Módulo Generador de código PHP para la herramienta SODRA a partir del intermediario XMI

S. E. Cruz Sánchez[1], S. G. Peláez Camarena[2], M. A. Abud Figueroa[3],
L. Rodríguez Mazahua[4], U. Juárez Martínez[5].
División de Estudios de Posgrado e Investigación
Instituto Tecnológico de Orizaba
Orizaba, Ver., México, 94300
esthefany.cruz.sanchez@gmail.com[1], sgpelaez@yahoo.com.mx[2],
{mabud[3], lrodriguezm[4], ujuarez[5]}@ito-depi.edu.mx
***Área de participación**: Investigación Educativa*

Resumen

Buena parte de las metodologías que se emplean en el modelado de aplicaciones Web se basan en el lenguaje de modelado unificado (UML) con un enfoque Orientado a Objetos, lo cual implica para los desarrolladores tener conocimiento amplio en el uso y gestión de los diagramas implementados para este lenguaje de modelado. Dado que el uso de UML es muy importante en el desarrollo software, lo cual permite entregas en tiempo y forma de los productos. Una característica importante de las aplicaciones Web es su constante evolución, por lo tanto, requieren desarrollarse a corto plazo. Por lo que la generación automática a partir de modelos es importante para reducir el tiempo de desarrollo en código esqueleto. En este artículo, se presenta la propuesta del proceso de desarrollo de un generador de código a partir de un archivo XMI, el cual es resultante de los modelos de cliente y navegacional.

***Palabras clave**: Desarrollo dirigido por modelos, Aplicaciones Enriquecidas de Internet, Lenguaje Unificado de Modelo, Intercambio de Metadatos XML.*

Abstract

A large part of the Web application modeling methodologies are based on the unified modeling language (UML) with an Object-Oriented approach, which implies for developers to have extensive knowledge in the use and management of diagrams implemented for this modeling language. Since the use of UML is very important in the development of software, which allows products to be delivered on time and form. An important feature of Web applications is their constant evolution, therefore, they need to be developed in the short term. So automatic generation from models is important to reduce development time in structural code. In this article, we present the proposal for the development process of a code generator from an XMI file, which is the result of client and navigation models.

***Key words**: Model-driven development, Rich Internet Applications, Unified Modeling Language, XML Metadata Interchange.*

Introducción

Hoy en día las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería Asistida por Computadora) se utilizan en el área educativa para el desarrollo de modelos que sirven como apoyo para el desarrollo de proyectos. Las herramientas CASE muchas veces permiten el diseño de modelos o diagramas de manera libre, pero escasas veces permiten la generación de código de forma gratuita, imposibilitando a los estudiantes el acceso al código fuente de los modelos, puesto que las licencias de estas herramientas tienen un costo muy elevado para liberar esa opción.
Es por eso que SODRA (Sistema de Objetos de Diagramación Relacional Amigable) en su tercera fase tiene como objetivo permitir la generación de código automático a partir de los modelos de cliente y navegacional, de manera gratuita y al alcance de todos.
La generación automática de código es una parte importante para el desarrollo de aplicaciones de software basadas en modelos, es esencial para los desarrolladores, ya que esto ayuda a utilizar el tiempo invertido de

manera racional; actualmente la producción de software es muy demandada, por lo tanto, su desarrollo debe ser rápido, sin dejar de lado los aspectos de calidad, eficiencia y eficacia, que estos requieren, por lo que, es deseable que los desarrolladores cuenten con alternativas que ayuden con el ahorro y optimización del tiempo.
La problemática que se plantea en este este artículo se relaciona con la herramienta SODRA, misma que, aunque SODRA permite la creación de los modelos del cliente y navegacional de forma interactiva, no cuenta con mecanismos para la generación de código, ya que se trata de una herramienta en proceso de desarrollo.
El tercer módulo será encargado de la traducción de los archivos XMI a código fuente. Este trabajo inicia con los documentos de entrada generados para los modelos del cliente y navegacional resultante de la herramienta SODRA en formato XMI, a partir de los cuales se generará código automático, como base para la aplicación modelada, en el lenguaje PHP.

