Arquitectura de MIDAS-CASE: una herramienta para el desarrollo de SIW basada en MDA

Juan M. Vara, Valeria de Castro, Paloma Cáceres, Esperanza Marcos

Grupo de Investigación Kybele
Universidad Rey Juan Carlos
Madrid (España)
{jmvara,vcastro,pcaceres,emarcos}@escet.urjc.es

Resumen. En los últimos años ha crecido el interés y la importancia del modelado en el desarrollo de cualquier tipo de software. Las ventajas del modelado radican en la facilidad que ofrece un buen diseño tanto a la hora de desarrollar, como a la hora de integrar y mantener cualquier sistema software. En este sentido la aparición de MDA (Model Driven Architecture) ha constituido un avance en la evolución del desarrollo de software. MDA propone la definición y uso de modelos a diferente nivel de abstracción, así como la posibilidad de la generación automática de código a partir de los modelos definidos y de las reglas de transformación entre dichos modelos. MIDAS es un marco metodológico basado en MDA, que propone distintos modelos y reglas de transformación entre ellos, para el modelado completo de Sistemas de Información Web. En este artículo se presenta la arquitectura de *MIDAS-CASE*, una herramienta que permite representar cada uno de los modelos propuestos en MIDAS y realizar las transformaciones entre ellos. Además se presenta el módulo para la definición de servicios Web que permite obtener código WSDL a partir del *modelo WSDL* propuesto en MIDAS.

1 Introducción

En la industria del software surgen continuamente nuevas tecnologías (Java, Linux, XML, SOAP, .NET, etc.) que se hacen cada vez más populares, obligando a las compañías a incorporar rápidamente dichas tecnologías a sus productos y a sus procesos de desarrollo. Además, con intervalos de tiempo cada vez más cortos, van apareciendo nuevas versiones de esas mismas tecnologías, que habrá que volver a considerar antes de haber completado el proceso anterior. Desde el punto de vista de la tecnología de la información, este planteamiento es poco adecuado puesto que generalmente es más importante el conocimiento del propio negocio, de los productos y de los procesos de desarrollo implicados, que la plataforma o tecnología específica de implementación.

Con la idea de que el modelado sea independiente de la tecnología final de implementación, OMG (Object Management Group) ha creado MDA (Model Driven Architecture) [8,13]. MDA es un marco de trabajo para el desarrollo de software dirigido por modelos, que plantea el modelado del negocio, en base a modelos totalmente independientes de la plataforma (PIM), para después transformarlos en modelos específicos de una plataforma determinada (PSM) a partir de guías de transformación entre los diferentes modelos. La principal ventaja de MDA es la posibilidad de transformar, mediante las guías de transformación entre modelos, un único PIM en distintos PSM para las diferentes plataformas y tecnologías, y la

posterior transformación de esos PSM en código. Para aprovechar estas ventajas, en los últimos años han aparecido numerosas herramientas basadas en MDA, que en la mayoría de los casos permiten la generación de código para distintas plataformas, como por ejemplo J2EE o .NET. Algunas son herramientas comerciales como Codagen, ArcStyler, Adaptive, etc. [19], y otras son herramientas de libre distribución como OpenMDX, ModFact, MTL y ATL, entre otras [1].

Nuestra propuesta en este artículo es la definición de la arquitectura de MIDAS-CASE, una nueva herramienta basada en MDA que se desarrolla dentro de MIDAS, un marco metodológico que propone distintos modelos y reglas de transformación entre ellos para el modelado completo de Sistemas de Información Web. El desarrollo de ésta nueva herramienta, viene motivado por dos razones principales. Una de las razones es que se propone como forma de *validación de los modelos* y *las reglas de transformación propuestas en MIDAS*. Para ello MIDAS-CASE cuenta con un repositorio XML, donde se almacenarán los diferentes modelos en formato XML. Los metamodelos de los modelos propuestos se han definido a través de un XML Schema, registrado en el repositorio [16]. La herramienta permitirá crear los distintos modelos propuestos por MIDAS y la validación de los modelos se establecerá de la siguiente manera: si el código XML generado para un modelo concreto, es conforme con el XML Schema almacenado en el repositorio, entonces se considera válido el modelo y las reglas de transformación de ese modelo a XML. De la misma manera si un diagrama concreto A, se transforma en otro B y este último también es conforme con el XML Schema correspondiente al metamodelo de B, entonces las reglas de transformación para transformar A en B, también son válidas.

La otra razón por la que se desarrolla MIDAS-CASE, es para *dar soporte de los perfiles UML de MIDAS*. La mayoría de las herramientas basadas en MDA permiten la representación de modelos estándar de UML [14], como modelos de actividad o modelos de clases, pero debido a la complejidad de ésta herramientas, es difícil o imposible en algunos casos, la inclusión de nuevos modelos propuestos como extensiones a UML. La generación de perfiles UML es una buena alternativa, que conjuga las ventajas de MDA y de la definición de modelos, para los nuevos dominios y entornos específicos [6]. Los perfiles UML permiten particularizar algunos conceptos de UML y generar lenguajes de modelado derivados de UML, a través de los mecanismos de extensión que incorpora el propio lenguaje. Por lo tanto, el uso de perfiles UML para especificar nuevos modelos garantiza que los modelos derivados sean modelos consistentes con UML, cumpliendo así con una de las principales premisas de MDA: el uso de estándares.

