



Una Arquitectura Modular para el desarrollo de un IDE que apoye a la Enseñanza de los Fundamentos de la Programación Orientada a Objetos

Carlos Omar Meza Ortiz¹, Rafael Rivera López², José de Jesús Ceballos Mejía³ y Abelardo Rodríguez León⁴

Resumen — En el presente artículo se describe la arquitectura modular requerida para la construcción de una herramienta de software que aplica la metodología de seis pasos para la enseñanza de los fundamentos de la programación orientada a objetos (POO) en una institución de educación superior (IES). Esta herramienta es un entorno de desarrollo integrado (IDE) que se construye como una aplicación de cliente enriquecida (ACE) basada en el desarrollo de un conjunto de módulos utilizando la plataforma de desarrollo de NetBeans. Esta herramienta forma parte de la jerarquía de módulos del proyecto “IDE TotalJava” que actualmente está en desarrollo por el programa de Maestría en Tecnologías de la Información en el Instituto Tecnológico de Tepic y que utiliza la metodología basada en UML y Java que se desarrolló en el Instituto Tecnológico de Veracruz.

Palabras claves— Arquitectura, módulo, metodología, UML, POO.

Introducción

Uno de los principales desafíos en la enseñanza de la Programación Orientada a Objetos (POO) en alumnos de nivel medio superior y superior es la interpretación de conceptos como encapsulado, herencia y polimorfismo, así como el desarrollo de la lógica de un programa que manipule objetos. El lenguaje de modelado unificado (UML) es una de las herramientas de modelado más utilizadas para la definición de la arquitectura de una aplicación que utiliza una gran variedad de diagramas (clases, secuencias, actividades, componentes, etc.) que describen las vistas estática y dinámica de una aplicación de software¹. Sin embargo, la gran variedad de estos diagramas tiende a complicar la enseñanza de la POO, ya que la mayoría están diseñados para ser utilizados por desarrolladores expertos.

La Metodología de Seis Pasos para la Enseñanza de la POO aplicada a alumnos del Instituto Tecnológico de Veracruz proporciona una solución a esta problemática bajo una serie de conceptos prácticos para la creación de diagramas UML y su conversión a código Java organizados en un conjunto de seis pasos que van desde la apreciación lógica de una aplicación hasta la generación de clases con sus atributos y la implementación completa de

¹ Carlos Omar Meza Ortiz es alumno de la Maestría en Tecnologías de Información del Instituto Tecnológico de Tepic, Nayarit, México. omar.8x@gmail.com (autor correspondiente)

² Rafael Rivera López es profesor-investigador del Departamento de Sistemas y Computación del Instituto Tecnológico de Veracruz, Veracruz, México. rafaelriveralopez@gmail.com

³ José de Jesús Ceballos Mejía es Maestro de la Maestría en Tecnologías de Información del Instituto Tecnológico de Tepic, Nayarit, México. cemj73@hotmail.com

⁴ Abelardo Rodríguez León es profesor-investigador del Departamento de Sistemas y Computación de Instituto Tecnológico de Veracruz, Veracruz, México. arleonver@gmail.com

todos sus métodos en lenguaje Java. Cabe mencionar que esta metodología no cuenta actualmente con una herramienta de software que apoye en el proceso de enseñanza-aprendizaje utilizando los seis pasos propuestos en ella².

Por otro lado, en el Instituto Tecnológico de Tepic se realiza el proyecto TotalJava donde se construye un entorno de desarrollo integrado (IDE) que tiene como objetivo ofrecer una herramienta práctica y dinámica para alumnos inexpertos como apoyo al proceso de enseñanza-aprendizaje de la programación de aplicaciones con Java³. La metodología UML de seis pasos se integra a la jerarquía de módulos de este proyecto dadas las características de enseñanza de la POO con UML referente a la lógica de programación. Esta metodología obtiene como producto las clases Java que en conjunción con el IDE de TotalJava apoyará en la construcción de la funcionalidad interna con un conjunto de componentes gráficos Swing.

Es por ello que el motivo del presente artículo es presentar la arquitectura modular para la construcción de una herramienta de software que aplica la metodología de seis pasos para la enseñanza de los fundamentos de la POO e integrarla dentro del desarrollo de TotalJava. La unión de estas dos herramientas disminuirá el tiempo de desarrollo y al entendimiento práctico-lógico de una aplicación.

Metodología de seis pasos para la enseñanza-aprendizaje de los fundamentos de programación

La metodología de seis pasos para la enseñanza de los fundamentos de programación es desarrollada en seis etapas. Las primeras tres conforman la fase de modelado con UML y las otras conforman la fase de codificación en Java. En la figura 1 se muestran las seis etapas de esta metodología. Esta propuesta busca evitar la complejidad inherente del diseño orientado a objetos utilizando toda la potencialidad de UML para los estudiantes sin experiencia. Para alcanzar los objetivos de un curso inicial de computación se proponen las seis etapas de forma que los alumnos adopten una metodología básica para construir soluciones usando el paradigma orientado a objetos. En la tabla 1 presentamos una descripción general de cada etapa de esta metodología.