El desarrollo de aplicaciones Web es un proceso que requiere tiempo y esfuerzo para su adecuada construcción, por lo que Carreón-Díaz [1] plantea el desarrollo de una arquitectura y de un prototipo de la herramienta SODRA que sirvan como base para el desarrollo de una aplicación Web, que permita la generación de diagramas para el modelo del cliente y navegacional. Se presenta un avance de la herramienta para los diagramas del cliente y navegacional.

En [2] se encontraron como contribuciones: una metodología de desarrollo para aplicaciones Web que admitió pruebas de semántica verdadera basada en navegador dentro de un IDE potente, para todos los navegadores seleccionados por el desarrollador. Además, el uso de objetos proxy para acceder a objetos basados en el navegador y la semántica desde el lenguaje principal durante el desarrollo. También el uso de objetos de proxy inverso para acceder al código del host desde eventos basados en navegador y bibliotecas durante el desarrollo. Así mismo, una transición perfecta del entorno de desarrollo al código totalmente basado en navegador para pruebas finales y la implementación. El código resultante se pudo implementar dentro del navegador debido a la traducción de alta fidelidad del lenguaje de desarrollo (Smalltalk) a JavaScript de calidad de producción.

Roubi et al. [3] al encontrar como problemática que las RIAs son aplicaciones complejas y su desarrollo requiere un diseño e implementación que consume mucho tiempo y las herramientas disponibles están especializadas en el diseño manual, se definió un Metamodelo para RIA que respeta el patrón de Model View Presenter y se desarrolló el motor de transformación que permitió la generación automática del modelo de salida. La principal contribución en el enfoque propuesto fue la simplificación de la IFML y la extracción de detalles técnicos.

La mayoría de los trabajos relacionados abordan un desarrollo basados en MDA (*Model-Driven Architecture*, Arquitectura Dirigida por Modelos), considerándose la generación automática de código esqueleto en distintos lenguajes, lo que permite establecer que, a partir de modelos es viable y factible su interpretación para producir como salida una estructura básica en un lenguaje de programación específico.

Como propuesta de este artículo, se consideran la generación automática de código esqueleto a partir de modelos, dado que, para poder realizar la generación de código automático, debe implementarse un mecanismo para la interpretación de un documento XMI (*XML Metadata Interchange,* Intercambio de Metadados XML) resultante del modelado de cliente y navegacional, a un lenguaje de salida particular, en este caso el lenguaje PHP.

Metodología

De acuerdo al planteamiento del problema y considerando la disponibilidad de los recursos tecnológicos para el desarrollo del producto objeto de esta investigación, así como el grado y curva de aprendizaje para el desarrollo de una herramienta que genere automáticamente código esqueleto ejecutable, se considera la utilización del lenguaje de programación Java, ANTLR (*ANother Tool for Language Recognition*, Otra Herramienta para el Reconocimiento de Idiomas) [4], PHP, XMI y OOHDM (*Object Oriented Hypermedia Design Method*, Método de Diseño de Hipermedia Orientado a Objetos) [5] como las tecnologías para el desarrollo del módulo, objetivo de este artículo. La solución propuesta se enfoca en desarrollar una aplicación Web que genere código esqueleto

automático a partir de archivos XMI del modelado de cliente y navegacional, resultantes de la herramienta SODRA. Mediante el uso de la aplicación SODRA, se generará el documento XMI para su procesamiento, y de esta manera, la integración del módulo a la aplicación, que genere el código esqueleto correspondiente, esto se plantea alcanzar, mediante el uso de la gramática desarrollada en ANTLR que implemente la capacidad de procesar el contenido del documento XMI y con programación en el lenguaje Java, generar el código de salida estructurado con base a las reglas que rigen al lenguaje PHP.

