Luis Joyanes Aguilar, Juan Manuel, Cueva Lovelle, Oscar Sanjuán Martínez, B Cristina Pelayo García, Carlos Enrique Montenegro-marin, and Jordan Pascual Espada.

Generación incremental de código basada en modelos. *Technology Journal LAC*, 1:181–194, 2012.

[12] A. Kleppe, J. Warmer, and W. Bast. *MDA Explained: The Model Driven Architecture - Practice and Promise*. 2003.

[13] J.M. Cueva Lovelle. *Handbook of Object Technology*. 1998.

[14] Stephen J Mellor and Marc J Balcer. *Executable UML: A Foundation for Model-Driven Architecture*. Addison-Wesley Professional, 2002.

[15] Stephen J Mellor, Kendall Scott, Axel Uhl, and Dirk Weise. *MDA Distilled: Principles of Model-Driven Architecture*, volume 88. Addison-Wesley, 2004.

[16] J. Miller and J. Mukerji. Mda guide version 1.0.1. Technical report, Object Management Group (OMG), 2003.

[17] Carlos E Montenegro-Marin, Juan Manuel Cueva-Lovelle, Oscar Sanjuán Martínez, and Vicente García-Díaz. Domain specific language for the generation of learning management systems modules. *Journal of Web Engineering*, 11(1):23–50, 2012.

[18] Francisco Ortín Soler. *Sistema Computacional de Programación Flexible diseñado sobre una Máquina Abstracta Reflectiva No Restrictiva*. PhD thesis, Universidad de Oviedo. Departamento de Informática, 2002.

[19] B. Pelayo García-Bustelo. Talisman : Desarrollo ágil de software. *Doctor*, 2007.

[20] B.C. Pelayo García-Bustelo, J.M. Cueva Lovelle, and and A.A.. Suárez Torrente, M.C. Juan Fuente. C3net: Framework para la construcción de mda en la plataforma .net. 2005.

[21] C. Raistrick, P. Francis, J. Wright, and C. Carter y I. Wilkie. *Model Driven Architecture with Executable UML*. 2004.

[22] Jeffrey Richter. *Applied Microsoft .NET Framework Programming*. Microsoft Press A Division of Microsoft Corporation, 2002.

[23] Jaime Solis Martinez. Desarrollo de software dirigido por modelos utilizando bpmn muy simple. Master's thesis, Universidad de Oviedo, 2011.

[24] Thomas Stahl and Markus Völter. *Model Driven Software Development: Technology, Engineering, Management*. 2006.

Capítulo 7

Apéndices

A Meta-model for designigning an academic web page

María Macías Alonso • Jaime Solís Martínez • B. Cristina Pelayo G-Bustelo • Juan Manuel Cueva Lovelle

Abstract:

In this paper we introduce the definition of a graphical meta-model through the use of the WebDesign tool. Using this meta-model we will be able to define the model by a set of rules defined in XML files. These files will be interpreted and in combination with the transformation templates they will generate a complete academic website that can be visited from any web explorer.

Keywords: web page generation, XML, MDA, DSL, Workflow.

María Macias Alonso • Jaime Solís Martínez • B. Cristina Pelayo G-Bustelo• Juan Manuel Cueva Lovelle

Department of Computer Science, University of Oviedo, Oviedo, Asturias, Spain.

e-mail: mariamac90@gmail.com

1. Introduction

Actually, websites are undergoing an increase in the number of users and in popularity. Websites are pages where all the existing information is accessible through a single point of entrance (Udgata, Vanam, & Madhav, 2008).

In order for non-computer related people to be able to develop websites different types of tools have appeared in the market. There are tools that allow the creation of websites in an easy and quick way, like Joomla, Drupal or Wordpress; There is a comparison between these tools in (Joomla & Patel, 2011). There is also another type of tools, which are focused in the creation of websites for learning and user interaction; these tools intend to favour the sharing of the learning objects between different content management systems (CMS). In (Powel & Gill, 2003) we can find a definition of CMS as well as an analysis of this type of products.