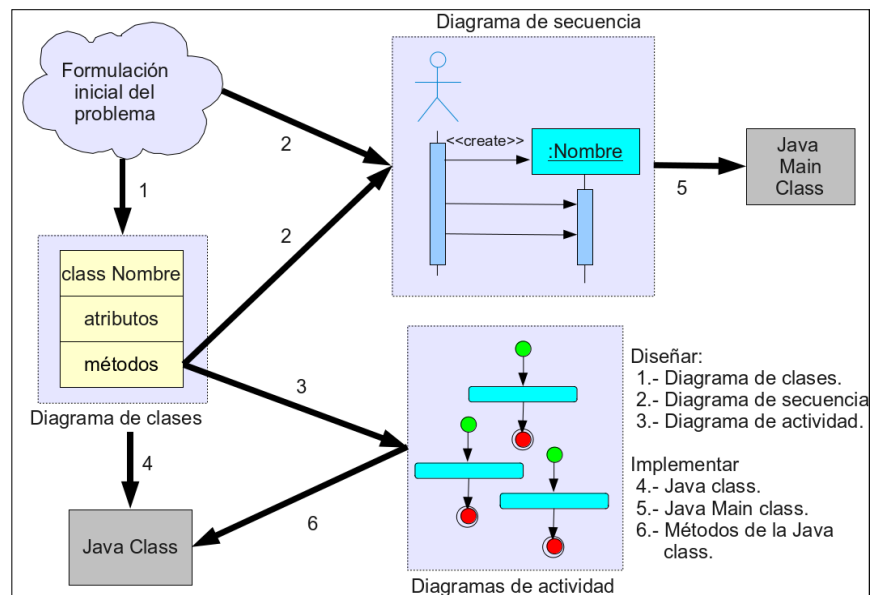


Figura 1.- Etapas de la metodología de seis pasos para la enseñanza de los Fundamentos de POO.

Desarrollo Modular

Frecuentemente programadores de software no visualizan la magnitud de las aplicaciones que están desarrollando; por ejemplo, se comienza a crear una aplicación de tipo Web o escritorio y es entonces cuando se percibe la magnitud del proyecto por la cantidad de paquetes y clases que se generan para la aplicación, adicionalmente de las clases proporcionadas por las librerías de terceros. Una solución a esta problemática es el desarrollo de aplicaciones en base a módulos o servicios utilizando tecnologías conocidas como plataformas de cliente enriquecidas (RCP, por

sus siglas en inglés), que separan las funcionalidades de una aplicación permitiendo el desarrollo paralelo o colaborativo integrándose de forma transparente en una sola aplicación. Para ello existen una serie de proyectos y especificaciones que permiten implementar el desarrollo modular en las aplicaciones tales como OSGi (*Open Services Gateway Initiative*), la plataforma de desarrollo de NetBeans, JPF (*Java Plugin Framework*) y JSPF (*Java Simple Plugin Framework*), entre otros^{4,5}. A continuación se describen las especificaciones de OSGi y de la plataforma de NetBeans 6.9, que son las más representativas en el desarrollo de aplicaciones de cliente enriquecidas.

OSGi

La implementación de un desarrollo modular basado en servicios es una de las novedades aplicadas al desarrollo de sistemas Web o de escritorio complejos con un gran número de clases interactuando. OSGi ha sido creado para generar un entorno de software de colaboración. Con frecuencia se crea una aplicación sin tener dimensionada la cantidad de funcionalidades que se implementarán a futuro. Es por ello que este tipo de marco de trabajo (*framework*) brinda la posibilidad de controlar las aplicaciones de una manera transparente y sin problemas de integración^{5,6}. Un ejemplo de una implementación modular es la de los repositorios de las distribuciones de Linux como Ubuntu y Linux Mint, en donde son descargados los paquetes necesarios para instalar una nueva aplicación, actualizando, moviendo o eliminando paquetes de forma transparente, en donde programadores de diversas áreas brindan de forma colaborativa a la comunidad de la distribución Linux la posibilidad de actualizar automáticamente la versión del sistema operativo.

Tabla 1.- Etapas de la metodología de seis pasos para la enseñanza de los Fundamentos de POO.