La arquitectura de MIDAS-CASE, principal propuesta de este artículo, y que se presentará más detalladamente en la sección 3, contempla tres niveles: la *interfaz de usuario*, la *lógica de la aplicación* y una Base de Datos (BD) XML como *repositorio*. Este repositorio representa una de las aportaciones de MIDAS-CASE, pues la arquitectura de la mayoría de las herramientas basadas en MDA, contiene módulos para el modelado, para compilación de modelos, y módulos para la transformación y generación de modelos, pero ninguna contempla la utilización de una BD para el almacenamiento de los modelos, con las ventajas derivadas de su utilización, como la recuperación del conjunto de modelos en los que aparece una clase determinada, la modificación del nombre de un elemento, en todos los modelos o diagramas que lo incluyen, etc.

El resto del artículo se organiza como sigue: en la sección 2 se presenta una breve introducción a MDA y a MIDAS; en la sección 3 se describe la arquitectura de MIDAS-CASE, detallando cada uno de los componentes de la misma; en la sección 4 se presenta el módulo para la definición de Servicios Web y finalmente, en la sección 5 se presentan las conclusiones más importantes y las líneas de trabajo futuro.

2 MDA y MIDAS

Model Driven Architecture (**MDA** - Arquitectura dirigida por modelos) [8,13] es un marco de trabajo para el desarrollo de software definido por el Object Management Group (OMG), que solventa en gran medida las dificultades expuestas. La característica fundamental de MDA es la definición de modelos como elementos de primera clase para el diseño e implementación del sistema, así como la definición de transformaciones de un modelo a otro de forma que éstas puedan ser automatizadas. MDA define tres niveles conceptuales de modelado en función del nivel de abstracción: en primer lugar, los requisitos del sistema se modelan en un modelo independiente de computación (CIM, Computer Independent Model) que sirve de puente entre los expertos del dominio y los desarrolladores que afrontan la realización del sistema; en el siguiente nivel, se encuentra el PIM (Platform Independent Model), que representa el modelado de la funcionalidad del sistema, omitiendo la plataforma en la que se implementará el sistema; finalmente se encuentra el PSM (Platform Specific Model), que combina las especificaciones contenidas en un PIM con los detalles de la plataforma elegida. Como se ha dicho anteriormente, la principal ventaja de este enfoque de MDA es poder definir transformaciones automáticas entre un PIM y distintos PSM.

MIDAS [2,11] es un marco metodológico dirigido por modelos para el desarrollo ágil de Sistemas de Información Web (SIW) basado en MDA, que propone el modelado de SIW teniendo en cuenta dos dimensiones: por un lado, el grado de dependencia de la plataforma, es decir a nivel de CIM, PIM y PSM; y por otro, los aspectos más relevantes de un sistema de información: contenido, hipertexto y comportamiento. Además, define reglas de transformación entre los distintos modelos. En la figura 1 se muestra la arquitectura dirigida por modelos propuesta en MIDAS.

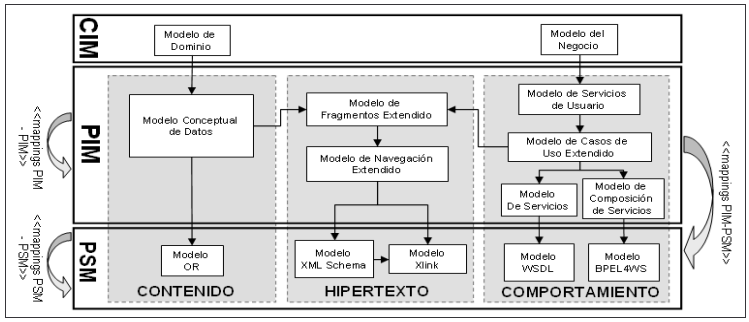

Fig. 1. Arquitectura de MIDAS

En el marco de MIDAS se ha propuesto el uso de algunas de las extensiones de UML que han surgido para el modelado Web [4,9] y se han realizado propuestas de

nuevas extensiones, como las extensiones de UML para Servicios Web (*Modelo WSDL*)[10]; para el diseño de Bases de Datos Objeto-Relacionales (*Modelo OR*) [12,17] y la extensión de UML para representar XML Schema (*Modelo XML Schema*)[18].

Dentro del contexto de MIDAS, se ha abordado el desarrollo de MIDAS-CASE. La herramienta permitirá diseñar cualquiera de los modelos propuestos por MIDAS para el desarrollo de SIW. En este trabajo se presenta concretamente uno de los módulos de la aplicación, que permite la obtención de código WSDL a partir del *Modelo WSDL* propuesto para la definición de servicios Web.