Se espera obtener como salida el código esqueleto del modelo planteado, lo que producirá como beneficio al programador poder dedicar mayor tiempo al aspecto funcional de la aplicación. Este ahorro de tiempo esperado, tendrá como beneficio que los programadores lo dediquen al análisis de soluciones óptimas y reducir los tiempos de entrega.

Como metodología se eligió a OOHDM, ya que esta apoya el desarrollo web, y considera especialmente el modelo navegacional, el cual es uno de los modelos que es posible modelar en SODRA, por lo tanto, su estructura se encuentra en el XMI que se procesará. La tercera fase de desarrollo de SODRA considera las siguientes fases que se muestran en la Figura 1:

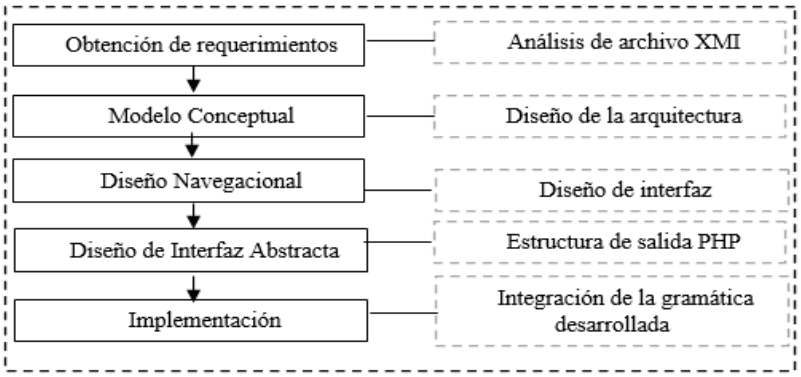


Figura 1. Metodología para el desarrollo del generador de código.

Obtención de requerimientos: El archivo de entrada para la generación de código automático es un archivo XMI resultante de la diagramación con la herramienta SODRA, el cual se analiza para conocer la estructura de sus componentes, los cuales apoyarán a la traducción al lenguaje de salida PHP.

Modelo conceptual: En esta etapa, se contempla el diseño de la arquitectura para el desarrollo de la aplicación, donde se utilizará un patrón arquitectónico Modelo – Vista – Controlador (MVC).

Diseño navegacional: En esta etapa, se realiza el mapeo entre los elementos de los diagramas de modelado de cliente y navegacional con los elementos semejantes en el lenguaje PHP, con el fin de encontrar la navegación correcta de la aplicación en PHP a partir del diagrama navegacional de la herramienta SODRA.

Diseño de interfaz abstracta: En esta sección se enfoca en los prototipos necesarios hasta alcanzar el diseño deseado de la interfaz de la aplicación encargada de la generación de código automático.

Implementación: El desarrollo de la gramática encargada de analizar y validar el archivo de entrada XMI, está contemplado en esta etapa, la cual será desarrollada con ANTLR4, la cual apoyará a la construcción del compilador en Java, el cual será capaz de dar una salida en lenguaje PHP.

Como lenguaje de programación se seleccionó a Java, el cual es un lenguaje de programación bastante robusto para el desarrollo de esta herramienta, este lenguaje se combinará con una gramática que se desarrollará en ANTLR, la que será la encargada de reconocer la estructura del archivo XMI, que proporciona como resultado SODRA. Como lenguaje de salida del generador se seleccionó a PHP, considerando que es un lenguaje adecuado para el desarrollo web.

Resultados preliminares y discusión

Gámez [6] da paso a una nueva forma de solución al hacer el análisis de requerimientos, por medio de los modelos de cliente y navegacional a través de grafos y teoría de conjuntos. Este enfoque apoya a la compresión más precisa y clara de los clientes acerca de los requerimientos que ellos plantean, visualizándolos en los diagramas de cliente y navegacional. Los artefactos que el autor propone se presentan a continuación. En la Tabla 2 se presentan las reglas para el modelado navegacional.