Etapa	Descripción
1.- Diagrama de clases	A partir de la formulación inicial del problema, utilizando la abstracción, se identifica al objeto principal de problema, así como sus características y su posible comportamiento. Las características son representadas como variables y el comportamiento es representado por funciones o métodos.
2.- Diagrama de secuencia	Un diagrama de secuencia es utilizado para indicar el orden en el cual los mensajes son enviados a los objetos que existen en el problema. Los diagramas de secuencia son utilizados para modelar el secuencia principal de operaciones en el programa.
3.- Diagramas de actividad	Los diagramas de actividad son utilizados para diseñar la lógica de cada método de una clase. La razón por la cual se asocia un diagrama de actividad a cada método que modela el comportamiento una clase es para simplificar el desarrollo de los algoritmos para el estudiante, evitando la construcción de soluciones complejas en un solo diagrama de actividad.
4.- Diseño de la clase instanciable	Utilizando el diagrama de clase diseñado previamente, se codifica un programa en lenguaje Java. Este programa contiene la clase que será usada para crear los objetos que interactúan en el sistema, por lo que se identifica como “clase instanciable”. Los alumnos deberán entender que en ambos casos un lenguaje es utilizado: para el diagrama de clases se usa un lenguaje gráfico y para el programa en Java se utiliza un lenguaje de programación.
5.- Diseño de la clase aplicación	Utilizando el diagrama de secuencia diseñado en la segunda etapa de este proceso, los alumnos deberán traducir este diagrama en una clase llamada “clase aplicación” donde el método main() será ubicado y los mensajes incluidos en el diagrama de secuencia serán implementados. En esta etapa es deseable no incluir mensajes complejos, de forma que el alumno identifique inicialmente la relación entre la secuencia de operaciones dentro del método main().
6.- Construcción de la lógica de cada método	Cada uno de los diagramas de actividad diseñados en la tercera etapa son traducidos a código en Java. Previamente a la traducción a un programa en Java, los alumnos deben aprender las reglas para escribir expresiones matemáticas y lógicas. También, los estudiantes deben aprender las reglas de sintaxis de las principales sentencias: secuenciales, condicionales y repetitivas. El código generado se debe colocar dentro de la clase instanciable definida en la etapa 1.

Sin duda las ventajas de la integración de la especificación OSGi, descritas en la tabla 2, se ven reflejadas en el rápido desarrollo de aplicaciones⁷. En la figura 2 se presentan las capas de OSGi que se desarrollan por encima de la plataforma Java y que proporcionan la capacidad de desarrollo modular. Estas capas se describen en la tabla 3.

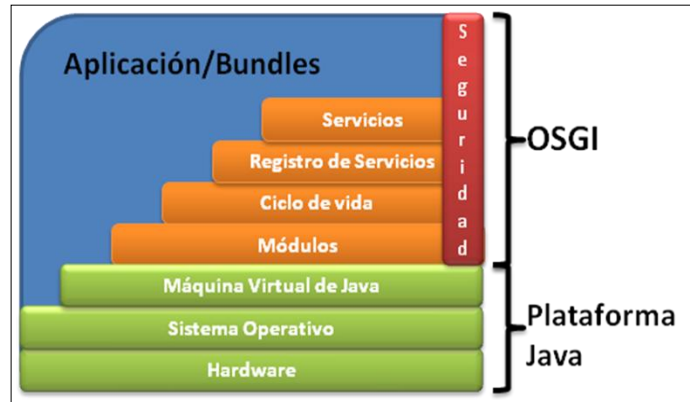


Figura 2.- Capas de OSGi.

Tabla 2.- Ventajas de la especificación OSGi.

Ventaja	Descripción
Reducción de complejidad	OSGi se basa en módulos o paquetes llamados bundles que ocultan sus propiedades a otros paquetes, con la capacidad de actualización, instalación o desinstalación en tiempo de ejecución. Además, por su diseño modular, es mucho más claro detectar los errores dentro de la aplicación.
Reuso	El modelo de componentes OSGi hace que sea fácil de utilizar componentes de terceros en una aplicación. Un número creciente de proyectos de código abierto y comercial proporcionan sus JAR bajo las especificaciones de OSGi.
Desarrollo	Capacidad para definir tu aplicación en partes independientes de forma clara y explícita, analizar, entender y resolver requerimientos dada la independencia entre los paquetes para vistas como funcionalidades dentro del sistema permitiendo el desarrollo colaborativo, y por consecuencia, disminuyendo el tiempo de producción.
Herramientas	La especificación permite la creación de múltiples implementaciones del framework para proporcionar libertad de elección. Algunos proyectos de código abierto son: Apache Félix, Equinoccio de Eclipse, Knopflerfish, entre otros.

Tabla 3.- Capas de OSGi.

Capa	Descripción
Módulos	En su nivel más bajo, la especificación OSGi define un modelo de implementación de los módulos basados en Java. La unidad de despliegue en OSGi se conoce como un conjunto. En lugar de crear un mecanismo de implementación completamente nueva, aprovecha el formato de archivos jar para crear los bundles.
Ciclo de Vida	Una vez que el paquete se instala en un marco de OSGi, el ciclo de vida OSGi gobierna el estado del paquete. Un paquete se puede instalar, iniciar, detener y desinstalar desde el marco, siguiendo el ciclo de vida prescritos por la especificación OSGi.
Registro de servicios	OSGi proporciona también un registro de servicios con la que se puede publicar y/o consumir servicios a través de la máquina virtual de Java. Por lo tanto, OSGi es a veces descrito como "SOA en una JVM."
Servicios	OSGi define también varios servicios básicos que se pueden proporcionar en el marco. Estos incluyen servicios de HTTP, de configuración, etc.
Seguridad	Esta capa se asegura que los bundles sean mostrados mediante una autenticación con firma digital. Además, la capa de seguridad puede apoyar a los permisos de Java para controlar la carga y ejecución de las clases del paquete.