3 Arquitectura de MIDAS-CASE

Se puede pensar en MIDAS-CASE como un conjunto de módulos - uno por cada tipo de modelo o diagrama que MIDAS-CASE permite representar - que colaboran bajo una arquitectura común. En la figura 2 se presenta esta arquitectura común.

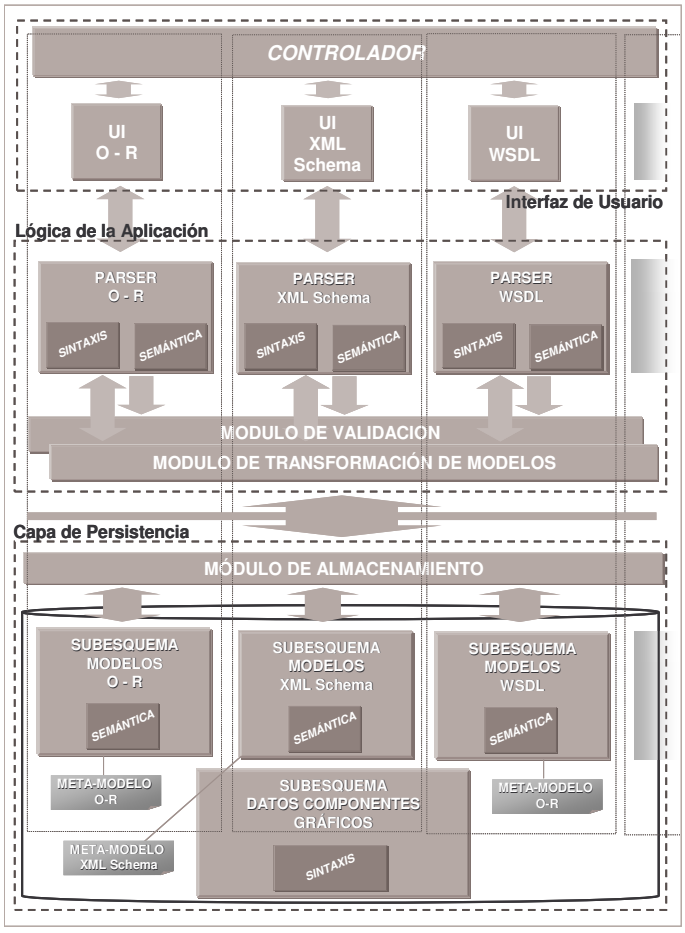

Fig. 2. Arquitectura de MIDAS-CASE

Nótese que en la figura sólo se incluyen los módulos para el diseño de modelos objeto-relacionales, la definición de servicios Web y la definición de XML Schemas. El resto de módulos se pueden ir añadiendo a esta arquitectura común a medida que vayan implementándose.

3.1 Interfaz de usuario

La primera de las capas corresponde a la interfaz de usuario, que presenta el diagrama o modelo que se está realizando en cada momento, y los controles que permiten añadir, eliminar o modificar elementos gráficos en dicho diagrama.

Dentro de esta capa se distinguen dos niveles: en la parte superior de la interfaz de usuario existe un módulo software - denominado controlador - que se ocupa de la parte común de la interfaz (los controles que siempre están disponibles, sea cual sea el tipo de modelo con el que se está trabajando) y de invocar a uno de los módulos del conjunto de módulos software - denominados UI - que constituyen la parte inferior de esta capa. Estos módulos UI se encargan de personalizar la interfaz de usuario: para cada tipo de modelo o diagrama que MIDAS-CASE permite representar, el módulo correspondiente contiene los componentes software que se corresponden con los elementos gráficos que pueden aparecer en ese modelo.

Por ejemplo, cuando se representa un XML Schema, el módulo más externo le pasa el control de la interfaz de usuario al módulo XML Schema UI, éste añade a la interfaz de usuario controles como element, attribute¸ etc., gestiona los eventos que producen esos controles (en general, añadir nuevos elementos al modelo) y gestiona también la interacción del usuario con los elementos gráficos que contiene el modelo (modificaciones sobre dichos elementos), mediante los componentes software que representan a cada elemento gráfico.

3.2 Lógica de la aplicación

La siguiente capa en la arquitectura de MIDAS-CASE corresponde a la lógica de la aplicación. Esta capa se subdivide a su vez en dos niveles: en el primero se sitúa el conjunto de módulos que generan la representación en formato XML del modelo que hay en pantalla - denominados *parser* - y debajo de estos, en el segundo nivel, están los módulos de validación y transformación.

Tal y como muestra la figura 2, el modulo de validación y el módulo de transformación de modelos se encuentran al mismo nivel dentro de la arquitectura, ambos pueden ser invocados desde cualquiera de los módulos parser, pero mientras que el módulo de transformación puede (y debe) a su vez invocar a otro módulo parser para realizar su función, el módulo de validación no puede invocar a ningún parser por sí solo.

3.2.1 Módulos parser

El conjunto de módulos parser constituyen la parte superior de la capa de la lógica de la aplicación. Estos módulos comparten la misma arquitectura interna y desempeñan el mismo cometido.