Modelado de cliente
A continuación se presentan las reglas para el modelado del cliente:

Regla 1: La letra inicial del nombre de la clase se debe escribir con mayúscula, los atributos y métodos se deben escribir con sus respectivas llaves separados por comas como se establece en la notación de la teoría de conjuntos, por ejemplo:

Persona {{nombre, apellido, edad}, {caminar( ), correr( )}}

Regla 2: La ausencia de atributos o métodos en la definición de una clase se representa con el símbolo de conjunto vacío "**Ø**", ejemplo:

Persona {**Ø**, {caminar ( ), correr ( )}}

Regla 3: Para la representación de una clase abstracta se antepone "Abs_" al inicio del nombre de la clase, por ejemplo:

Abs_Persona {{nombre, apellido, edad}, {caminar( ), correr( )}}

Regla 4: Para representar una interfaz se antepone "In_" al inicio del nombre de la clase, ejemplo:

In_Persona {Ø, {caminar( ), correr( )}}

Regla 5: Para una clase que implementa una interfaz, se representa utilizando el símbolo de implicación "⇒", ejemplo:

Alumno ⇒ In_Persona

Regla 6: La herencia se representa utilizando el símbolo "∪" para unión de conjuntos, ejemplo:

Clase 1: Persona{{nombre, apellido},{caminar( )}}
Clase 2: Alumno{{numctrl, carrera},{estudiar( )}}
"Alumno hereda de persona", se representa como:

Alumno ∪ Persona

Regla 7: La composición y agregación de clases, se representan utilizando los símbolos "∈" y "⊂" respectivamente. Por ejemplo, para las siguientes clases:

Videoclub {{direccion, telefono},{agregarPel( )}}
Socio{{id, nombre},{actualizarDatos( )}}
SocioVip{ **Ø**,{aplicaDesc( )}}
Pelicula{{idp, nombre, genero},{actualizarInfo( )}}

La representación de la composición se muestra en el siguiente ejemplo utilizando el símbolo "∈":

Película ∈ Videoclub

El ejemplo a continuación muestra cómo representar la agregación mediante el símbolo "⊂":

Socios ⊂ Videoclub

Regla 8: Para establecer la asociación entre clases, utilizar el símbolo de intersección "∩", por ejemplo para las siguientes clases:

Cliente ∩ Hotel

La notación anterior significa que existe la asociación entre la clase "Cliente" y la clase "Hotel", la palabra "busca" sobre el símbolo de asociación especifica el tipo de relación que existe entre las dos clases; para el ejemplo se lee "Cliente busca hotel".

 ISSN 2007-8102

Regla 9: Para indicar la multiplicidad (en agregación, composición y asociación), realizarlo como en el siguiente ejemplo:

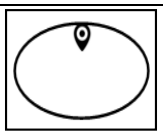
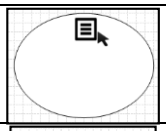
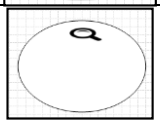
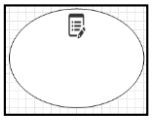
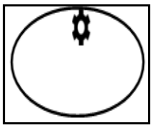
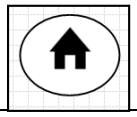
Busca
Cliente 1..* ∩ 1..* Hotel

El ejemplo anterior se lee; "Uno o más clientes buscan uno o más hoteles".