Plataforma NetBeans 6.9

La plataforma de NetBeans ofrece las características de una aplicación de cliente enriquecida en base a su arquitectura modular formada por un conjunto de API's, como se muestra en la figura 3, en donde cada módulo se identifica como una nueva funcionalidad, teniendo la posibilidad de un desarrollo paralelo ya que solo se requiere integrar el módulo a la aplicación en desarrollo⁸. Algunas de las características importantes de la plataforma son:

- Marco de trabajo para la interfaz gráfica basada en el paquete Swing.
- Editor de datos.
- Personalización de pantalla.
- Asistentes.
- Sistema de datos.
- Internacionalización.
- Ayuda del sistema.
- Facilidad de integración.
- Alto grado de cohesión y coherencia.

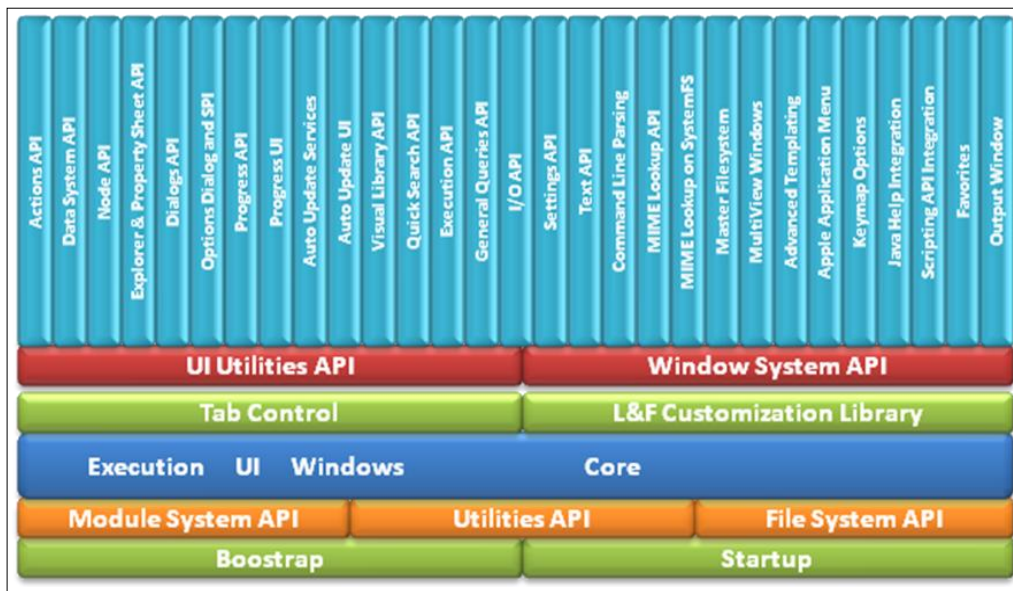


Figura 3.- Arquitectura de la plataforma de desarrollo de NetBeans.

Un módulo o *plugin* es un archivo Jar con atributos especiales en su archivo manifiesto^{9,10}. En el IDE de NetBeans 6.9 se cuenta con un asistente (*wizard*) para realizar cuatro tipos de módulos, que se describen en la tabla 4.

Tabla 4.- Tipos de Módulos de la Plataforma de Desarrollo de NetBeans.

Módulo	Descripción
Module	Crea un módulo vacío.
Module Suite	Este consiste en una serie de módulos relacionados formando una sola aplicación.
Library Wrapper Module	Este tipo de módulo es requerido cuando se tiene construido un Jar y se quiere definir como módulo. Otros módulos en la misma suite puede declarar dependencias para compartir acceso a la librería de clases.
NetBeans Platform Application	Crea una aplicación de la plataforma de NetBeans vacía, la cual no contiene ningún módulo pero sí una estructura básica para comenzar a integrarlos, con una serie de componentes básicos como el “screen splash” configurable, la adición de módulos, la localización de componentes, entre otros. Todo esto para iniciar nuestra aplicación de cliente enriquecida.

Comunidad de NetBeans Plugin UML.

Existe una comunidad de desarrolladores en NetBeans que trabaja con el *plugin* o módulo UML para el IDE NetBeans 6.9¹¹. Dado el interés en primera instancia de conocer la herramienta surge la idea de integrar el enfoque de la “metodología de seis pasos” a dicho *plugin*, sin embargo, se evalúa la posibilidad de que en cierto momento dado que se cuenta con un líder bajo cierto enfoque y lógica de desarrollo no se cumpla en su totalidad con la implantación de la Metodología UML propuesta. Es por ello que se limita a estudiar el funcionamiento de la herramienta, así como la experiencia en desarrollo que brinde la comunidad en la construcción en base a *plugins* o módulos.