Cada módulo parser se encarga de recoger la información que le proporciona el correspondiente modulo UI y generar dos documentos XML: uno que recoge la semántica del modelo o diagrama que en ese momento presenta la interfaz, y otro que

recoge su sintaxis, esto es, las posiciones y dimensiones de los distintos elementos gráficos que hay en el diagrama. Al separar el almacenamiento de un diagrama o modelo en dos documentos XML distintos, se consigue separar la información relevante de la información accesoria. Por ejemplo, en un diagrama conceptual de clases, lo importante es saber los atributos que posee una clase concreta, o si entre dos clases dadas existe una asociación, y no el tamaño del rectángulo que representa la clase en el diagrama o si la línea que representa la asociación entre las dos clases es recta o presenta ángulos. Es el documento XML que contiene la semántica del diagrama, el que luego se utiliza en la generación automática de código, la validación del modelo o la transformación del modelo en un modelo de otro tipo.

Para recoger la sintaxis del modelo, se utiliza una variación de GXL (Graph Exchange Language) [7,21], un estándar para el formato de intercambio de información de herramientas basadas en grafos, que se define como un sublenguaje de XML. A partir de la propuesta de GXL, que utiliza una DTD para definir la estructura que debe tener el documento XML, se define un XML Schema ad-hoc que se ajusta mejor a las necesidades concretas de la aplicación. Este XML Schema define la estructura que debe tener el documento XML para recoger la sintaxis de cualquier modelo

3.2.2 Módulo de Validación

Uno de los dos módulos que se encuentran en la parte inferior de la capa de la lógica de la aplicación es el módulo de validación. Su función es informar sobre la corrección del modelo definido. Para ello, toma el documento XML que contiene la semántica del modelo, generado por el parser, y lo valida contra el XML Schema que define el metamodelo del tipo de modelo que se quiere definir. Así, si el documento XML es conforme al XML Schema - o lo que es lo mismo, si el modelo es conforme al metamodelo - entonces el modelo es correcto, en caso contrario, el módulo de validación proporciona al parser la información que específica cuáles son las incorrecciones del modelo, y éste transmite esta información al correspondiente módulo UI para que se informe por pantalla de las posibles incorrecciones del modelo.

3.2.3 Módulo de Transformación entre modelos

En el mismo nivel de la arquitectura que el módulo de validación, se encuentra el módulo de transformación entre modelos. Este módulo genera un nuevo modelo (de otro tipo), a partir del modelo que en ese momento se presenta en la interfaz. Para ello, se definen un conjunto de reglas de transformación (*mapping-rules*) que rigen el proceso de transformación y que permiten pasar de un documento XML, conforme al XML Schema que define el metamodelo del modelo original, a un nuevo documento XML, conforme al XML Schema que define el metamodelo del tipo de modelo destino.

El módulo de transformación aplica esas reglas de transformación sobre los documentos XML con la semántica y la sintaxis del modelo (generados por el parser), obteniendo dos nuevos documentos XML, que contienen la sintaxis y la semántica del nuevo modelo.

3.3 Capa de persistencia

La última de las capas de la arquitectura de MIDAS-CASE es la capa de persistencia. Dada la creciente importancia del lenguaje XML como herramienta para el almacenamiento e intercambio de información, se opta por este formato para guardar los diferentes diagramas y/o modelos que se diseñen con MIDAS-CASE. En este sentido, tras estudiar las distintas opciones para la gestión de contenidos XML [3, 20], se propone la utilización de una BD XML, concretamente se emplea la solución de Oracle para la gestión de XML: Oracle XML DB [5, 15]. El esquema de la BD en que se guardan los documentos XML en Oracle XML DB se define en un XML Schema y se utiliza un XML Schema para definir la estructura de los documentos XML que almacenan los modelos generados con MIDAS-CASE. De esta forma, los XML Schemas elaborados para definir la estructura de los modelos a almacenar han servido también para definir la estructura de la BD donde se van a almacenar. Así, como muestra la figura 2, la base de la capa de persistencia en MIDAS-CASE es una BD Oracle XML DB, conectada con el resto de la aplicación por el módulo de almacenamiento, que lanza las sentencias de inserción, borrado y recuperación de datos, en este caso, documentos XML.

A continuación se describe el proceso de gestión del repositorio: cada tipo de modelo que se desea que MIDAS-CASE sea capaz de representar viene definido por un metamodelo. Esos metamodelos se recogen en distintos XML Schemas - uno por metamodelo - que son registrados en Oracle XML DB, con lo que se genera automáticamente una estructura para el almacenamiento de documentos XML conformes a ese Schema [16]; dado que el XML Schema dicta el metamodelo, cada instancia del XML Schema, es un modelo, por lo tanto, la estructura generada para almacenar documentos XML, es un almacén de modelos. Cuando se habla aquí del modelo, sólo se hace referencia a la semántica del mismo, pero si se desea recuperar un modelo o diagrama tal y como se diseñó en pantalla, es necesario guardar también la sintaxis del modelo, es decir, las características de los componentes gráficos. En este caso, como los conceptos que se manejan cuando se habla de la sintaxis del modelo, son los mismos para todos los tipos de modelo, no es necesario definir una estructura de datos diferente para cada tipo de modelo, un mismo XML Schema sirve para definir la estructura de los documentos XML que contienen la sintaxis de cualquier modelo. Así, se define dicho XML Schema y se registra en Oracle XML DB, con lo que se genera la estructura de almacenamiento para los documentos XML que especifican la sintaxis de los distintos modelos.