Model driven development (MDD) is very useful when dealing with CMS. In (Montenegro-Marin, Cueva-Lovelle, Martínez, & García-Díaz, 2012) MDD is used to create a common meta-model for the creation of learning management systems (LMS); the meta-model introduced in this research is based on a study of different platforms like Moodle or Claroline. This research also introduces a Domain Specific Language (DSL) tool that validates the meta-model and allows the deployment of the learning modules in three different CMS: Moodle, Claroline and Atutor.

Code transformation and the Model Driven Architecture (MDA) methodology (Miller & Mukerji, 2001) allow the definition of a meta-model like in (Daissaoui, 2010), where it enables the transformation of a class diagram into a model/view/controller (MVC) schema that can be used to develop a complete website. In order to achieve the code transformation needed in our proposal we will use XML (Specification & Group, 2003) files, which are used in other works like (Pietriga & Vion-Dury, 2001).

The main goal of our proposal is to simplify the creation of a website to people with little or none programming skills, making the web design and development process an easy and simple task. In order to achieve this goal we have defined a meta-model with fixed rules; these rules will be defined via XML files, where the website model is defined and this model will be finally interpreted to generate the site. MDA technology enables the development of this proposal, which has got the following features: portability, interoperability and reusability. Relying on the concepts found in MDA and using the meta-models we will be able to generate code for different platforms like .NET, Java or PHP.

A prototype tool has been developed in order to promote the practical application of our proposal. This tool uses workflow processes (Chappell, 2005) to manage a database, offering received data treatment and a persistent storage. The prototype has been developed in the .NET platform, although it could be ported to any other wished platform.

2. Related Work

XML is becoming the dominant standard in the hypertext level of the website management. In (Chatvichienchai, Iwaihara, & Kambayashi, 2004) the goal is to integrate XML with relational database systems that allow the storage, retrieval and update of XML files.

Learning across the web is done in a daily basis and in other works like (Chen, Li, & Jia, 2005) the learning process is eased by automatically generating an e-book for a specific user and a specific subject. This type of technology can be make the learning process easier.

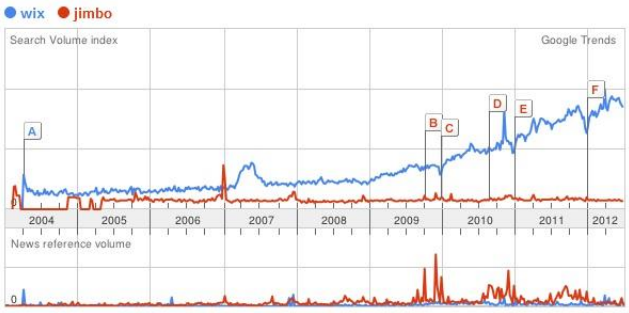
The main contribution of this proposal is the definition of a meta-model that can generate an academic website without the need of writing code. Once the meta-model is complete it will undergo several transformations that will process the contents in the meta-model and generate the site using XML files and MDA methodology. This methodology offers the support needed for the definition of the rest of the models; in (García-diaz et al., n.d.) MDA methodology is used to generate artefacts incrementally based on models, optimizing the computational resources, minimizing the impact upon the applications that could already be in production and allowing traceability for the system's evolution. MDA application scenario is very wide, in (García-Díaz et al., 2010) it is used for generating and controlling a food traceability system.

In an e-learning scope, MDE and MDA along with DSL offer the ideal support for the generation of meta-models. In (Montenegro-Marin et al., 2012) this approach is used to create a meta-model based on a comparison between LMS to develop a graphical tool capable of generating the learning models that can be deployed in different CMS.

As it can be seen, early works benefitted from MDE and MDA technologies, solving problems efficiently and making the development process more agile. Thus, these aspects in combination with the efficiency and success of these technologies have been considered in this proposal for developing a graphical tool for the definition of an academic web page.

3. Considerations

There a several commercial tools similar to the proposal described in this paper; examples of these tools are Wix (web de Wix) and Jimbo (web de Jimbo). These tools give the user the ability to create a custom design of the web page through a graphical interface without the need of advanced programming knowledge. We have focused our efforts in evaluating Wix because Google Trends states that this tool is used more than Jimbo, as it can be seen in the following graph.