Propuestas para el Desarrollo Modular

Una vez realizado el estudio de la plataforma NetBeans 6.9 y OSGi, y tomando como base la aportación y experiencia en la realización del proyecto TotalJava se concluye que para realizar la aplicación UML en base a la metodología de seis pasos se tienen las siguientes propuestas:

- *Desarrollar una aplicación independiente (standalone) basada en módulos utilizando la plataforma de desarrollo de NetBeans 6.9.* El desarrollo modular con esta plataforma es más rápido y controlado ya que se cuenta con la herramienta para manejo de recursos gráficos y de tecnologías de desarrollo modular para la creación del mismo. Una opción es utilizar el diseñador de componentes propio de la plataforma para la construcción de los diagramas UML haciendo uso de la propiedad de “arrastrar y soltar” desde la paleta de propiedades del entorno, así como la generación de archivos y administración de ventanas de la misma plataforma.
- *Extender el plugin de UML soportado por la comunidad de NetBeans*, para que se implementen las características necesarios para implantar la metodología de seis pasos, tomado en cuenta que ya se tiene acceso al repositorio de la comunidad de desarrollo.
- *Desarrollar una aplicación de cliente enriquecida basada en OSGi*, ya que puede desarrollar una aplicación haciendo uso de diversas tecnologías no propias de la plataforma de desarrollo NetBeans 6.9 utilizando OSGi. El realizar la aplicación de forma independiente implica el uso de tecnologías como:
 - El framework Java Swing, para la creación de la interfaz gráfica.
 - El control Flamingo Ribbon para el desarrollo del menú principal..
 - OSGi y Ant, para un desarrollo modular.
 - MyDoggy, para la construcción de los *dockings* (paneles de propiedades, archivos, entre otros).
 - Toolkit Batik, para la creación de SVG's.
 - Visual Library API, para la elaboración gráfica de los diagramas UML (clases, secuencias, actividades).
 - XML, para la configuración de los diagramas UML generados.

Arquitectura modular aplicada a la Metodología de seis pasos para la enseñanza de los fundamentos de la POO

De acuerdo a la investigación realizada se concluye tomar la plataforma de desarrollo de NetBeans 6.9 para el desarrollo de la herramienta. Para el desarrollo de la aplicación de cliente enriquecida se diseña una arquitectura modular para implementar la metodología de seis pasos. Esta arquitectura se presenta en la figura 4 y la descripción de cada módulo se presenta en la tabla 5.

Desarrollo y validación del primer prototipo

De acuerdo a la investigación realizada y a los módulos propuestos para el desarrollo de la herramienta que integre la Metodología de seis pasos para apoyar la enseñanza de los Fundamentos de programación, la cual se denomina “Metodo6Pasos” se presentan los elementos desarrollados para construir un primer prototipo de dicha aplicación:

- a) Estructura base para el desarrollo modular de la aplicación “Metodo6Pasos”.
- b) Desarrollo de un asistente (*wizard*) para crear un proyecto de tipo UML.
- c) Generación de la estructura de archivos del proyecto.
- d) Generación de la vista gráfica para el desarrollo de los Diagramas UML requeridos.
- e) Desarrollo del panel de componentes UML.

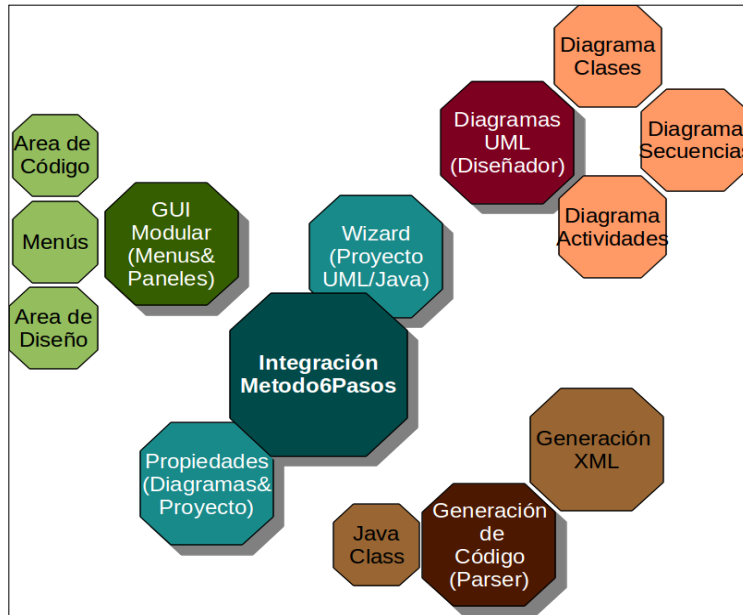


Figura. 4.- Arquitectura de la herramienta “Metodo6Pasos”.

Tabla 5.- Módulos de la herramienta para la metodología de seis pasos.