El esquema de la BD es finalmente un conjunto de subesquemas, uno por cada tipo de modelo contemplado en MIDAS, más el subesquema que almacena la sintaxis de todos los modelos generados con MIDAS-CASE. Cuando se guarda un modelo, el módulo de almacenamiento recoge el documento XML que contiene su semántica, generado por la capa superior, la lógica de la aplicación, y lo añade al subesquema de la BD que generó en su momento el XML Schema correspondiente al meta-modelo de ese tipo de modelo. Al mismo tiempo, el modulo de almacenamiento añade al subesquema destinado a guardar la sintaxis de los distintos modelos, el documento XML que contiene la sintaxis del modelo.

4 Modulo para la definición de Servicios Web

La figura 2 muestra como, la arquitectura de MIDAS-CASE, además de ser una arquitectura de tres capas, se puede concebir como un sistema compuesto de varios subsistemas, uno por cada tipo de modelo contemplado en MIDAS.

Como muestra de la funcionalidad ofrecida por estos subsistemas, se presenta a continuación el módulo para la definición de Servicios Web, que permite representar un servicio Web gráficamente en un diagrama de clases estereotipadas, y obtener su definición en el lenguaje WSDL, a partir del modelo WSDL propuesto en MIDAS.

4.1 Interfaz de usuario del módulo para la definición de servicios Web

Como se mencionaba en el apartado 3, la interfaz de usuario de MIDAS-CASE varía en función del tipo de modelo que se está representando en cada momento, de modo que siempre estén disponibles los controles correspondientes a los conceptos que se incluyen en el metamodelo del modelo que se está diseñando. En el caso del módulo para la definición de Servicios Web, los conceptos que se manejan los define el lenguaje WSDL (Web Services Description Language) [22]. WSDL no admite una representación gráfica directa, así que para poder representar gráficamente la definición de un servicio Web se ha utilizado la extensión de UML para representar servicios Web, que se presenta en [10]. En ese trabajo se propone una extensión de UML basada en WSDL, que permite, por un lado, representar un servicio Web en notación gráfica, y por otro, la generación automática de código: la descripción en WSDL de un servicio Web, a partir de un diagrama.

En la figura 3 se presenta la interfaz de MIDAS-CASE cuando se ha cargado el módulo para la definición de servicios Web, incluyendo el diseño de un servicio Web que proporciona información sobre vuelos. Una barra de botones en la parte izquierda de la pantalla, permite al usuario añadir al diagrama cualquiera de las clases que se incluyen en [10] como extensiones de una clase UML: binding, definition, element, etc. En la parte superior se proporciona la tradicional barra de herramientas y la parte central presenta el modelo o diagrama que se está diseñando.

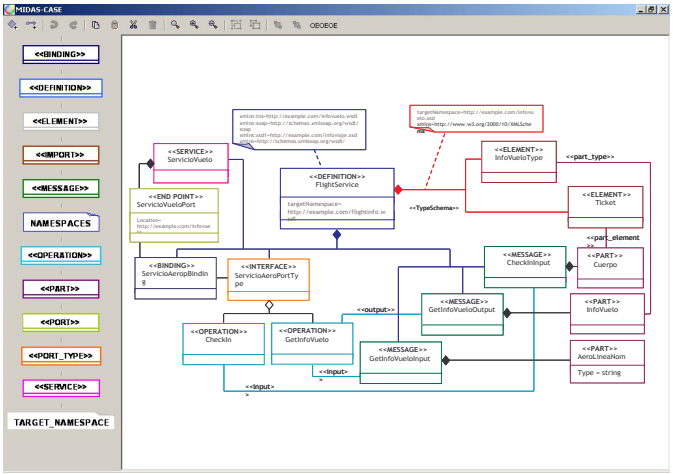

Fig. 3. Interfaz de usuario del módulo para la definición de servicios Web

4.2 Generación de documentos XML en el módulo para la definición de servicios Web

En el caso del módulo para la descripción de servicios Web, el documento XML que contiene la semántica del diagrama es ya la definición de un servicio Web, puesto que es un documento XML conforme al XML Schema que define el metamodelo de la extensión de WSDL. Es decir, cuando MIDAS-CASE valida la representación gráfica de un servicio Web, actúa como un generador automático de código, porque el documento XML que se almacena en el repositorio (el que contiene la semántica del modelo) es la definición en WSDL de un servicio Web.