The main disadvantage in Wix is the need to include ads in order to publish the website but it also has other limitations: it only supports the use of pages with no need for server interaction, it does not contain support for data bases (at least in the free version) nor the possibility of storing information.

Despite the complete and easy to use graphical interface that Wix possesses, the limitations related with the use of databases and information storage do not allow the creation of an academic web page identical to the one that could be designed with the WebDesign, the tool proposed in this paper.

There are other design tools that allow the creation of more complete websites; some of theses tools are Joomla, Drupal or Wordpress. In this case, Google Trends establishes that Wordpress is the most widely use of this group, as it can be seen in the following figure.

In order to use Wordpress the installation and configuration of apache and MySQL is mandatory in order for the website to work correctly. Wordpress' philosophy consists in designing web pages like blogs; in some cases this could fit the needs but in our case, an academic website, this design does not match our needs because the main goals of our site is to inform about the academic contents published as well as allow online interaction for problem resolution through a contact form and online preregistration for the offered study programs.

Another disadvantage of Wordpress is the need of programming skills in order to include database interaction, as the user must be capable of programming using php in order to install and configure the plugins needed for including this feature.

The described tools are capable of generating web pages but there are several difficulties in adapting these sites to the needs that can arise in an academic context. These difficulties favour the definition of the tool proposed in this paper, which will be described in detail in the following section.

4. Proposed system.

WebDesign is capable of joining the ease of use present in Wix with the complex web structure that can be created with Wordpress, managing to reduce the configuration issues that the web page administrator comes through when using these commercial web designing approaches and being able to include a contact form and a preregistration service without the need of any programming skills.

WebDesign is capable of designing, through a graphical interface, a meta-model that enables the definition of a model for generating academic websites. WebDesign will generate XML files containing the definition of the model's rules. By means of a transformation process the XML files will be interpreted and with the inclusion of transformation templates (http://msdn.microsoft.com/es-es/library/bb126445.aspx) the academic website will be generated. These templates manage to create web pages in various languages in an easy and quick way.

The following figure shows how WebDesign is capable of defining the meta-model graphically without the need of programming skills and how this meta-model generates a complete academic website through a simple transformation.

4.1 Meta-model

WebDesign will build the meta-model of the web page and generate XML files containing the description of the meta-model's rules. There are several models inside our meta-model:

- Enumeration model: this model contains the definition of all the elements that will be included in the website.

```
<siteWeb title="web engineering master">
<Site>
 <Initiation/>
 <Program/>
 <Introduction/>
 <Preregistration/>
 <Resolvepreregistration/>
 <Contact/>
 </Site>
 </sitioWeb>
```

- Page model: this model contains the definition of the content included in each of the pages contained in the website.

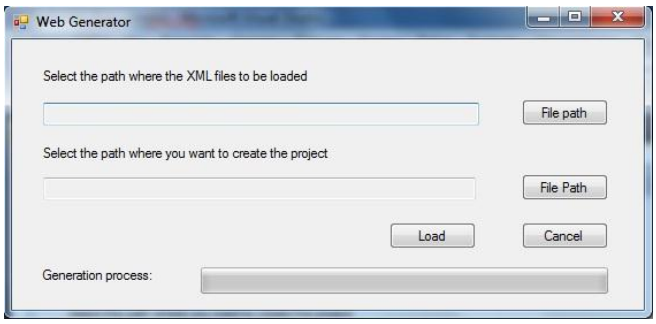
```
<titulo3>Two specialties: Professional and Research </titulo3>
<texto>
The Master is a mixed supply, the
which means that the student can choose one or the other
specialization.
During his second year, students decide if you want to get
a graduate degree with a qualification to practice
profession, according to new European regulations,
or if instead you want a title equivalent to
Current PhD to the achievement of
his doctoral thesis.
<enlace direccion="programa/">More on proprietary</enlace>
</texto>
</bloque>
<bloque col="12">
<titulo3>Organizers</titulo3>
<imagen ruta="Img/empresas/uniovi.gif"/><texto>
University of Oviedo</texto>
<imagen ruta="Img/empresas/euitio.gif"/><texto>
School of Computer Engineering</texto>
</bloque>
```