Módulo	Descripción
Interfaz Gráfica de Usuario	<p>En este módulo se define estructura de la interfaz gráfica general de la herramienta de software como una aplicación de la plataforma de NetBeans. Para el desarrollo de esta interfaz se debe tomar en cuenta lo siguiente:</p> <ul style="list-style-type: none"> - Menús: Propios de la aplicación para el control de acciones a implementar en el diseño y compilación de código de los diagramas UML creados en el proyecto. - Área de diseño: Se define un panel vacío para el diseño de los diagramas UML. - Área de código: En el que mostrarán los archivos XML correspondientes al diseño de los diagramas UML. - Dependencias entre paneles: En este caso se implementarán las dependencias necesarias de la interfaz gráfica para habilitar o deshabilitar paneles de acuerdo al diseño del diagrama UML que se este creando. - ScreenSplash: Para mostrar al inicio de la aplicación en el momento que sea ejecutada por el alumno o usuario. - Instalador: Generar un instalador para opcional a la plataforma (Linux, Windows o MAC).
Diseñador de Diagramas UML	<p>En este módulo se desarrolla el diseñador para la creación de los diagramas UML donde se incluyen los diagramas de clase, diagramas de secuencia y diagramas de actividades. Se tomarán en cuenta componentes gráficos contenidos en una paleta para agilizar el diseño e implementación, además de:</p> <ul style="list-style-type: none"> - Uso de Visual Library para crear la escena (Scene), y el uso de objetos “<i>widjets</i>” para crear los componentes UML y sus relaciones. - Un panel de propiedades por componentes UML dependiendo del diagrama en construcción.
Generación de Código	<p>En este módulo se genera el archivo XML de acuerdo a la metodología de seis pasos desde los diagramas construidos en el diseñador. Esto se posible por la definición de dependencias entre los módulos y por lo tanto, entre los componentes. Se considera lo siguiente:</p> <ul style="list-style-type: none"> - Detección de <i>targets</i> (puntos clave) para la construcción del archivo XML. - Parser Java 1.6 para la compilación del código. - Construcción final de la clase Main contenida en una carpeta adicional al proyecto UML/Java compilado.

Módulo	Descripción
Integración Metodo6Pasos	En este módulo, denominado “Suite” en la plataforma de desarrollo de NetBeans, se definen todos los módulos que forman a la aplicación, sus dependencias, las librerías externas y la configuración de instalación.
Gestor de Asistentes (Wizards)	En este módulo se realiza la creación de asistentes (<i>wizards</i>) que guiarán al usuario para la creación de un proyecto. En el momento de concluir el <i>wizard</i> se crea en el panel de proyectos la estructura del proyecto a nivel de archivos con la carpeta de diagramas, la cual contiene archivos de diseño de los mismos, una carpeta de compilación que contendrá las clases creadas, así como archivos de configuración del proyecto.
Gestor de Propiedades	El módulo “Gestor de propiedades” para el proyecto así como los diagramas que se estén diseñando. Se pretende implementar <i>wizards</i> para la creación de los diagramas en términos de asignación de nombres, características propias del diagrama, por ejemplo: atributos, relaciones, métodos, actores, entre otros. Que ayuden al desarrollo rápido y confiable al usuario.

Estructura base para el desarrollo modular de la aplicación “Metodo6Pasos”

En este punto se crean los componentes Swing que formarán la estructura base del desarrollo de la aplicación “Metodo6Pasos” en relación a la interfaz gráfica y funcionalidades generales, tomando en cuenta lo siguiente:

- Menús (Configuración adaptable a la aplicación).
- Paneles (Archivos, Propiedades y Diseño).
- Barra de Herramientas.
- Screen Splash (Diseño e implementación).
- Look and Feel (Requerido para dar una apariencia diferente a todos los componentes de la aplicación).
- Instalador.

Desarrollo de un asistente (wizard) para crear un proyecto de tipo UML

En este punto se definen los pasos para crear un proyecto UML el cual contendrá los diagramas de clases, de secuencia y de actividades. Generando las carpetas y archivos correspondientes para el diseño y generación de código Java correspondiente al proyecto, como se ve en la figura 5. Se tomaron en cuenta dos pasos para la generación del proyecto “Metodo6PasosProject”:

1. Se define el tipo de proyecto UML y la opción para generar todos los archivos de diseño que contendrá a los diagramas UML mencionados.
2. Se define un nombre y la URL en donde se almacenará el proyecto.

Al finalizar de completar los dos pasos anteriores se generará una carpeta y los archivos requeridos para el proyecto en la URL asignada.

Generación de la estructura de archivos del proyecto

En este punto se define la estructura de carpetas y archivos para detectar los diagramas en desarrollo, además de archivos de configuración del proyecto. Se tomará en cuenta el estudio de la Interfaz Project de la API de NetBeans “org.netbeans.api.project”.

Generación de la vista gráfica para el desarrollo de los Diagramas UML requeridos

En este punto se creará la escena gráfica que contendrá a los componentes UML de acuerdo al tipo de diagrama que se esté creando.

Desarrollo del panel de componentes UML

En este punto se crearán los componentes gráficos UML para utilizarlos en la creación de los diagramas correspondientes, se crearán tres pestañas dedicadas al diagrama de clases, secuencias y actividades.

En la figura 6 se presente el conjunto de componentes gráficos creados durante la construcción del prototipo para la herramienta Metodo6Pasos desarrollada como una aplicación de cliente enriquecida a través de la Plataforma de Desarrollo de NetBeans 6.9.