En la figura 4 se presenta parte de la representación gráfica del modelo que se mostraba completo en la figura 3, concretamente la información referente al *mensaje GetInfoVueloInput*, la *operación GetInfoVuelo* y la relación que los une, y en la figura 5 los documentos XML que recogen la sintaxis (a) y la semántica (b) de dicho servicio Web (sólo los fragmentos que corresponden a la parte del diagrama que se muestra en la figura 4). Se observa que el contenido del fragmento XML de la figura 5(a) hace referencia exclusivamente a posiciones y tamaños, como las dimensiones de la caja con que se representa cada elemento y los puntos por los que debe pasar la línea que les une; mientras que el fragmento XML de la figura 5(b) es un fragmento válido de la definición de un servicio Web en WSDL.

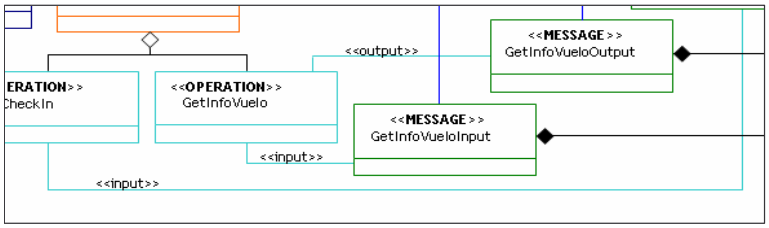

Fig. 4. Fragmento de la representación gráfica de un Servicio Web

```
.....
<element>
        <node type="operation" Id="operation_GetInfoVuelo"
        Name="GetInfoVuelo">
                <position coordX="412" coordY="523"/>
                <dimension height="68" weight="125"/>
         </node>
</element>
<element>
        <node type="message" Id="message_GetInfoVueloInput"
        Name="GetInfoVueloInput">
                <position coordX="548" coordY="554"/>
                <dimension height="68" weight="125"/>
        </node>
</element>
<element>
        <edge type="association" Id="edge15" from="operation_GetInfoVuelo"
                to="message_GetInfoVueloInput">
                <pattern>
                        <start><position coordX="475" coordY="590"/></start>
                        <point><position coordX="475" coordY="610"/></point>
                        <end><position coordX="548" coordY="590"/></end>
                </pattern>
                <text>input</text>
        </edge>
</element>
.....
```

SINTAXIS

```
.....
<message name="GetInfoVueloInput">
        <part name="AeroLineaNom"
        typeOfPart="builtinType">
                <bultinType>xs:string</bultinType>
        </part>
</message>
<message name="GetInfoVueloOutput">
        <part name="InfoVuelo" typeOfPart="udtType">
                <udtType>infoVueloType</udtType>
        </part>
</message>
<message name="CheckInput">
        <part name="Cuerpo" typeOfPart="element">
                <element>xs:string</element>
        </part>
</message>
<interface name="ServicioAeroPortType">
        <operation name="GetInfoVuelo">
                <input name="GetInfoVueloInput"/>
                <output name="GetInfoVueloOutput"/>
        </operation>
        <operation name="CheckIn">
                <input name="CheckInput"/>
        </operation>
</interface>
.....
```

SEMÁNTICA

Fig. 5. Documentos XML que recogen la (a)Sintaxis y la (b)Semántica del fragmento de servicio Web de la figura 4.

4.3 Repositorio para descripciones de Servicios Web

Cuando se presentó la arquitectura general de MIDAS-CASE en el apartado 3, se señaló que el repositorio de datos de la herramienta descansa sobre una BD XML: el repositorio para la gestión de XML que Oracle incorpora en Oracle XML DB.