- Form model: model containing the definition of the contents needed in a web form.

```
<Contact>
    <Fields>
      <field Id="Name" Text="Name"
      validacion="Enter your name, please!"/>
      <field Id="Email" Text="E-Mail"
      validacion="Enter your e-mail, please!"
      formato="E-mail format invalid"/>
      <field Id="subject" Text="Subject"
      validacion="Enter the subject, please!"/>
      <field Id="Message" Text="Message"
      validacion="Enter your message, please!"/>
    </Fields>
    <buttonSend Id="Send" Text="Send"/>
  </Contact>
```

4.2 Transformation process

In order to match the proposed goals for our proposal we must process the XML files generated by WebDesign. In order to complete this task we have designed a simple tool that receives two paths: the path of the XML files produced by WebDesign and the path where the user wants the web page to be generated.



The transformation phase is a two-step process. In first place, all the XML files generated by WebDesign will be put together in a single file that represents the whole website whilst the second step creates a web project that contains the transformation templates that will generate a web page via the XML file.

4.3 Generated Web Project

The generated web project contains all the information introduced by the user in WebDesign, which will be transformed by the templates in order to create a complete academic website. In this case we have done the creation of the website for the Web Engineering Master's Degree at the Oviedo University. This website will be used as an example of the features present in WebDesign.

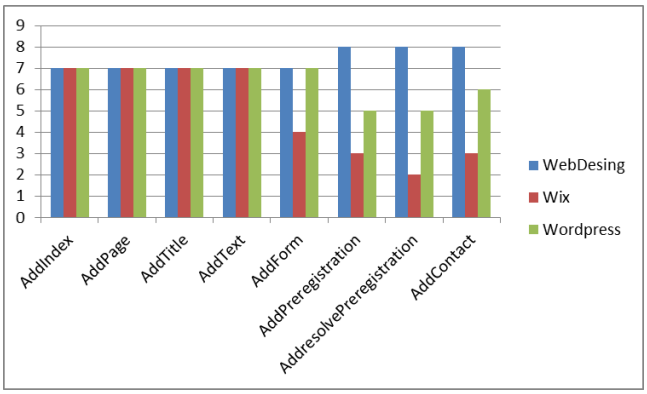
The following image illustrates how we can generate HTML5 code that can be interpreted by any browser by making the content of a XML file undergo a simple transformation process.

Using HTML5 with the framework960 of CSS and CSS3 we also manage to adapt the size of the website to the capacity of the display from where the website is accessed.

5. Performance evaluation

In order to test the efficiency of WebDesign we have done some tests comparing it with two of the similar tools we have presented in earlier sections: Wix and Wordpress. Although Wix and Wordpress do not have an academic focus we have decided to make them undergo an isolated testing process as a way of making the results comparable to those obtained by WebDesign.

20 different users with a medium level of computer knowledge did these tests. Each of tests done where focused in completing a specific task with the tools and the users could give each task a value between 0 (not satisfied) and 10 (totally satisfied). The results shown in the following graph represent the mean value of the 20 tests done for each of the tasks.



The testing results show that WebDesign is better in the scenarios where users need to add web components that require some programming load; this is the case of the form and preregistration related pages included in the site. The user obtains greater satisfaction in these tasks when using WebDesign because our tool makes them nearly automatic whilst commercial tools require the user to do configuration tasks once these components have been added.

In conclusion, the results in the graph show that WebDesign has obtained a good acceptance rate and that we have managed to get better results than Wix or Wordpress, which allow us to think that WebDesign is a better approach when designing an academic website.

6. Conclusions & future work

The designing of websites through the use of automatic tools is a technique with a growing rate of popularity, as Internet is a worldwide showcase where people are able to advertise their products at a relatively low cost and these tools enable users to generate complete web pages in a short amount of time.