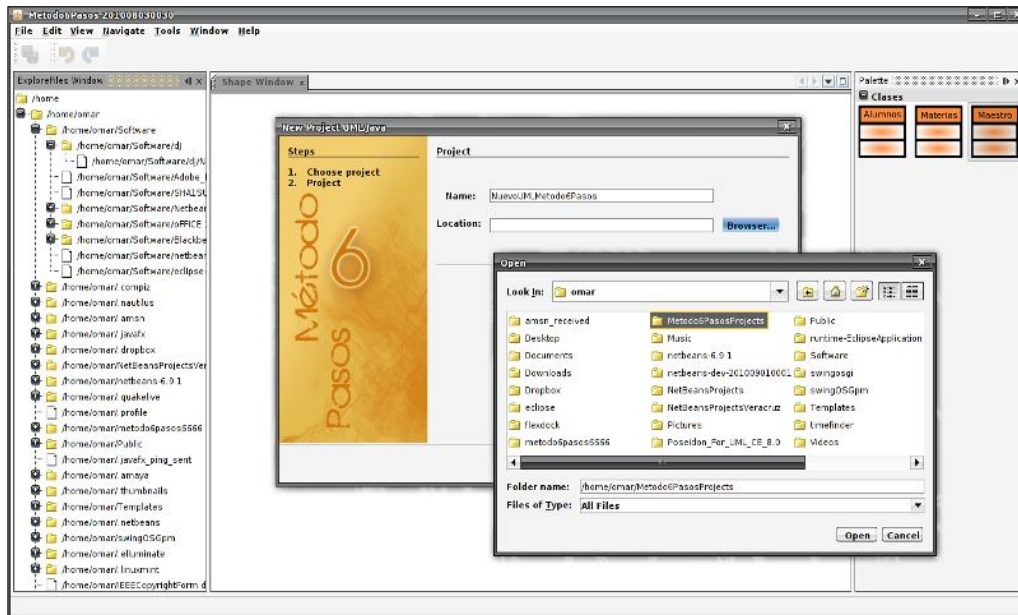


Figura 5.- Prototipo de asistente para la creación de un proyecto del tipo Método6Pasos.

Conclusiones

El desarrollo modular está impactando en la generación de herramientas de software de todo tipo dadas sus características, aunque se requieren conocimientos bien definidos de la conceptualización y el diseño de sistemas. Es por ello que nace la integración de la arquitectura modular mencionada para la realización de la herramienta UML “Metodo6Pasos” tomando como base el desarrollo modular de la plataforma de NetBeans 6.9, por su rapidez, reutilización de módulos, una implementación manejable, detección de errores, soporte a tecnologías OSGi, control de generación de código y la posibilidad de una programación colaborativa. Así como su integración en la jerarquía de módulos del IDE TotalJava que se realiza en la Maestría en Tecnologías de Información del Instituto Tecnológico de Tepic, dado que se adaptará a la especificación OSGi para hacer posible la cohesión de proyectos.

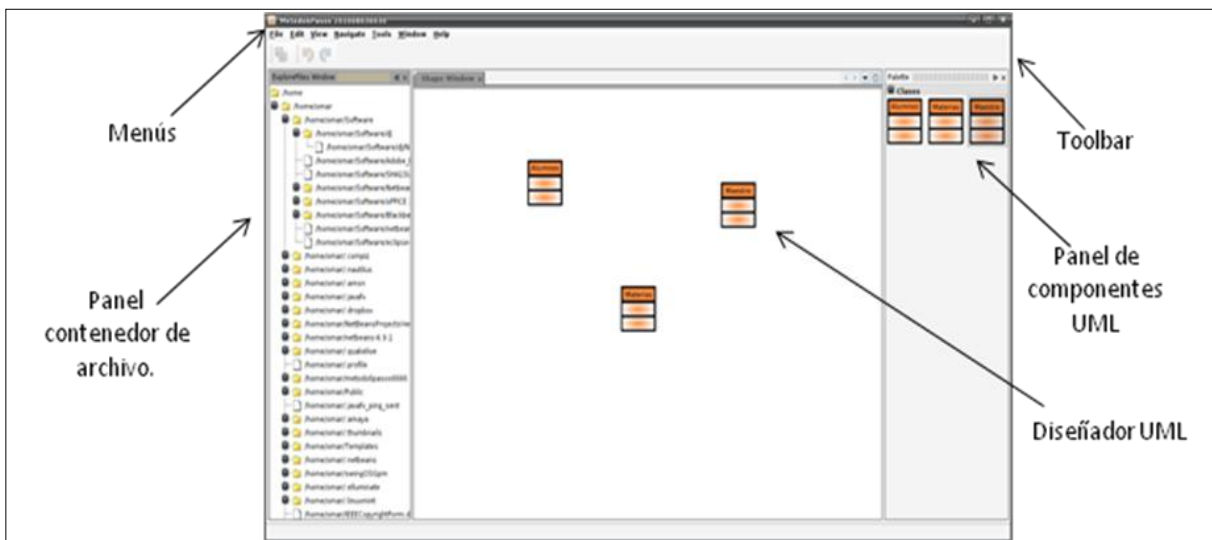


Figura 6.- Interfaz gráfica de la herramienta con los paneles de archivos del proyecto, diseñador, paleta de componentes y menús.