Para definir el subesquema de esta BD correspondiente a las definiciones de servicios Web, se ha elaborado un XML Schema que contiene el metamodelo de la extensión de UML propuesta en [10]. En la figura 6 se presenta parte de dicho XML Schema. La figura muestra que la definición de cada uno de los tipos y elementos que se incluyen en el Schema, contiene al menos un atributo perteneciente al espacio de nombres http://xmlns.oracle.com/xdb. Estos atributos son anotaciones [16] que proporcionan mayor control sobre la definición del subesquema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://kybele.escet.urjc.es/UMLWSDL2.02Schema-testing" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:xdb="http://xmlns.oracle.com/xdb" xmlns:uws=http://kybele.escet.urjc.es/UMLWSDL2.02Schema-testing
 elementFormDefault="qualified" attributeFormDefault="unqualified">
<!--#################### DEFINICIÓN DE TIPOS COMPLEJOS ##########################-->
<!--######################### Tipo DEFINITONS ####################################-->
<xs:complexType name="definitionsType" xdb:SQLType="XDB_DEFINITIONS_TYPE">
 <xs:annotation>
  <xs:documentation>
   Definición del tipo de Datos DefinitionsType
  </xs:documentation>
 </xs:annotation>
 <xs:sequence>
  <xs:element name="targetNameSpace" type="xs:anyURI" xdb:SQLName="TARGETNAMESPACE" xdb:SQLType="VARCHAR2"/>
  <xs:element name="nameSpace" minOccurs="0" maxOccurs="unbounded" xdb:SQLName="NAMESPACE"
   xdb:SQLType="XDB_NAMESPACE_TYPE" xdb:SQLInline="false" xdb:defaultTable="XDB_NAMESPACE_TABLE">
    <xs:complexType>
     <xs:attribute name="location" type="xs:anyURI" use="required" xdb:SQLName="LOCATION" xdb:SQLType="VARCHAR2"/>
     <xs:attribute name="prefix" type="xs:string" default="default" xdb:SQLName="PREFIX" xdb:SQLType="VARCHAR2"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="include" type="uws:includeType" minOccurs="0" maxOccurs="unbounded" xdb:SQLName="INCLUDE"
   xdb:defaultTable="XBD_INCLUDE_TABLE"/>
  <xs:element name="import" type="uws:importType" minOccurs="0" maxOccurs="unbounded" xdb:SQLName="IMPORT"
   xdb:defaultTable="XBD_IMPORT_TABLE"/>
  <xs:element name="type" type="uws:typeDefType" minOccurs="0" maxOccurs="unbounded" xdb:SQLName="TYPE"
   xdb:defaultTable="XBD_TYPE_TABLE"/>
  <xs:element name="typeSchema" type="uws:typeSchemaType" minOccurs="0" xdb:SQLName="TYPESCHEMA"
   xdb:defaultTable="XBD_TYPESCHEMA_TABLE"/>
  <xs:element name="message" type="uws:messageType" minOccurs="0" maxOccurs="unbounded" xdb:SQLName="MESSAGE"
   xdb:defaultTable="XBD_MESSAGE_TABLE"/>
  <xs:element name="part" type="uws:partType" minOccurs="0" maxOccurs="unbounded" xdb:SQLName="PART"
   xdb:defaultTable="XBD_PART_TABLE"/>
  <xs:element name="interface" type="uws:interfaceType" minOccurs="0" maxOccurs="unbounded" xdb:SQLName="INTERFACE"
   xdb:defaultTable="XBD_INTERFACE_TABLE"/>
  <xs:element name="operation" type="uws:operationType" minOccurs="0" maxOccurs="unbounded" xdb:SQLName="OPERATION"
   xdb:defaultTable="XBD_OPERATION_TABLE"/>
  <xs:element name="binding" type="uws:bindingType" minOccurs="0" xdb:SQLName="BINDING"
   xdb:defaultTable="XBD_BINDING_TABLE"/>
  <xs:element name="service" type="uws:serviceType" minOccurs="0" maxOccurs="unbounded" xdb:SQLName="SERVICE"
   xdb:defaultTable="XBD_SERVICE_TABLE"/>
  <xs:element name="endpoint" type="uws:endpointType" minOccurs="0" maxOccurs="unbounded" xdb:SQLName="ENDPOINT"
   xdb:defaultTable="XBD_ENDPOINT_TABLE"/>
 </xs:sequence>
</xs:complexType>
<!--########################## Tipo includeType ##########################-->
<xs:complexType name="includeType" xdb:SQLType="XDB_INCLUDE_TYPE">
 <xs:annotation>
  <xs:documentation>
  ..............
```

Fig. 6. Parte del XML Schema del Metamodelo de la extensión de UML para WSDL

5 Conclusiones y trabajos futuros

En este trabajo se ha presentado la arquitectura de MIDAS-CASE, una herramienta CASE basada en MDA para el desarrollo de SIW. MIDAS-CASE se engloba en el marco de MIDAS, una metodología dirigida por modelos para el desarrollo ágil de SIW, que propone el modelado de SIW teniendo en cuenta por un lado, el grado de

dependencia de la plataforma, es decir a nivel de CIM, PIM y PSM; y por otro, los aspectos más relevantes de un sistema de información: contenido, hipertexto y comportamiento.

MIDAS-CASE es una herramienta que permite representar gráficamente, en UML extendido, cualquiera de los modelos que forman parte del desarrollo de un SIW propuestos por MIDAS. Además, incorpora reglas que permiten la transformación automática entre estos modelos.