A continuación se presentan las reglas para el modelado navegacional:

Modelado navegacional

Tabla 2 Notaciones con grafos del modelo navegacional

Nombre	Símbolo	Descripción
Navigation Node		Representa un nodo de navegación, es decir, un punto de la navegación en el que el usuario puede trabajar con la información, recuperar o modificar datos en el sistema.
MenuNode node		Representa un menú, es decir, sirve para manejar caminos alternativos de navegación.
Query Node		Representa puntos de la navegación donde el sistema solicita información al usuario que es esencial para continuar con la navegación.
Index Node		Representa puntos de navegación donde al usuario se le facilita una lista de posibles resultados a visualizar, todos referidos a la misma información.
Process Node		Representan procesos/tareas a realizar en la aplicación.
Navigation Link		Representan un link de navegación de un nodo de navegación a otro, no puede usarse con nodos de proceso.
Process link		Representa un link de proceso que va de un nodo de proceso a otro, no se puede usar para nodos de navegación (para nodos de navegación se usan links de navegación).
User Item		Sirve para indicar qué usuario tiene acceso a determinado destino de navegación
Home Page		Representa el punto inicial de la navegación, es decir, la página de inicio de la aplicación.

 ISSN 2007-8102

Con base en los artefactos anteriormente mencionados, se desarrolló una herramienta que incluye diferentes tipos de funcionalidades, tales como, copiar, pegar, cortar, pegar, cortar, bloquear, desbloquear, selección múltiple de elementos, generación de enlaces entre los nodos así como la aplicación del patrón de diseño Memento para deshacer y rehacer acciones sobre los lienzos. A continuación en la Figura 2 se muestra el prototipo de la herramienta. En la Figura 2 se muestra un ejemplo de un diagrama del cliente.

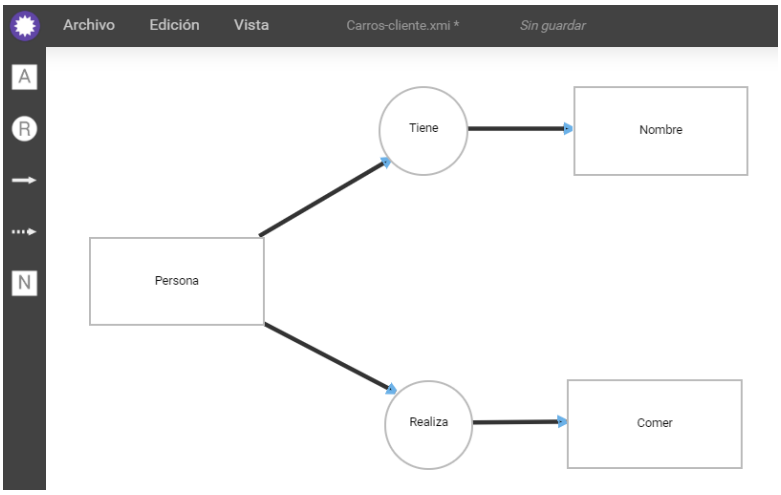


Figura 2. Prototipo del diagrama del cliente.

En la Figura 3 muestra la notación que equivale al diagrama del cliente.

Definición de clases:

Persona {{Nombre},{Comer}}

Figura 3. Notación equivalente al diagrama del cliente.

En la Figura 4 muestra un ejemplo de diagrama navegacional y la Figura 5 su respectiva notación.