Agradecimientos

Los autores extienden su agradecimiento al Gobierno del Estado de Nayarit, al FOMIX, a CONACyT, y a los Institutos Tecnológicos de Tepic y de Veracruz por los apoyos otorgados para la realización de este proyecto.

Referencias

- ¹ Zayas, C. (2009). Aplicación de NetBeans para la Enseñanza de los Fundamentos de Programación. Tesis de Licenciatura. Instituto Tecnológico de Veracruz.
- ² Rivera, R., Rivera, E., Rodríguez, A. (2009) Another Approach for the Teaching of the Foundations of Programming using UML and Java, *Proceedings of the 3rd WSEAS Int. Conf. on Computer Engineering and Applications (CEA'09)*, pp. 279-283, Ningbo, China.
- ³ Sánchez, M., Ceballos, J. (2009). Análisis de marcos de trabajo para la creación de aplicaciones de escritorio para Java, Instituto Tecnológico de Tepic, Nayarit, México. Congreso Estatal de Ciencia y Tecnología 2009.
- ⁴ JPF Team. [2007] [En línea] [Citado el 29 de 09 del 2010] Java Plug-in Framework (JPF) Project. [<http://jpf.sourceforge.net/>].
- ⁵ Hall, S.R., Pauls, K., McCulloch, S. y Savage, D. (2010) OSGi in Action. Creating Modular Applications in Java.
- ⁶ Walls, C. (2009). Modular Java: Creating Flexible Applications with OSGi and Spring. Editorial Pragmatic Bookshelf.
- ⁷ OSGi Alliance. [2010] [En línea] [Citado el 29 de 09 del 2010] Benefits of Using OSGi. [<http://www.osgi.org/About/WhyOSGi>].
- ⁸ Jürgen, P. (2010) NetBeans Platform 6.9 Developers Guide. Editorial Packt.
- ⁹ Böck, H. (2009) The Definitive Guide to NetBeans Platform 6.5. Editorial Apress.
- ¹⁰ Oracle Corporation and/or its affiliates. [En línea] [Citado el 29 de 09 del 2010] NetBeans Platform Learning Trail. [<http://NetBeans.org/kb/trails/platform.html>].
- ¹¹ NetBeans Community. [En línea] [Citado el 29 de 09 del 2010] UML Project Members. [<http://NetBeans.org/projects/uml/members?field=role&order=asc>]

Notas Biográficas

Carlos Omar Meza Ortíz recibió el título de Licenciado en Informática por el Instituto Tecnológico de Tepic, MEXICO, en 2009. Cursa la Maestría en Tecnologías de la Información en el Instituto Tecnológico de Tepic.

Rafael Rivera López recibió el título de Ingeniero en Sistemas Computacionales por el Instituto Tecnológico de Veracruz, MÉXICO, en 1989, y el grado de Maestro en Ciencias de la Computación por el Instituto Tecnológico y de Estudios Superiores de Monterrey, en Morelos, MÉXICO, en el año 2000. Él está trabajando para obtener el grado de Doctor en Ciencias de la Computación en el Instituto Tecnológico de Estudios Superiores de Monterrey, MÉXICO. Actualmente es Profesor de Tiempo Completo en el Departamento de Computación y Sistemas del Instituto Tecnológico de Veracruz, en MEXICO. Sus intereses de investigación incluyen la programación orientada a objetos, la optimización, la inteligencia artificial y la robótica.

José de Jesús Ceballos Mejía recibió el título de Licenciado en Informática por el Instituto Tecnológico de Tepic, MEXICO, en 1994, y el grado de Maestro en Ciencias de la Computación en la Fundación Arturo Rosenblueth, MEXICO, en el año 2000. Actualmente, es profesor de tiempo completo de la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Tepic. Es responsable técnico ante CONACYT del proyecto de la Maestría en Tecnologías de la Información y del proyecto para la constitución del Centro de Investigación y Desarrollo de Tecnologías de la Información. Sus intereses son el desarrollo de IDE's para Java.

Abelardo Rodríguez León recibió el título de Ingeniero en Sistemas Computacionales por el Instituto Tecnológico de Veracruz, MÉXICO, en 1989, el grado de Maestro en Ciencias de la Computación por la Universidad Veracruzana, MÉXICO, en 1992, y el grado de Doctor en Ciencias Computacionales por la Universidad Politécnica de Valencia, ESPAÑA, en 2007. Actualmente es Profesor de Tiempo Completo en el Departamento de Computación y Sistemas del Instituto Tecnológico de Veracruz, en MEXICO. Sus intereses de investigación incluyen la programación paralela, el cómputo distribuido y la optimización.