Así, la principal aportación de MIDAS-CASE estriba, no sólo en facilitar el proceso de desarrollo de cualquier SIW, sino más aún, en validar todas las propuestas que se engloban en el marco de MIDAS. Otra de las ventajas de MIDAS-CASE es su facilidad para incorporar nuevas extensiones o perfiles UML: su arquitectura, compuesta de pequeños subsistemas, minimiza la cantidad de trabajo necesaria para incorporar a la funcionalidad de la herramienta, un nuevo tipo de modelo. Para ilustrar esta idea, se ha presentado en este trabajo el módulo ya implementado para la definición de servicios Web, que permite definir servicios Web a partir del modelo WSDL propuesto en MIDAS.

En la actualidad, se continúa trabajando en el desarrollo del resto de módulos que permitirán incorporar a la herramienta, el resto de extensiones del lenguaje UML que ya existen en la actualidad, así como las que vayan apareciendo con posterioridad. De igual forma, se está estudiando la posibilidad de potenciar la generación automática de código en MIDAS-CASE, dotándola de capacidades para obtener implementaciones de los modelos representados en lenguajes como JAVA o .NET.

Agradecimientos

Este trabajo se ha llevado a cabo dentro del proyecto DAWIS (TIC 2002-04050-C02), financiado por la Subdirección General de Proyectos de Investigación del Ministerio de Ciencia y Tecnología y el proyecto EDAD (07T/0056/2003 1) financiado por la Comunidad de Madrid.

Referencias

1. Ambrosio, J., Tools for the code generation. Recuperado de http://www.adtmag.com. Julio, 2003.
2. Caceres, P., Marcos, E., Vela, B., A MDA-Based Approach for Web Information System Development. Workshop in Software Enginnering in conjunction with the UML conference. San Franciso, 2003.
3. Chaudhri, A.B., Rashid, A. y Zicari, R. (Eds.). XML Data Management. Native XML and XML-Enabled Database Systems. Addison Wesley, 2003.
4. Conallen, J., Building Web Applications with UML. Addison-Wesley, 2000.
5. Drake, M., Oracle XML DB White Paper. Oracle Corporation, 2003.
6. Fuentes, L., Vallecillo, A., Una introducción a los perfiles UML. UML y la Ingeniería del Modelado. Novática, vol. 168, pp.6-11, marzo-abril 2004.
7. Holt, R. C., Winter, A. y Schürr, A., GXL: Toward a Standard Exchange Format. *Seventh Working Conference on Reverse Engineering*. IEEE Computer Society. Los Alamitos, 2000.
8. Kleppe, A., Warmer, J., Bast, W., MDA Explained, The Model Driven Architecture: Practice and Promise. Addison Wesley, 2003.

9. Koch, N., Baumeister, H. and Mandel, L., Extending UML to Model Navigation and Presentation in Web Applications. Workshop Modelling Web Applications en el UML'2000. Ed. Geri Winters y Jason Winters, York, England, Octubre, 2000.
10. Marcos, E., De Castro, V., Vela, B., Representing Web Services with UML: A Case Study. International Conference on Service Oriented Computing (ICSOC). M. Orlowska, S. Weerawarana, M.P. Papazoglou, J. Yang (eds.). Springer Verlag, pp.15-27, 2003.
11. Marcos, E., Cáceres, P., De Castro, V.: An approach for Navigation Model Construction from the Use Cases Model. The 16th Conference On Advanced Information Systems Engineering. CAISE'04 FORUM, pp. 83-92.
12. Marcos, E., Vela, B., Cavero, J.M.: Methodological Approach for Object-Relational Database Design using UML. Journal on Software and Systems Modeling (SoSyM). Springer-Verlag. France, R., Rumpe, B. (eds.) ISSN: 1619-1366. Volume SoSyM 2, pp.59-72, 2003.
13. Miller, J., Mukerji, J., MDA Guide Version 1.0.1. Recuperado en http://www.omr.org/mda.
14. OMG. OMG Unified Modelling Language Specification. Version 1.5. Recuperado de: http://www.omg.org, 2003.
15. Oracle Corporation. Oracle XML DB. Technical White Paper. Recuperado de: www.otn.com, enero, 2003.
16. Vara, J.M., Acuña, C.J., Marcos, E., Lopez, M.: Desarrollo de un Sistema de Información web: una experiencia con Oracle XMLDB. Revista CUORE. Aceptado para su publicación.
17. Vela, B., Cavero, J. M., Marcos, E. Diseño de Bases de Datos Objeto-Relacionales con UML. 1ª Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC'2001). Anales de las Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento. Ed. Silvia Teresita Acuña y Cecilia María Lasserre. Editorial Universidad Nacional del Jujuy, 2001, pp. 59-68. Buenos Aires (Argentina). 13-15 June 2001.
18. Vela, B. and Marcos E., Extending UML to represent XML Schemas. The 15th Conference On Advanced Information Systems Engineering. CAISE'03 FORUM. Klagenfurt/Velden (Austria). 16-20 June 2003. Ed. J. Eder, T. Welzar. Short Paper Proceedings. ISBN: 86-435-0549-8. 2003.
19. Paul Harmon: The OMG's Model Driven Architecture and BPM. Newsletter of Bussines Process Trends. Vol 2 (5), Mayo 2004.
20. Westermann, U. y Klas W. An Analysis of XML Database Solutions for the Management of MPEG-7 Media Descriptions. ACM Computing Surveys, Vol. 35 (4), pp. 331-373, diciembre, 2003.
21. Winter, A., Kullbach, B., Riediger, V., An Overview of the GXL Graph Exchange Language. Software Visualization, International Seminar, Dagstuhl Castle, Alemania. Mayo 2002.
22. W3C Web Services Description Language (WSDL) Version 1.2: Bindings. W3C Working Draft 3, Marzo 2003. Recuperado de http://www.w3.org.