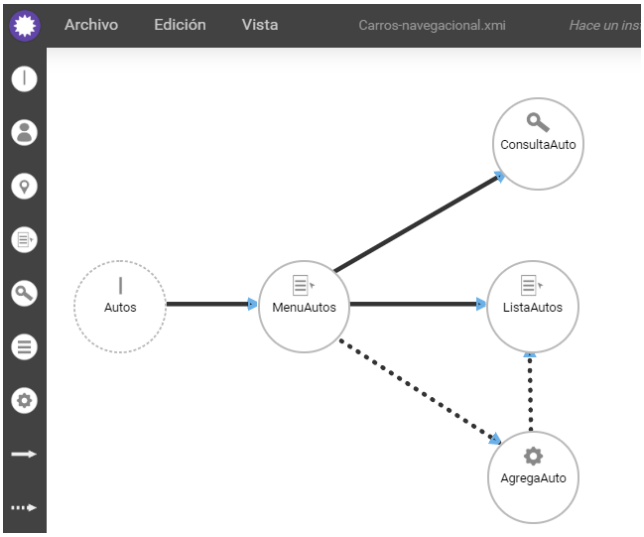


Figura 4. Prototipo de diagrama navegacional

```
{Autos}{NodoInicio} – [EnlaceNavegacional] -> {MenuAutos}{NodoMenu};
{MenuAutos}{NodoMenu} – [EnlaceNavegacional] -> {ListaAutos}{NodoLista};
{MenuAutos}{NodoMenu} – [EnlaceNavegacional] -> {ConsultaAutos}{NodoConsulta};
{MenuAutos}{NodoMenu} – [EnlaceProceso] -> {AgregarAutos}{NodoProceso};
{AgregarAutos}{NodoProceso} – [EnlaceProceso] -> {ListaAutos}{NodoLista};
```

Figura 5. Notación del prototipo de diagrama navegacional

La herramienta SODRA es capaz de generar diagramas de cliente y navegacional de manera interactiva, que servirán para desarrollar un caso de estudio que valide la integración de los clientes al proceso de análisis de requerimientos en el desarrollo de sus aplicaciones web. Las funcionalidades que permite realizar esta herramienta como copiar, pegar, arrastrar, agrupar, exportar, importar, hacer, deshacer, guardar y entre otros, apoya a la curva de aprendizaje con respecto a otras herramientas que utilizan las mismas técnicas de usabilidad.

Trabajo a futuro

Como trabajo a futuro, se considera la generación de código en otros lenguajes de programación (Java Server Faces, .Net, entre otros), para enriquecer la herramienta SODRA, no limitándola a un solo lenguaje, lo que la convertirá en una herramienta mucho más robusta y eficiente.

Conclusiones

La integración de los artefactos y SODRA abren camino a la mejora de la herramienta, lo que hace relevante la generación código automático para enriquecer la herramienta. Los generadores de código automáticos son herramientas de apoyo muy importantes, ya que favorecen a los desarrolladores a dedicarse a las áreas de análisis más complejas invirtiendo el tiempo ahorrado con la generación de código esqueleto. Los generadores de código resultan ser un gran beneficio que apoya también a las empresas dedicadas al desarrollo de software, ya que de esta manera tendrán una mayor probabilidad de entregar en tiempo y forma los productos solicitados por el cliente, Lo que resulta en beneficios económicos. Considerar la creación de gramáticas encargadas de validar el archivo XMl resultante de SODRA, es muy relevante, ya que esta es realmente la encargada de reconocer los elementos a traducir al lenguaje de salida.

Agradecimientos

Los autores agradecen al Consejo Nacional de Ciencia y Tecnología (CONACyT) y al Tecnológico Nacional de México (TecNM).

Referencias

[1] F. Carreón-Díaz de León, "SODRA Arquitectura de una herramienta para crear diagramas del modelo cliente y navegacional", Desarrollo e implementación de las ciencias computacionales, pp. 104-111, 2019.

[2] N. Bouraqadi and D. Mason, "Mocks, Proxies, and Transpilation as Development Strategies for Web Development", Proceedings of the 11th edition of the International Workshop on Smalltalk Technologies - IWST'16, pp. 1-6, 2016. Available: 10.1145/2991041.2991051.

[3] S. Roubi, M. Erramdani and S. Mbarki, "Extending Graphical Part of the Interaction Flow Modeling Language to Generate Rich Internet Graphical User Interfaces", 2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), pp. 1-7, 2017.

[4] ANTLR", Antlr.org, 2019. [Online]. Disponible: https://www.antlr.org/. [Accedido: 04- Ago- 2019]

[5] A. Carrillo Ramos, Herramienta Multimedia De Apoyo A La Enseñanza De La Metodología Rup De Ingeniería Del Software, 1st ed. 2009, pp. 10-15.

[6] C. Lima Gámez, "Propuesta de artefactos basados en una notación con grafos y conjuntos para el modelado conceptual de aplicaciones Web", Research in Computing Science, vol.107, pp. 41-50, 2015.