Through the definition of a meta-model using WebDesign we have managed to build an academic website easily and

quickly. Once WebDesign has created the XML files containing the information of the designed website, the transformation templates are capable of creating a live website, where updates and translation to other languages are tasks that can be completed with very little effort. The accessibility rate of the website grows due to the use of actual and standard technologies like HTML5 and CSS3, which make the site accessible not only from computers but also from mobile devices like tablets and smartphones.

Our future work is focused in two main aspects of the WebDesign tool: the meta-model and the transformation templates. We intend to improve the meta-model in order for WebDesign to create not only academic web pages but multiple purpose sites and we are going to develop transformation templates in other platforms.

References

Chappell, D. (2005). Introducing Microsoft Windows Workflow Foundation: An Early Look. *httpmsdnmicrosoftcomenuslibraryaa480215aspx*. Retrieved from http://msdn.microsoft.com/en-us/library/aa480215.aspx

Chatvichienchai, S., Iwaihara, M., & Kambayashi, Y. (2004). Authorization Translation for XML Document, (December 2002), 111-138.

Chen, J., Li, Q., & Jia, W. (2005). Automatically Generating an E-textbook on the Web. *World Wide Web*, *8*(4), 377-394. doi:10.1007/s11280-005-1319-5

Daissaoui, A. (2010). Applying the MDA Approach for the automatic generation of an MVC2 web application. *Meta*.

Deursen, A. V. (1997). Domain-Specific Languages versus Object-Oriented Frameworks: A Financial Engineering Case Study. *Language*, 35-39. Citeseer. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.2995&rep=rep1&type=pdf

García-Díaz, V., Fernández-Fernández, H., Palacios-González, E., G-Bustelo, B. C. P., Sanjuán-Martínez, O., & Lovelle, J. M. C. (2010). TALISMAN MDE: Mixing MDE principles. *Journal of Systems and Software*, *83*(7), 1179-1191. Elsevier Inc. doi:10.1016/j.jss.2010.01.010

García-diaz, V., Núnez-valdéz, E. R., Aguilar, L. J., Manuel, J., Lovelle, C., Martínez, O. S., García-, B. C. P., et al. (n.d.). Generación incremental de código basada en modelos, 181-194.

Joomla, S.-, & Patel, S. K. (2011). Performance Analysis of Content Management, *21*(4), 39-43.

Landon, B., Henderson, T., & Poulin, R. (2006). *Peer Comparison of Course/Learning Managemment Systems, Course Materials Life Cycle, and Related Costs. Final Report* (pp. 1-90). Massachusetts Institute of Technology. Retrieved from http://www.wcet.info

Miller, J., & Mukerji, J. (2001). Model Driven Architecture, Object Management Group www.omg.org/mda. *Object Management Group Draft Specification ormsc20010701 July*, *9*. Retrieved from http://www.omg.org/mda

Montenegro-Marin, C. E., Cueva-Lovelle, J. M., Martínez, O. S., & García-Díaz, V. (2012). Domain Specific Language for the Generation of Learning Management Systems Modules. *J Web Eng*, *11*(1), 23-50. Retrieved from http://dblp.uni-trier.de/db/journals/jwe/jwe11.html#Montenegro-MarinCMG12

Pietriga, E., & Vion-Dury, J. Y. (2001). VXT: Visual XML Transformer. *Proceedings IEEE Symposia on HumanCentric Computing Languages and Environments Cat No01TH8587*, 404-405. Ieee. doi:10.1109/HCC.2001.995300

Powel, W., & Gill, C. (2003). Web Content Management Systems in Higher Education, (2), 43-50.

Specification, A. A., & Group, O. M. (2003). XML Metadata Interchange (XMI) Specification. *Management*, *01*(May), 1-134. Retrieved from http://www.omg.org/spec/XMI/

Udgata, S. K., Vanam, S., & Madhav, M. N. V. (2008). Regression based automated test tool for web portals. *TENCON 2008 - 2008 IEEE Region 10 Conference*, 1-6. Ieee. doi:10.1109/TENCON.2008.4766818